Forecasting af finansielle aktiver

Det Samfundsvidenskabelige Fakultet

Aalborg Universitet

Danmark

02-06-2020

AALBORG UNIVERSITY
STUDENT REPORT

Titel:

Forecasting af finansielle aktiver

Projektperiode:

Forårssemester 2020

Projektgruppe:

17

Forfattere:

Alex Korsgaard Pedersen

alexpe15@student.aau.dk

Michael Linddal Thomsen

mlth14@student.aau.dk

Vejleder:

Roman Jurowetzki

Sidetal: 63,25

Afleveringsdato: 2. juni 2020

Inst. for økonomi og ledelse

Aalborg Universitet

https://www.business.aau.dk/

Abstract

In many decades economist and investors have been trying to perfect the disciplin of forecasting financial assests to provide the most profit. This Master's thesis examines the practical challenges of forecasting financial assets, through Artificiel Inteligence and Econometrics. The main purpose of the thesis is to investigate which type of model, that has the best performance in forecasting five equities. This topic has resulted in the problem statement "Can the forecasting performance of equities in selected linear and nonlinear modelles be compared in different portfolio strategies". It resulted in two type of models: Autoregressive Integrated Moving Average model (linear) and Long Short Term Memory (nonlinear), where the results overall were in favor of the nonlinear model LSTM. It was expected from the start of the thesis, that the LSTM model would perform better than ARIMA, due to the different purpose of the two models. ARIMA focuses more on the causality between parameters whereas LSTM focuses on accuracy in the predictions. But even though LSTM performed overall better than ARIMA, and there were significant difference between the two models in four out of the five predictions. Despite the low succesrates in the form of prediction the correct direction of the equity, both models showed that they could beat the passiv portfolio using mean-variance analysis. Overall the master thesis found that LSTM was better than ARIMA and therefore found that there is a different between linear and nonlinear models when it came to predicting the future price of an equity.

Contents

Abstract **iii**

List of Figures **vi**

1 Indledning **1**

1.1 Kunstig intelligens i den finansielle sektor . . . 2

1.2 Machine Learning kontra økonometri . . . 3

1.3 Problemformulering . . . 6

1.3.1 Specialestruktur . . . 6

1.4 Grafer og plots . . . 6

2 Lineære modeller **7**

2.1 Lineær modeller . . . 7

2.2 Stationaritet . . . 8

2.3 Autoregressive Moving Average - ARMA . . . 8

2.4 Autoregressive Integrated Moving Average . . . 9

2.4.1 Unit Root . . . 10

2.4.2 Augmented Dickey-Fuller Test . . . 10

2.5 Evalueringsmetoder . . . 11

2.5.1 Root Mean Squared Error . . . 11

2.5.2 Mean Absolute Error . . . 11

2.5.3 Pesaran Timmermann test . . . 12

2.5.4 Anatolyev & Gerko test . . . 12

2.5.5 Diebold-Mariano test . . . 12

3 Ikke-lineære modeller **14**

3.1 Artificial Intelligence . . . 14

3.2 Machine learning . . . 15

3.3 Neural Networks . . . 18

3.3.1 Back Propagation . . . 20

3.4 Recurrent Neural Network . . . 22

3.5 Long Short Term Memory . . . 25

3.6 Rolling Window . . . 28

3.7 Opsumering . . . 28

4 Litteraturreview **29**

5 Databehandling og forecast **31**
5.1 Deskriptiv statistik . . . 31
5.2 Udvælgelse af ARIMA-model . . . 34
5.2.1 Forecasting ARIMA . . . 36
5.3 Opsætning af LSTM . . . 37
5.4 Udvælgelse af LSTM-modeller . . . 38
5.4.1 Forecasting LSTM . . . 41

6 Sammenligning af ARIMA og LSTM **42**
6.0.1 Opsumering . . . 45

7 Porteføljestrategier **46**
7.0.1 Mean-Varaince analyse . . . 46
7.0.2 Antagelser . . . 47
7.0.3 Resultater . . . 49

8 Diskussion **51**
8.1 Data generering . . . 52
8.2 Local Interpretable Model-agnostic Explanations . . . 53
8.3 Kombination af Machine Learning og Økonomi . . . 53
8.4 Kan data fortælle alt? . . . 54

9 Konklusion **56**

Bibliography **58**

A Bilag **63**

B Appendiks **71**
B.0.1 Opsummering . . . 75

List of Figures

1.1 Udvikling i mængden af data (Kolb, 2019) . . . 1

3.1 Kunstig intelligens overblik . . . 14
3.2 Typer af Machine Learning (Ayodele, 2010) . . . 16
3.3 Arkitektur af neural netværk (Danish, 2019) . . . 18
3.4 Neural netværk med bias-led (Kelleher, 2019) . . . 19
3.5 Aktiveringsfunktioner (Amidi and Amidi, 2019) . . . 19
3.6 Gradient descent (Saiful, 2020) . . . 21
3.7 RNN med input og memory buffer (Kelleher, 2019)) . . . 23
3.8 Cell State . . . 26
3.9 Forget Gate . . . 26
3.10 Input Gate del 1 . . . 27
3.11 Input Gate del 2 . . . 27
3.12 Output Gate . . . 27

5.1 Udvikling i de fem udvalgte aktier . . . 31
5.2 Dagligt afkast af de fem udvalgte aktier . . . 32
5.3 Rullende forecast for 15/11 til 30/12-19 med ARIMA . . . 36
5.4 LSTM opbygning for JNJ . . . 39
5.5 LSTM model opbygning for GOOG . . . 39
5.6 LSTM model opbygning af EQNR . . . 39
5.7 LSTM model opbygning af LB . . . 39
5.8 LSTM model opbygning af CCL . . . 39
5.9 Learning rate . . . 40
5.10 Læringsrate - JNJ . . . 40
5.11 Læringsrate - GOOG . . . 40
5.12 Læringsrate - EQNR . . . 40
5.13 Læringsrate - LB . . . 40
5.14 Læringsrate - CCL . . . 41
5.15 Rullende forecast for 15/11 til 30/12-19 med LSTM . . . 41

6.1 Akkumuleret fejlforcast i pct. mellem ARIMA og LSTM . . . 44

7.1 Mean-Variance analyse . . . 46

A.1 Porteføljesammensætning og udvikling for strategi 1 . . . 64
A.2 Porteføljesammensætning og udvikling for MV-analyse uden shorting . . . 65

A.3 Porteføljesammensætning og udvikling for MV-analyse med shorting 66
A.4 Porteføljesammensætning og udvikling for Buy & sell med ARIMA-forecast 67
A.5 Porteføljesammensætning og udvikling for Buy & sell med LSTM-forecast 68
A.6 Porteføljesammensætning og udvikling for Buy & sell with shorting med ARIMA-forecast . . 69
A.7 Porteføljesammensætning og udvikling for Buy & sell with shorting med LSTM-forecast . . . 70

B.1 Læringsrater 74

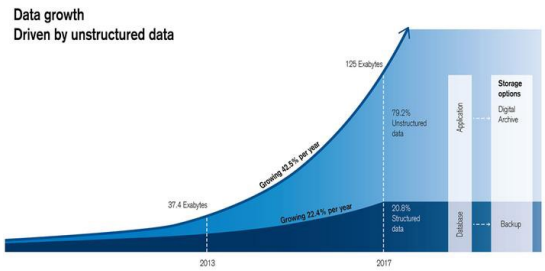
Indledning 1

Modellering og forecasting af tidsserier er ingen let opgave, og især ikke når disse modelleringer anvendes med kompleks information relateret til det finansielle marked, og med hensyntagen til kompleksiteten og de til tider kaotiske dynamikker, som optræder på de finansielle markeder. Derudover afhænger prædikteringer for finansielle tidsserier ikke kun af serien i selv, men ligeledes af makroøkonomisk data og/eller nyhedsdata. Derfor er det endnu vanskeligere at forudsige fremtidige værdier af et finansielt aktiv, da denne type forecast ikke kun er underlagt dataets egen natur, men samtidig de udvalgte variablers natur og kompleksitet.

Med den stigende mængde information tilgængelig med hensyn til hyppigere data, samtidig med hensyn til markedsrelaterede nyheder og information, bliver muligheden for at skabe investeringsstrategier (f.eks. porteføljestyring og -sammensætning) mere og mere kompliceret for investorer, der søger at skabe rentable investeringer. Kompleksiteten eksisterer ikke kun for en investor med hensyn til viden og erfaring, men også for eksisterende algoritmer, der er ansvarlige for at guide, informere eller endda automatisk handle med finansielle aktier. I dette komplekse scenarie, med mere information end en enkelt investor kan håndtere på egen hånd, får forecastingsalgoritmer større betydning ved at skelne irrelevant fra relevant information fra hinanden, således usikkerheden fra dynamiske markedsoplysninger reduceres og det ideelle forecast for en skiftende og ukendt fremtid kan blive realiseret

Anvendte forecastingsalgoritmer er økonometriske eller statistiske metoder baseret på tidligere oplysninger normalt i form af lineære eller ikke-lineære rammer. Blandt disse metoder er kunstig intelligens (AI) Machine learning (ML) en god kandidat, i betragtning af dens præstation på andre områder, herunder billede- og talegenkendelse og samt dens evne til at opnå abstrakt og ikke-lineær information uden forudgående viden om prædikteringen i sig selv. Endvidere er stigningen i mængden af data og computerkraft ført til en opblusning på dette område. Dette har imidlertid skabt en konkurrence mellem de nyere ikke-linære modeller og de eksisterende økonometriske modeller.

Figure 1.1: Udvikling i mængden af data (Kolb, 2019)



En mulig indikation på hvorfor AI har spredt sig, kunne skyldes at verdenen er blevet mere databaseret (Big Data), og som figur 1.1 grafisk illustrer, så er omkring 21 pct. af alt data, der er tilgængeligt til analyse, struktureret. Dermed er potentialet og behovet for at kunne håndtere ustruktureret data til diverse analyser stort.

Grundet det uforløste potentiale indenfor Big Data, er der sket en stor stigning i investering i AI. Ifølge Research Manager Marianne D'Aquila, hos International Data Corporation, er hun ikke i tvivl om, at AI ikke længere befinder sig i startfasen, men er en velfungerende metode, der er klar til implementering i virksomhedernes fremtidige planer (IDC, 2019).

"*Artificial Intelligence (AI) has moved well beyond prototyping and into the phase of execution and implementation. Strategic decision makers across all industries are now grappling with the question of how to effectively proceed with their AI journey. Some have been more successful than others, as evidenced by banking, retail, manufacturing, healthcare, and professional services firms making up more than half of the AI spend...*" (IDC, 2019)

Med det niveau AI befinder sig på, har den finansielle sektor adapteret denne viden og anvender den som et aktiv værktøj i deres forretning, som Marianne D'Aquilla udtaler.

1.1 Kunstig intelligens i den finansielle sektor

Disciplinen i forecasting af finansielle aktiverder har eksisteret i årtier og er endnu ikke blevet mestret til fulde. Det har altid været en investorers formål at tilegne sig en profit gennem investering af finansielle aktiver, hvor forecasting anvendes som et redskab for at opnå denne profit. Eugene Fama fremstillede tilbage i 1970'erne *the efficient capital markets* (ECM), der omhandler, hvordan markedet reflekterer alt tilgængelig information. Når ny information kommer, spredes det til markedet og inkorporeres på et givet finansielle aktiv uden nogen form for forsinkelse. Dette forhindrer investorerne fra at kunne forecaste fremtidigt afkast på aktivet.

> *"The accepted view was that when information arises, the news spreads very quickly and is incorporated into the prices of securities without delay. Thus, neither techincal analysis, which is the study of past stock prices in an attempt to predict future prices, nor even fundamental analysis, which is the analysis of financial information such as company earnings and asset values to help investoes select "undervalued" stocks, would enable an investor to achieve returns greater than those that could be obtained by holding a randomly selected portfolio of individual stocks, at least not with comparable risk."* (Malkiel, 2003)

Gennem de seneste års forskning er tiltroen til ECM, blevet reduceret. Dette fremkommer af, at flere økonomer har til et vist punkt vidst det muligt at forudsige aktiepriser, gennem mønstre i prisudviklingen. Denne tankegang anvendes i ikke-lineære modeller, der i de seneste år har opnået anerkendelse til at forudsige udviklingen i finansielle aktiver. Styrken for ikke-lineære modeller kommer fra naturen i dataet, der oftest fremkommer uforudsigeligt og dermed ikke lineært (Hsieh, 1991). Anerkendelsen af ikke-lineære modeller til forecasting af aktier, kommer på bekostning af de lineære modeller.

Som tidligere beskrevet er der i takt med computerens teknologiske fremskridt, kommet en større efterspørgsel efter Machine learning (ML) og ikke-lineære modeller indenfor finansverden, specielt på finansielle aktiver. Her har det vist sig at neural networks (NN) kan håndtere nogle af udfordringerne i

den finansielle verden, som f.eks. store mængder af data, usikkerhed og gemte/ubekendte sammenhænge. NN's håndteringsevne af ikke-lineære data har ført til fremgang på en række områder i den finansielle verden såsom forecasting, kapitalforvaltning, hvidvaskning og konkursrisiko (Johnson and Whinston, 1998). Det er set over en årsrække, at de nye NN-modeller anvendes på samme niveau som de traditionelle lineære metoder, til analyser inden for finansverdenen. Det er derfor vigtigt at undersøge hvilken type værktøj der bedst kan forecaste f.eks. aktiekurser, obligationer osv., for at skabe den største profit for en investor (Fadlalla and Lin, 2001).

1.2 Machine Learning kontra økonometri

For at kunne sætte Machine Learnings ikke-lineær modeller op mod økonometriens lineære modeller, er det nødvendigt at starte med en definition af hvert element:

"*Econometrics is the quantitative application of statistical and mathematical models using data to develop theories or test existing hypotheses in economics and to forecast future trends from historical data. It subjects real-world data to statistical trials and then compares and contrasts the results against the theory or theories being tested.*" (Hayes, 2019)

"*Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.*" (Expert System Team, 2017)

Således kan de basale sammenligninger begynde. Begge metoder bruger data samt statistiske modeller til at foretage konklusioner. Dermed syntes ML at beskæftige sig med en række bredere spørgsmål end blot økonomi ud fra deres definitioner.
Gennem økonometri er det muligt at være i stand til at kunne bearbejde tidsserie data med et solidt statistisk fundament. De statistiske færdigheder anvendes til at kunne forstå anvendelsen af klassificering og clustering, eller tidsserie prædiktion og regression, såsom linear regression, logistic regression, Kmeans, ARIMA, VAR osv., men netop disse statistiske værktøjer er præcis de samme som ML bruger i deres algoritmer. Så hvad er forskellen egentlig mellem de to modeller?
En grundlæggende forskel mellem ML og økonometri ligger i det teoretiske grundlag. Økonometrien har et solidt fundament i matematik, statistik og sandsynlighedsteori. Algoritmerne er matematisk robuste og forklarer således algoritmens adfærd. En anden forskel er, at økonometri kun har en løsning ud fra en given specificeret model, hvilket betegnes *The best linear unbiased estimator* (BLUE). Et datasæt beregner derfra parametrene til en parametrisk regression ved hjælp af en algebraisk formel givet af den almindelige *ordinary least squares* (OLS) estimat. Her betyder "best", den laveste varians af estimatet sammenlignet med andre unbiased, lineære estimatorer. Mens de fleste ML-algoritmer er for komplekse til at blive beskrevet med en enkelt matematisk formel. Dermed anvender ML en mere empirisk analytisk

tilgang igennem algoritmer, til at estimere og sammenligne mange forskellige alternative modeller. ML tilgangen er i kontrast til den traditionelle økonomisk tilgang, hvor det er individet der vælger en model baseret på økonomiske principper og derefter estimer.

Økonometriske modeller (dvs. parametriske det meste af tiden) er baseret på økonomisk teori og de traditionelle statistiske inferens-værktøjer, (såsom *the maximum likelihood method* og *the current method*) der bruges til at estimere værdierne for en parametervektor, θ, i en parametrisk model, $m\theta$. I ML konstrueres dermed en ikke-parametriske model ofte baseret udelukkende på data (uden antagelser om fordelingen), og de anvendte meta-parametre (*tree depth*, *penalty parameter* osv.) optimereres ved hjælp af *cross-validation*, *grid search algorithm* osv.

I økonometriens verden tales der om *goodness of fit*, der er en robust matematisk test for at validere modellens evne til at finde sammenhænge i dataets mønster. Der anvendes ligeledes R^2, som er andelen af variansen i den afhængige variabel, der er forudsagt fra de uafhængige variabler. Desuden findes der AIC og BIC (Akaike og Bayesian information criterion), der evaluerer kvaliteten af hver model i forhold til andre modeller eller variabelvurderinger gennem p-værdi. Evalueringen af ML-modeller afhænger af dens forudsigelser og dermed er den underliggende idé, at hvis modellen er i stand til at forudsige præcist, så har den med succes lært de skjulte mønstre i dataet. For at sikre, at modellen ikke *overfitter*, bliver datasættet adskilt i et træningssæt og testsæt og derefter anvendes *cross-validation* til at verificere modellens generaliseringsevne, og at der ikke er nogen bias i adskillelsen af data. Endelig vil specialet anvende Key Performance Indicators angiver et mål for gabet mellem træningssættets og testsættets output i form af enten *RMSE* eller *MAE*.

Både økonometri og ML forsøger at definere en funktion, der beregner et sæt prædiktorvariabler, der vil modellere i følgende resultat:

$$X = (x_1, x_2, ..., x_n)$$

$$y = f(X) + \epsilon$$

ϵ er tilfældige uafhængige og normalfordelt variabler, der kaldes residualerne og kommer fra økonometriens verden, ellers hører $y = f(X)$ til ML.

Det kan synes at de to termer er ved at konvergere, men forskellen ligger helt konkret i at ML's formål er at forudsige y, mens økonometrien fokuserer på beregningen af β for forudsigelsen:

$$y = \beta_0 + \beta_1 x_1 + ... + \beta_n x_n$$

Forskellen mellem de to metoder til forecasting af finansielle aktiver ligger derfor i den teoretiske forståelse. I økonometri er formålet at anvende metoder og værktøjer på økonomiske problemstillinger, hvor formålet er at forstå den økonomiske virkelighed gennem verificering af økonomisk teori samt at afklare kausale sammenhænge.

Modsat tager ML udgangspunkt i at kunne prædiktere, og besidder ikke formålet i at kunne forklare sammenhænge i dataet, men fokuserer på at prædiktere den korrekte værdi gennem algoritmer, der er designet til at se mønstre i et givet data.

Prædiktionsmodeller

Fundamental er prædiktionsmodeller relativ simple, i den forstand at formålet er at estimere en værdi y, ved at anvende input X. Hvis det forekommer at middelværdien i variablen X's fejlled er nul, så kan y prædikteres på følgende måde:

$$\hat{y} = \hat{f}(X)$$

Hvor det gælder at f er den systematisk information om y leveret af X, hvilket resulterer i prædiktionen af y. Den eksakt formel for f er som regel ikke relevant, så længe den er i stand til at prædiktere y, normalt omtales funktionen f som værende en "black box"[1].

Præcisionen i prædiktionen af modellen kan opdeles i to parter, en reducerbar og en ikke reducerbar:

$$E(Y - \hat{Y})^2 = \underbrace{[f(X) - \hat{f}(X)]^2}_{\text{Reducerbar}} + \underbrace{\text{Var}(\epsilon)}_{\text{Ureducerbar}}$$

Grunden til at det sidste led er ureducerbar, skyldes at ϵ forklarer det som modellen ikke var i stand til at prædiktere. Dermed for at øge modellens prædiktionsevne, er hovedformålet at minimere den reducerbar del, ved at anvende de mest passende statistiske teknikker til at estimere funktionen f.

Kausalitetsmodeller

Modsat prædiktionsmodellerne, har kausalitetsmodeller til formål at kunne forstå forholdet mellem y og X, dermed hvordan y ændrer sig som funktion af X. Dermed antages det, at f ikke skal behandles som en black box, da det ikke vil være muligt at undersøge effekterne af X på y, uden at have kendskab til funktionen f.

Ved estimering af resultaterne på funktionen f, anvendes oftest parametriske metoder. Disse metoder er en fremgangsmåde, der har til formål at forenkle estimereing af funktionen f, ved at antage en parametrisk form. Med den følgende antagelse estimeres funktionen f gennem de antaget parameter. Ved anvendelsen af parametriske modeller er der to generelle trin der skal følges:

- Lav en antagelse omkring funktionen f. Den mest anvendte antagelse er at f er lineær:

$$f(X) = \beta_0 + \beta_1 x_1 + \beta_2 x_+ ... + \beta_p x_p$$

- Anvend data til at fit modellen og derefter find værdier til parameterne, så følgende gælder:

$$Y \approx \beta_0 + \beta_1 x_1 + \beta_2 x_+ ... + \beta_p x_p$$

Den mest anvendte metode til at tilpasse data til den givne model er *ordinary least squares* (OLS) (Wang, 2019).

Afvejelse mellem nøjagtighed og kausalitet er derfor en vigtig overvejelse, som skal gøres som evt. investor eller økonomisk aktør på det finansielle marked, og således årsagen til efterfølgende problemformulering for dette speciale.

[1] En black box er et system eller et objekt, der kan ses med hensyn til dets input og output uden nogen viden om dets interne funktion.

1.3 Problemformulering

Kan forecasting performance af aktiepriser i udvalgte lineære og ikke-lineære modeller jævnføres i forskellige aktive porteføljestrategier?

Dette speciale vil afgrænse sig selv ved nærmere at undersøge og sammenholde prædikteringer fra ARIMA som værende den lineær model og LSTM som værende den ikke-lineær model.

1.3.1 Specialestruktur

Følgende indledning og introduktion af problemformulering giver anledning til følgende struktur for resten af specialet.

Kapitel 2 præsenterer en hurtig gennemgang af teorien bag traditionelle ARIMA-modeller, som skal bruges til at sammenligne med LSTM, samt de anvendte vurderingskriterier for forecasting performance som dette speciale vil benytte.

Kapitel 3 vil placere Machine Learning i terminologien Artificial Intelligence og præsentere den biologiske motivation til konstruktion af neural networks og diskutere metodologien bag træning og validering i ML. Kapitlet introducerer yderligere Neural Networks i detaljer med Recurrent Neural Networks, der slutter i LSTM.

Kapitel 4 vil være et kort litteraturreview med lignende artikler omkring forecasting af aktier ved brug af ARIMA og LSTM. Derved kan specialet også kortlægge og benytte sig af tidligere erfaringer og dermed anvende den viden til modellering.

Kapitel 5 vil præsentere en gennemgang af de fem udvalgte finansielle aktiver med tilsvarende deskriptiv statistik. Endvidere vil forecasting og udvælgelse af hhv. ARIMA- og LSTM-modellerne blive introduceret og fastsat.

Kapitel 6 har til formål at afveje de to typer forecastingsværktøjer mod hinanden og dermed konkludere ud fra resultaterne i forecasting af de fem finansielle aktiver, hvilken model der forudsagde mest præcist.

kapitel 7 vil samle prædikteringerne og anvende dem i en Mean-Variance Analyse, og dermed afgøre forecastingskvaliteten på hhv. ARIMA og LSTM ift. porteføljesammensætning. Ud fra resultaterne vil det vurderes, om modellerne kan anvendes i en porteføljesammensætning til at skabe profit.

Kapitel 8 vil diskutere en række muligheder for ydereligere forbedringer af forecast hos ARIMA og LSTM. Dermed vil der åbnes op for mulige udvidelser til modellerne for at forøge deres performance til forecasting af finansielle aktiver.

Kapitel 9 vil binde hele specialet sammen med konklusioner ud fra den udarbejdede analyse og dermed besvare den angivne problemformulering.

1.4 Grafer og plots

Alle beregninger og plots i specialet er udarbejdet enten igennem R 3.6.1 (R Core Team, 2020) eller Spyder (Python 3.7.7). Tablerne er opstillet igennem LaTeX tablegenerator, til grafisk illustrationer er ggplot2 3.2.1 anvendt. Data på de fem finansielle aktiver der anvendes i specialet er hentet fra Yahoo Finance fra perioden 2007-01-01 til 2019-12-31, som daglig data. Hvis koderne til projekts tabeller og figurer ønskes, skal forfatterne af specialet kontaktes.

Lineære modeller 2

Dette afsnit tager afsæt i lineære modeller, og vil beskrive en række teoretiske elementer, som tests, modeller og fremgangsmåde i tidsserie økonometri.

2.1 Lineær modeller

Specialet benytter sig af tidsserieregression til forecasting. Tidsserie analyse anvendes til at udtrække meningsfuld statistiske sammenhænge i data, hvor tid er en underliggende faktor. Formålet er at kunne forudsige fremtidige værdier, baseret på tidligere observeret værdier. Tidsserie tager højde for det faktum at data over tid, har en intern struktur som bl.a. autokorrelation, trends eller er præget af sæson. For at kunne prædiktere fremtidige værdier, kræves det at den underlæggende proces af dataet besidder tidsinvariante egenskaber, som betegnes stationaritet.

Generelt set opdeles tidsserie data i to kategorier, hvor det er kritisk at kende forskellen på de to typer:
Stokastisk data har indbygget tilfældighed, hvor det samme sæt af parameter værdier og startsbetingelser vil lede til et forskelligt output.
Deterministisk data, modsat stokastiske, er outputtet fuldt determineret af parameterværdier og startsbetingelserne.

Det er med fordel at have data der er deterministisk, da det ikke styres af tilfældigheder. Det forekommer oftest i stokastiske tidsserier at tilfældigheder skyldes en *white noise process*. Dette er en samling af ukorrelerede variabler ϵ_t, der har en middelværdi lig nul og en begrænset varians σ_ϵ^2. Det antages at variablerne er uafhængige og identisk distribueret (i.i.d). Matematisk udtrykkes det som følgende:

$$E[\epsilon_t] = 0 \qquad\qquad var[\epsilon_t] = \sigma_\epsilon^2 < \infty$$

I en *white noise process* hvor variablerne er normalfordelt, så gælder det at:

$$\epsilon_t \sim N(0, \sigma_\epsilon^2)$$

Oftest forekommer en form for afhængighed mellem to data punkter i en tidsserie. Derfor er det nødvendigt at anvende autokorrelation, der er et værktøj til at vurdere afhængigheden af to med forskellige tidspunkter i tidsserien. Autokorrelationen anvendes til at bestemme, hvordan den korrekte model modelleres i den givne tidsserie. Før autokorrelationen defineres er det vigtigt at undersøge kovariansen mellem de to punkter i tidsserien. Derfor anvendes *autokovariance*, der er defineret som kovariansen mellem to observationer af en serie. Matematisk kan autokovariansen og autokorrelationen opstilles på følgende måde:

$$cov(\hat{y}_t, \hat{y}_{t-1}) = 1/T \sum_{t=j+1}^{T} (y_t - \bar{y}_{j+1,T})(y_{t-j} - \bar{y}_{1,T-j}) \tag{2.1}$$

$$\hat{\rho} = \frac{cov(\hat{y}_t, \hat{y}_{t-1})}{var(\hat{y}_t)} \tag{2.2}$$

Autokorrelationen måler den lineær afhængighed mellem de to punkter i tidsserien og anvendes i forbindelse med tidsseriens stationaritet.

2.2 Stationaritet

Stationaritet defineres som følgende i specialet:

Stationaritet: *En tidsserie y_t er stationær hvis $E[y_t^2] < \infty$ og for enhver heltal k at hverken $E[y_t^2]$ eller $Cov(y_t, y_{t+k})$ afhænger af tiden, t.*

En stationær proces har dermed egenskaben at gennemsnit, varians og autokorrelation ikke afhænger af tiden. Dermed skal dataet forekomme uden trends, med konstant varians og autokorrelation, samt ingen periodisk fluktuering (sæson). Antagelserne af stationaritet er således:

- Den forventede værdi af y_t, er konstant, og er ikke en funktion af tiden, t:
  $E[y_t] = \mu$
- Variansen for y_t skal være en konstant, og er ikke en funktion af tiden, t:
  $var(y_t) = \sigma^2$
- Det er gældende at i hvert tidspunkt i tiden t, afhænger kovariansen mellem y_t og y_{t+k}, svarende til afstanden k. $Cov(y_t, y_{t+k}) = f(k)$. Derfor bliver kovariansen mellem y_t og y_{t+k} ikke ændret over tid (Stock and Watson, 2007).

Det forekommer ofte at tidsserier er ikke-stationære, og derfor er det nødvendigt at transformere tidsserierne til stationære for anvendelse til forecasting. Her er et brugbart redskab differentieringen, der kan stabilisere middelværdien i tidsserien, ved fjernelse af ændringerne i niveauet for tidsserien og dermed reducere trends og sæson.

$$Y_i = Z_i - Z_{i-1} \tag{2.3}$$

Dermed vil det differentieret data indholde én enkelt observation mindre end det originale data. I de fleste tilfælde vil en enkelt differentiering skabe stationaritet af en tidsserie. I særtilfælde kan det være nødvendigt at differentiere to gange, det kan dog forekomme at over-differentieret modeller kan medføre forkerte estimater.

Differentiering er et godt redskab til at håndtere ikke-stationære modeller der besidder en *unit root*. Det er ikke muligt for ikke-stationære tidsserier med trend eller drift, at transformeres om til en stabil ARMA-model gennem differentiering. Til dette kan der justeres for trend gennem regression modeller, ved at *detrende* estimeres modellen med lineær trend og subtrahere de fittede værdier fra de observerede værdier (Enders, 2014).

2.3 Autoregressive Moving Average - ARMA

Autoregressive er baseret på ideen om den nuværende værdi y_t, kan forklares som en funktion af de forhenværende værdier p. Hvor p angiver antallet af observationer tilbage i fortiden, der er nødvendig for at forudsige den aktuelle værdi. Af følgende grund betegnes den som AR(p).

AR(p): En autoregressive model af orden p, er defineret som:

$$y_t = \phi_1 x_{t-1} + \phi_2 x_{t-2} + ... + \phi_p x_{t-1-p} + \epsilon_t \tag{2.4}$$

Hvor y_t er stationær og at ϕ er en parameter og ($\phi_p \neq 0$) (Shumway and Stoffer, 2011).

Alternativt til AR(p), hvor det antages at y_t er lineær. Så antager Moving Average af orden q, MA(q), at *white noise* (ϵ_t) på højreside af ligningen er kombineret lineært, til at forme den observerede data.

MA(q): En moving average model af orden q, er defineret som:

$$y_t = \epsilon_t + \theta_1\epsilon_{t-1} + \theta_2\epsilon_{t-2} + ... + \theta_q\epsilon_{t-q} \tag{2.5}$$

Hvor der er q lags og θ er parameter, ($\theta_q \neq 0$) (Shumway and Stoffer, 2011).

En af de mest anvendte tidsserier modeller er Autoregressive Moving Average (ARMA), der er en fusion af AR(p) og MA(q).

ARMA(p,q): En given tidsserie defineret som ($y_t : t = 0, \pm1, \pm2, ...$) er en ARMA($p$,$q$) hvis tidsserien er stationær og følgende er gældende:

$$y_t = \phi_1 y_{t-1} + ... + \phi_p x_{t-p} + \epsilon_t + \theta_1\epsilon_{t-1} + ... + \theta_q\epsilon_{t-q} \tag{2.6}$$

Her gælder det at $\phi_p \neq 0, \theta_q \neq 0$ og $\sigma_\epsilon^2 > 0$. Parameterne p og q er kaldet autoregressive og moving average orden. Således vil en ARMA med $p = 0$ reduceres til MA(q), og vice versa hvis $q = 0$ (Shumway and Stoffer, 2011).

2.4 Autoregressive Integrated Moving Average

Hvis y_t var givet som en *random walk*: $y_t = y_{t-1} + \epsilon_t$, vil der ved differentiering findes at: $\Delta y_t = \epsilon_t$ er stationær. Den *integrated* ARMA model, der betegnes Autoregressive Integrated Moving Average (ARIMA), er en udvidelse af en ARMA model, hvor differentiering indholdes.

En givet tidsserie y_t er *integrated* af orden d, I(d) hvis:

$$\Delta^d x_t = \epsilon_t \tag{2.7}$$

Dette er gældende hvis vi differentiere d gange og modtager en *discrete white noise* tidsserie.

ARIMA(p,d,q): En proces y_t er en ARIMA model med orden (p,d,q) hvis:

$$\Delta^d x_t = (1 - B)^d x_t \tag{2.8}$$

Hvor p, d og q er ikke negative heltal og B er defineret som en lag operator. Generelt defineres en ARIMA(p,d,q) model som følgende:

$$\phi(B)(1 - B)^d x_t = \theta(B)\epsilon_t \tag{2.9}$$

Her kan det bemærkes at en ARMA(p,q) proces er tilsvarende en ARIMA(p,0,q) proces. Hvis det forekommer $d \geq 1$, er det gældende at tidsserien y_t ikke er stationær, og for at opnå en tidsserie der er stationær, skal serien differentieres d gange. Det siges dermed at tidsserien er *integrated* af orden d og matematisk formuleres det som $(x_t) \sim I(d)$.

Hvis det forekommer at en tidsserie ikke opfylder betingelserne for at være en ARMA(p,q) model, kan det ikke besluttes at serien er stationær. Dette kan skyldes forskellige årsager:

- Det tilsvarende AR element har en rod på enhedscirklen, og derved en *unit root*, og af den grund skal tidsserien differentieres.
- Processen indeholder en tidstrend og er nødsaget til at blive *detrended*.

Unit root test kan anvendes til at afgøre om hvorvidt tidsserien er ikke-stationær grundet en tidstrend eller en rod på eller udenfor enhedscirklen(Shumway and Stoffer, 2011).

2.4.1 Unit Root

Unit root er en ikke-stationær AR eller ARMA tidsserier. Den simpleste af dem er er en *random walk*: $y_t = y_{t-1} + \epsilon_t$. Formålet med en unit root test, er at teste hvorvidt AR(p) polynomium har en rod på eller udenfor enhedscirklen.

Unit root er en stokastisk trend i tidsserieøkonometri, der også refereres til som en *'random walk with drift'*. En tidsserie med en *unit root*, har et mønster, der er uforudsigelig og er ikke-stationær. Antagelserne om en klassisk regression bygger på (y_t) er stationær og fejlleddet har en middelværdi lig nul, samt begrænset varians. I 1974 påviste Granger og Newbold gennem Monte Carlo metoden, at en spuriøs regression bestående af to uafhængige *'drifts'* frie for *random walks* vil producere stærke beviser for et lineært forhold. En spuriøs regression har en høj R^2- og t-statistik, der ser ud til at være signifikant, men er uden økonomisk mening. Dermed fremstår output til at se fornuftig ud, men undersøges de mindste kvadreret estimater (*least squares estimates*), fremkommer det at estimaterne ikke er konsistente og dermed er de statistiske inferens ikke holdbare (Enders, 2014).

2.4.2 Augmented Dickey-Fuller Test

Augmented Dickey-Fuller test (ADF) er en metode til at tjekke om data er stationært igennem unit root. ADF adskiller sig fra den klassiske Dickey-Fuller test, i det den tillader højere ordens autoregressive processer ved at indkludere Δy_{t-p}. Testen er som følgende:

$$\Delta y_t = \gamma y_{t-1} + \sum_{i=2}^{n} \beta_i \Delta y_{t-i+1} + \epsilon_t \tag{2.10}$$

Δ beskriver ændringen i tidsserien på y_t på tidspunktet, t. γ angiver om der er en unit root tilstede. Nulhypotese opstilles således at hvis γ er lig med nul, ville dette indikere tilstedværelse af en unit root, og vice versa.

$$H_0 = \gamma = 0$$
$$H_1 = \gamma < 0$$

Hvis det forekommer at nulhypotesen forkastes, er det muligt at opstille en ARMA(p,q) model (Enders, 2014).

Hvis stationaritet opnås, skal ARIMA-modellen ikke lide af seriel korrelation før forecasting kan begynde. Seriel korrelation vil blive gennemgået og redegjort for i analysen angående modeludvælgelse for ARIMA i kapitel 5.2. Næste afsnit vil præsentere ud fra hvilke forecasting performance parametre, dette speciale vil vurdere LSTM og ARIMA ud fra.

2.5 Evalueringsmetoder

Der findes en lang række forskellige værktøjer og metoder til at måle performance af forecasting. Overordnet set er der ingen konsensus om, hvilket redskab der bedst måler performance af forecastet. Specialet vil tage udgangspunkt i to tabsfunktioner *root mean squared error* (RMSE) og *mean absolute error* (MAE). En tabsfunktion beskriver hvor god modellen er til at forudsige den forventet pris på et givet aktiv. Valget bag RMSE og MAE ligger i, at ikke alle loss-funktioner passer til alt slags data. Loss-funktionerne opdeles i to kategorier: klassifikation og regression. Ved forecasting er valget faldet på regression, da det ikke giver mening at snakke om et klassificeringsproblem. Ved et klassificeringsproblem opstilles en binær problemstilling, om aktierne stiger eller ej, hvilket ikke er relevant. Det sker da formålet med specialet er forecasting af de fremtidige værdier for aktier med vis nøjagtighed ift. at kunne porteføljesammensætte.

2.5.1 Root Mean Squared Error

RMSE er en ofte anvendt metode til at måle en models fejl i prædiktering af kvantitativt data. RMSE udtrykkes matematisk på følgende måde:

$$RMSE = \sqrt{\sum_{i=1}^{n} \frac{(y_t - \hat{y}_t)^2}{n}} \tag{2.11}$$

RMSE beskriver standardafvigelsen i residualerne, der er et mål for, hvor langt regressionslinjen er fra observationerne. Styrken i metoden ligger i, at den håndterer outliers i dataet bedre, grundet den anvender kvadrerede værdier.

2.5.2 Mean Absolute Error

MAE måler den absolutte forskel mellem den observerede værdi y_t og den estimerede værdi $\hat{y}_t$. Matematisk udtryk defineres MAE som følgende:

$$MAE = \sum_{i=1}^{n} \frac{|y_t - \hat{y}_t|}{n} \tag{2.12}$$

MAE er ikke ligeså hårdfør over for outliers, da den ikke kvadrerer fejlleddet, og dermed ikke kun består af positive afvigelser. Dermed vil værdien af fejlledet forøges, når $\epsilon > 1$, og dette medfører at outliers i datasættet vil få en højere vægtning i RMSE end MAE (Grover, 2018).

2.5.3 Pesaran Timmermann test

Pesaran Timmermann testen er bygget til at vurdere hvorvidt den anvendte model til forecasting reelt kan forudsige retningen af ændringen, bestemt af et givet signifikansniveau, α. I Pesaran og Timmermanns videnskabelig artikel "*A simple nonparametric test of preditive performance*", anvendes en ikke-parametrisk test for at måle prædiktionen. For en given observeret tidsserie, y_t og et forecast af tidsserien, x_t, defineres testen som følgende:

$$S_n = \frac{\hat{P} - \hat{P}_*}{[v\hat{a}r(\hat{P}) - v\hat{a}r(\hat{P}_*)]^{\frac{1}{2}}} \tag{2.13}$$

hvor følgende er defineret som:

$$\hat{P} = n^{-}1 \sum_{t=1}^{n} I(y_t, x_t)$$

$$\hat{P}_* = \hat{P}_y \hat{P}_x + (1 - \hat{P}_y)(1 - \hat{P}_x)$$

$$v\hat{a}r(\hat{P}) = n^{-1} \hat{P}_*(1 - \hat{P}_*)$$

$$\hat{P}_y = n^{-1} \sum_{t=1}^{n} I(y_t)$$

$$\hat{P}_x = n^{-1} \sum_{t=1}^{n} I(x_t)$$

$$I(\cdot) = \begin{array}{ll} 1 = & if(\cdot) > 0 \\ 0 = & otherwise \end{array}$$

I de ovenstående matematiske ligninger repræsenterer $\hat{P}_y$ og $\hat{P}_x$ den givne sandsynlighed for at henholdsvis y_t og x_t er større end nul. Dermed forstås det som andelen af gange hvor retningen af y_t kan forudsiges korrekt. Endvidere er $\hat{P}_*$ defineret som sandsynligheden for at y_t og x_t begge er større end nul adderet med sandsynligheden for at de ikke er.

S_n er givet til at følge en normalfordeling under nulhypotesen om at x_t ikke er i stand til at kunne forudsige retningen i ændringer i y_t. n er defineret som antallet af den samlede mængde af forecast observationer. Grundet Pesaran Timmermann testen er en en-sidet test og hvis nulhypotsen forkastes, kan det siges at forecastet forudsiger retningen i en given tidsserie (Pesaran and Timmermann, 1992).

2.5.4 Anatolyev & Gerko test

Anatolyev og Gerko test blev af dem selv beskrevet som en "Market Timing Test" for den betinget middelværdi uafhængighed af finansiel afkast. Anatolyev og Gerko testen er konstrueret på baggrund af en handelsstrategi, der udsender et købsignal, hvis forecastet for næste periodes afkast er positiv og et salgssignal ellers. Det gennemsnitlige afkast af denne handelsstrategi sammenlignes med en benchmarkstrategi, der udsteder tilfældigt køb- og salgsignal. Således tager nulhypotesen udgangspunkt i, at den betinget middelværdi i forecastet er uafhængig (Anatolyev and Gerko, 2005).

2.5.5 Diebold-Mariano test

Det er oftest interessant at måle, om hvilken model der performer bedre end den anden når det kommer til forecasting. Det åbenlyse vil være at vælge modellen der besidder de mindste fejlmålinger, men specialet

vil samtidig undersøge forecastingsfejlene i modellerne er signifikant forskellige fra hinanden. Til at se hvilken model der performer bedst, anvendes Diebold-Mariano testen (DM) (Enders, 2014). Diebold-Mariano er i stand til at teste hvorvidt modellerne har en signifikant forskellig evne til at forecaste. Diebold-Mariano bruger e_i^2 som tabsfunktion der er givet som følgende:

$$\bar{d} = \frac{1}{H}\sum_{i=1}^{H}[g(e_{1i}^2(j)) - g(e_{2i}^2(j))] \tag{2.14}$$

Her er e_{1i} og e_{2i} defineret som forecastingsfejlen for henholdsvis model 1 og 2. Værdien af $\bar{d}$ vil være lig nul under nulhypotesen om der forekommer forecastnøjagtighed. Grundet at det er middelværdien af tabet, så vil det forekomme af $\bar{d}$ vil gå mod en normalfordeling. Den anden hypotese er at nøjagtigheden i de to forecasts niveau er forskellige fra hinanden. γ_i beskrives af Diablo og Mariano, som værende den i'te autokovarians af d_i sekvensen. Ved j-step-ahead vil DM se følgende ud:

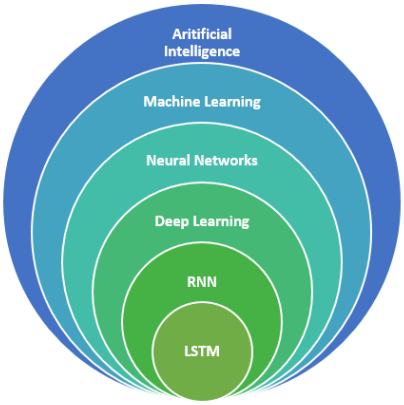
$$DM = \frac{\bar{d}}{\sqrt{\frac{\gamma_0+2\gamma_1+\ldots+2\gamma_q}{H+1-2j+H^{-1}j(j-1)}}} \tag{2.15}$$

For at teste om d_i er forskellige fra nul, kan d_i regresseres på en konstant ved at anvende en t-test, med *robuste standard errors*. Ligning 6.2 er følsom overfor variablen q og det gælder at hvis en eller mere af $\gamma_i < 0$, så kan de estimerede varianser forrkomme negative (Enders, 2014).

Ikke-lineære modeller 3

For at kunne forstå og bruge Long Short-Term Memory (LSTM) korrekt og derved undersøge dens performance mod den klassiske økonometrisk ARIMA-model, er det nødvendigt at træde et par skridt tilbage og få dannet et overblik. Dette overblik er forsøgt afbilledet i figur 3.1.

Figure 3.1: Kunstig intelligens overblik



Derfor vil dette metodeafsnit starte med at forklare om AI, for til slut at kunne rationalisere LSTM som er en gren af Deep Learning (DL) og en særlig type af Neural Networks (NN), som altsammen er en del af ML, der udspringer af AI.

3.1 Artificial Intelligence

Tankegangen om AI har eksisteret i flere årtier, siden det blev introduceret i rå form af neurofysiologen Warren McCulloch og logikeren Walter Pitts, der byggede en simpel kunstig model, hvor både input og output blev betragtet binært. Formålet med modellen var at summere alle inputs til en værdi, baseret på hvorvidt inputtet enten var stimulerende eller hæmmende (Russell and Norvig, 2003). Fødslen af AI kan krediteres den amerikanske matematiker John McCarthy, der til trods for McCulloch og Pitts anses at betragtes som pioneren inden for AI (Childs, 2011). Udviklingen har været eksponentiel, da AI's natur først nu har kunne realiseres og studeres grundet computerens udvikling. Selve udtrykket blev brugt for første gang i 1955, hvor John McCarthy ansøgte om midler til Dartmouth-konferencen, der gjorde AI til et selvstændigt forskningsområde (Det Etiske Råd, 2007).

AI er den underkategori inden for datalogien, der beskæftiger sig med at få maskiner til at være selvtænkende. Maskinerne skal kunne gå målrettet efter f.eks. genkendelse af mønster og

drage konklusioner og agere ud fra disse, og ligeledes kunne tilpasse sig til nye konklusioner og handlingsmuligheder. AI er dermed sagt på en anden måde:

"*A self-improving machine (or program) is created. When computers are able to learn by reading existing natural language text and when an AI has problem solving capacity equivalent to the computer science community.*" (Hall, 2011)

Mulighederne med AI er således mange, da en fungerende AI både kan behandle sprog i tekst og tale, udvise ræsonnement, problemløse og planlægge ud fra given data. Et velfungerende AI system er således præget af en form for logik og rationalitet, der kan sammenlignes med den menneskelig kognitive tænkning, hvorved systemer og/eller regler opstilles og derfra behandles data. Denne formelle logik, som prægede AI frem til 1980'erne, blev med McCarthys metodologiske position ikke ændret væsentligt, siden den først blev formuleret i (McCarthy, 1959) og uddybet (McCarthy and Hayes, 1969).
Motivationen til at bruge formel logik er den, at en logisk formalisering hjælper med at forstå selve ræsonnementsproblemet. Dermed bliver påstanden den, at uden en forståelse af hvad ræsonnementetsproblemet er, vil det ikke være muligt at implementere en løsning. Plausibelt som dette argument kan synes, er det kontroversielt og strider imod AIs kunstige indlæring; en alternativ metode ville søge at lære eller udvikle den ønskede forståelse af ræsonnementsproblemet.

3.2 Machine learning

Dette alternativ kom i starten af 1980'erne, hvor ML opnåede anerkendelse. Det gik op for mange inden for AI-feltet, nødvendigheden af *non-monotonic* logik[1]. I denne sammenhæng kan nævnes: (McDermott and Doyle, 1980), (Reiter, 1980) og selv (McCarthy, 1980) indså problemet.
Problemet ved *monotonicity* blev første gang nævnt af (Minsky, 1974), og derved blev Minksy ophavsmanden til dette udtryk. Problemet skulle findes i, at den pålagte logik og regelsættet ikke ændrede sig over tid og ved tilføjelse af ny data. Dermed kan allerede valide deduktioner ikke ændres, hvilket gælder for naturlove som f.eks. tyngdekraften. Dette mente Minsky var en enorm udfordring, da stort set alt andet end naturlovene konstant ændres og dermed ændres konklusionerne ved tilføjelsen af mere data. Af dette udsprang ML, der kunne lære nye regelsæt og drage nye konklusioner. ML er defineret på flere måder:

- "*Machine Learning at its most basic is the practice of using algorithms to parse data, learn from it, and then make a determination or prediction about something in the world.*" (Copeland, 2016)'
- "*Machine Learning focuses on the question of how to get computers to program themselves (from experience plus some initial structure).*" (Mitchell, 2006)
- "*Machine learning is based on algorithms that can learn from data without relying on rules-based programming.*" (Pyle and José, 2015)

Fælles for dem alle er, at algoritmerne skal kunne lære over tid, ved bl.a. at optimere en række parametre

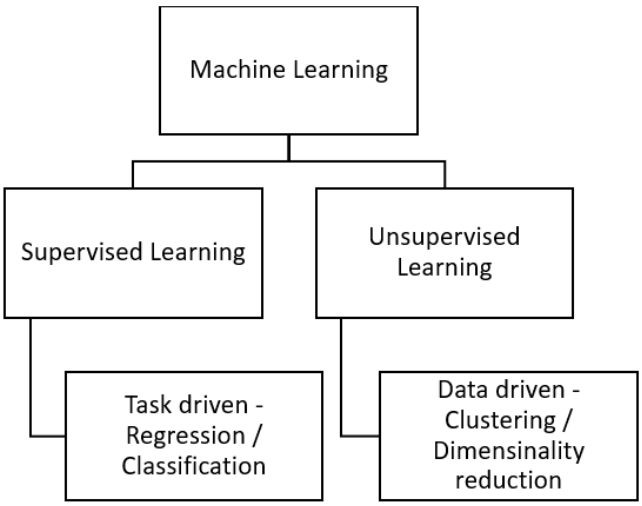
[1]Non-monotonic logik er med andre ord udtænkt til at fange og repræsentere foreløbige konklusioner, hvilket gør det muligt for ræsonnører at trække deres konklusion(-er) tilbage på baggrund af nye beviser.

til at udføre en given opgave. Dermed er ML den del af AI, der beskæftiger sig med automatisering og effektivisering af statistiske modelleringsmetoder, der benyttes til analyser af store mængder af kompleks data.

Dermed er AI overgruppen omhandlende skabelse af intelligente maskiner, der kan simulere menneskelig tænkningskapacitet og adfærd, mens ML er applikationen eller undergruppen, der giver maskiner muligheden for at lære af data uden at blive programmeret eksplicit. Nogle typiske benyttelser af ML, er f.eks. klassificeringer, *clustering* osv. Disse bliver opnået gennem historisk data, som "fodres" til algoritmen. ML er således anvendeligt på økonomiske og finansielle problemstillinger for dette speciale, da meget af nutidens data er kendetegnet ved en ikke-lineær natur, men til tider værende lineær af natur og en blanding mellem af disse to, f.eks. Fourier decomposition[2]. Således kan sammenhænge mellem to variabler ændre sig over tid, og derfor skal logik og regelsæt i ML også kunne ændre sig. Vejen til den 'bedste' ML-model afhænger af dataet, og derfor skal det klargøres, for at kunne beslutte hvilken model der skal anvendes for at fit det specifikke data med. Manglen på en "super" model, der kan løse alle opgaver ud fra alle former for data, dvs. enhver model, der hævder at tilbyde overlegen fleksibilitet ved analyse af datamønstre, gør det normalt til prisen af indførsel af ekstra antagelser eller ved at ofre vigtige generaliseringer. Denne afvejning mellem nøjagtighed og antagelser kaldes i statistik også for: *There ain't no such thing as a free lunch.*

ML deles op i to primære grupper, som illustreret i figur 3.2, afhængig af dataets natur, som derved giver anledning til forskellige træningsalgoritmer. For dette speciale vil supervised learning (SL) blive brugt (Ayodele, 2010). Ved SL forstås, at algoritmen præsenteres for dele af input og output fra et datasæt. Målet er, at modellen skal lære en eller flere underliggende funktioner (afhængig af antal variabler) ved at kortlægge input til output, og dermed være i stand til at spotte sammenhængene i dataet. Dataet opdeles således i to; træningssæt og testsæt. Et træningssæt giver algoritmen mulighed for at lære at justere parametrene igennem mindre grupper som kaldes *batches*, således at prædiktions fejlene minimeres. Et testsæt anvendes til validering af prædiktionen.

Figure 3.2: Typer af Machine Learning (Ayodele, 2010)



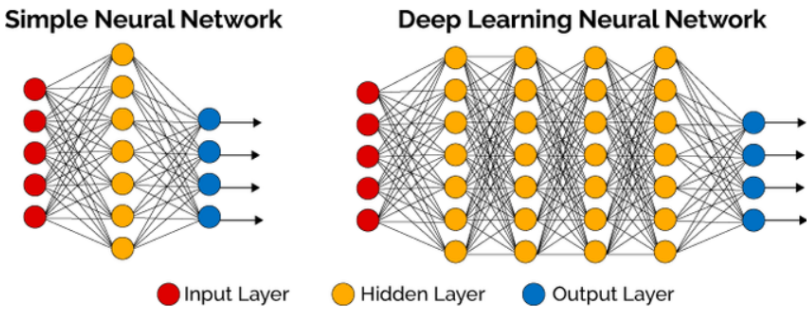
[2]Fourier decomposition er undersøgelsen af den måde, generelle funktioner kan repræsenteres eller tilnærmes med summer af enkle trigonometriske funktioner

En optimal brug af SL kræver typisk en form for data-bearbejdning, ofte med et valg omkring dataindsamling. Her findes forskellige metoder såsom “brute-force”, hvilket betyder måling af alt tilgængeligt data i håb om, at de rigtige (informative og relevante) funktioner kan isoleres, f.eks. afprøvning af samtlige mulige input til en funktion, indtil det ønskede resultat optræder (Ayodele, 2010). Et datasæt, der er indsamlet med "brute-force"-metoden, er imidlertid ikke direkte egnet til induktion tankegang. Det indeholder i de fleste tilfælde støj og manglende funktionsværdier (Zhang, 2002). En anden metode til dataindsamling er brugen af relevant teori og argumentation, til udvælgelses af data, hvilket vil være specialets metode. Valget af data vil blive præsenteret og redegjort for, i kapitel 5.
Der findes forskellige metoder og fremgangsmåder, til forarbejdning af data. Dette afhænger alt sammen af dataets natur, da eksempelvis manglende observationer og/eller outliers kan forekomme. Alle disse til- og fravalg af metoder vil ikke blive fremført i specialet, da det må antages for at være uden for emneområdet, dog vil der blive redegjort og begrundes for de specifikke udvalgte metode i analysen. Ligeledes er underkategorien Unsupervised Learning i figur 3.2 ikke en del af specialet og vil således ej heller blive redegjort for, men blot nævnes i sammenhæng med ML. Dermed vil fremtidige referencer i specialet omkring ML være udmøntet på SL.
Udfordringen i ML er at finde algoritmer, der er kraftige nok til at lære komplekse funktioner, samt robuste nok til at give troværdige resultater. Af ML-modeller kan eksempelvis nævnes: *Decision Tree model* (DT), *Support Vector Machine* (SVM) og neural networks.
- DT-modeller er produceret af algoritmer, der identificerer forskellige måder at opdele et datasæt i grenlignende segmenter. Disse segmenter danner et inverteret *decision tree*, der starter med en root-node øverst på træet. DT-modeller er effektive, idet overvejelsen af alle mulige resultater fremtvinges. Dette kan dog forekomme meget omfattende, da hver gren giver anledning til nye konklusioner og i sig selv måske kræver ydereligere analyser. Beslutningsreglen om at danne en ny gren eller segment under root-noden er baseret på en metode, der udtrækker forholdet mellem analyseobjektet (*target field*) og et eller flere felter, der fungerer som input til at oprette nye grene eller segmenter. Værdierne fra inputtet bruges til at estimere den sandsynlige værdi i target field. Target field kaldes også et resultat, respons eller det afhængige field.
- SVM er kort fortalt en model, der anvendes når alt andet har slået fejl eller dataet simpelthen er uoverskueligt. SVM er en læringsmaskine til klassificeringsproblemer i to grupper. Maskinen implementerer input-vektorer som er ikke-lineært kortlagt til et højt dimensioneret rum. I dette rum konstrueres en lineær beslutningsflade. Særlige egenskaber ved beslutningsoverfladen sikrer SVM høje generaliseringsevner ved anvendelse af polynomiske input-transformationer.
- Neural networks har sin fordel ved forecasting, og er således et oplagt valg for dette speciale (Ayodele, 2010).

3.3 Neural Networks

Neural Network (NN) er en numerisk tilnærmelsesmetode, der igennem en række algoritmer bestræber sig på at genkende underliggende sammehænge i data gennem en proces, der efterligner måden hvorpå den menneskelige hjerne fungerer. Udtrykket NN er en henvisning til neurobiologi, da nogle af de centrale begreber delvist blev udviklet ved at hente inspiration fra forståelsen af den menneskelige hjerne. Det kan basalt set, sættes op på følgende måde i figur 3.3.

Figure 3.3: Arkitektur af neural netværk (Danish, 2019)



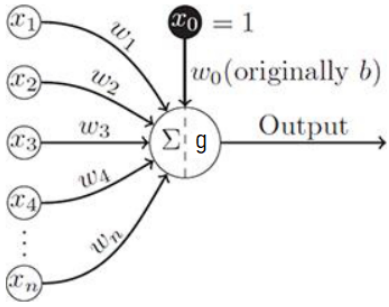
Ligesom i neurovidenskaben så forbindes input med output via en synapse, hvilket kan ske igennem i en eller flere hidden layer, hvilket kaldes for Deep Learning (DL). Disse forskellige forbindelser mellem layer afhænger af styrken af forbindelsen, som matematisk har forskellige vægtninger. Fælles for input-, hidden- og outputlayer er, at de alle består af neuroner. En neuron eller node er organiseret i layers, hvor neuronen har flere vægtet inputforbindelser fra det foregående layer, som derefter indgår i en typisk ikke-lineær aktiveringsfunktion i de hidden layers, hvor inputtet behandles, og til sidst har en output-forbindelse (Brownlee, 2018). Det kaldes også et feedforward neural network (FNN), hvor input bearbejdes gennem layers i en "lige" linje (fra venstre til højre), og ikke i en cirkel som i andre typer af NN.

Layer mellem input og output kaldes hidden primært grundet det ikke er visuelt tilgængelig for eksterne systemer. Derudover forekommer det i litteraturen, at der ikke vides hvad der foregår inde i modellen i de hidden layers, altså *the black box*. Dette er en skarp kontrast til de traditionelle økonometriske modeller, hvor det oftest er kointegration, fejlkorrektion og parameterestimering, som er interessant at afdække og tolke på. Det er ikke tilfældet i NN-modeller, da transformeringerne sker i en såkaldt ȷblackboxȷ eller hidden layers (Molnar, 2019). Dette sætter således en begrænsning for NN, da strukturel inferens og responsanalyse ikke implementeres (Mullainathan and Spiess, 2017). Formålet med NN er at transformere input til output, hvilket sker gennem hidden layers, som afdækker komplekse og (typisk) ikke-lineære sammenhænge i opnåelse af den bedst mulige forecastingsevne. Dermed kan en enkelt forecasting involvere millioner af matematiske operationer (afhængigt af udformning af NN), og således kan en investor ikke følge den nøjagtige kortlægning fra input til output. Overvejelser af millioner af forskellige vægt og kombinationer mellem hvert layer, som interagerer på en kompleks måde, skal kunne forstås for at udregne forecasting i NN (Molnar, 2019).

De fleste NN benytter sig af *back propagation* for at justere netværkets indlæring. NN er givet i en række inputs (x), som afbilledet i figur 3.4. Dette er forbundet med et layer, hvor hver forbindelse har en vægt (w), som summeres i det hidden layer (z) eller ligning 3.1 plus et bias-led. Dette bias-led er mere analogt med aflytningsparameteren i ligningen af en linje, der bevæger linjen op og ned på y-aksen. Formålet

med dette bias-led er at flytte beslutningsgrænsen væk fra det oprindelige (Kelleher, 2019).

Figure 3.4: Neural netværk med bias-led (Kelleher, 2019)

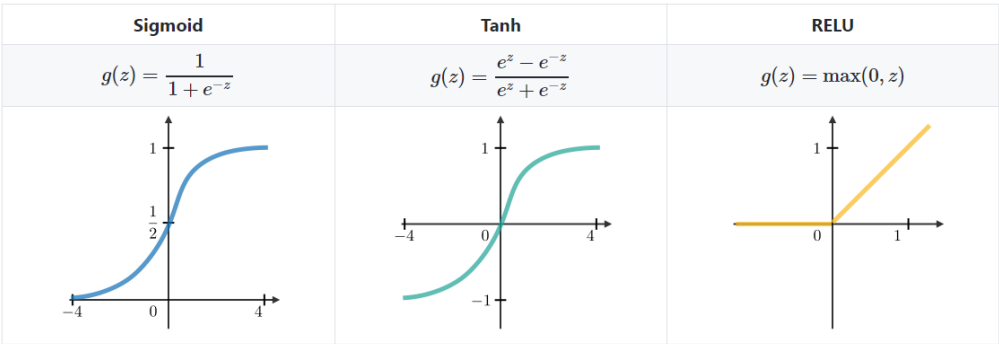


$$(\sum_{i=1}^{n} x_i * w_i) + b = z \tag{3.1}$$

$$g(z) = output \tag{3.2}$$

Den vægtet summering (z) sendes videre igennem en aktiveringsfunktion, g (i ligning 3.2 eller phi i figur 3.4), som derved giver enten output eller input til næste layer. Der findes som tidligere nævnt forskellige aktiveringsfunktioner, hvor nogle af disse er afbilledet i 3.5 (Kelleher, 2019). Disse er sigmoid (Logistic), *the hyperbolic tangent* (TanH) og ReLU[3] aktiveringsfunktion. Derudover kan *step* funktion (binær) også nævnes, som anvender ingen transformation. Den bruges dog sjældent, fordi den reducerer et NN til en regression med polynomtransformationer. Dermed bliver den vægtet summering, udtrykt på den vandrette akse, omdannet til et output, udtrykt på den lodrette akse, afhængig af den valgte aktiveringsfunktion. Denne proces bliver således gentaget afhængig af antal layers i netværket (Mueller and Massaron, 2019).

Figure 3.5: Aktiveringsfunktioner (Amidi and Amidi, 2019)



Aktiveringsfunktionerne fungerer som et filter, der hjælper med at vælge de relevante "signaler" i dataet og undgår de svage og støjende signaler, da funktionen kasserer visse værdier udenfor et given interval (Kelleher, 2019). Dermed handler valget af den korrekt aktiveringsfunktion meget om dataets natur.

- **Sigmoid**: Funktionen arbejder inden for outputintervallet af 0 og 1, og er ikke-lineær. Sigmoid-funktionen lider imidlertid af det, der kaldes *vanishing gradient*[4], hvilket vil sige at funktionen nægter at lære efter et vist punkt.

[3]ReLU har flere varianter og denne sammenhæng kunne nævnes: ELU (Exponential linear unit), PReLU (Parametric Rectified Linear Unit) og LeakyReLU (Mueller and Massaron, 2019).

[4]Gradient: Angiver i hvert punkt for en vektor, hvor meget og i hvilken retning en differentiabel funktion, f, vokser kraftigst.

- **Tanh**: En tanh-funktion er en skaleret sigmoid-funktion. Den har et interval inden for -1 til 1. Den store forskel mellem sigmoid- og tanh-funktionen er, at tanh-funktionsgradienten er stærkere, hvilket betyder at detektering af små forskelle er lettere ved positive tal, hvilket gør en evt. klassificeringen mere følsom. Ligesom sigmoid-funktionen lider tanh af *vanishing gradient*. Problemet med *vanishing gradient* vil blive nærmere omtalt i afsnittene om RNN og LSTM.
- **ReLU**: En ReLU eller *Rectified Linear Units* giver et output i intervallet fra 0 til uendelig. Den kan dermed ved første øjekast ligne en almindelig lineær funktion, men ReLU-funktionen er også ikke-lineært (da den ikke kan blive negativ). En fordel ved ReLU er, at den kræver mindre processingskraft, og når inputtet nærmer sig 0 gives der færre potentielle outputs. Denne fordel kan imidlertid også over tid set som en ulempe, da de neurale netværksvægtninger ikke giver den ønskede effekt længere og dermed stopper med at lære, således at de berørte neuroner dør - hvilket betyder at de ikke reagerer på noget input, hvilket også kaldes et *dying ReLU* (Mueller and Massaron, 2019).

3.3.1 Back Propagation

En af grunde til at NN lære over tid kommer fra udtrykket *back propagation* (BP), som gjorde sit indtog i starten af 70'erne med (Linnainmaa, 1970), og blev tilpasset af (Rumelhart et al., 1986), som for alvor gjorde BP til et populært værktøj indefor ML. BP giver som sagt de rette justeringer, der kræves af optimeringsalgoritmen, og sørger for dens udbredelse i hele netværket. Dette kan ses i tabsfunktionen, som aktiverer nødvendigheden af, at lære visse elementer i dataet bedre (store forudsigelsesfejl giver et højere tab). Det var (Dreyfus, 1973), som først satte tabsfunktionen i forbindelse med BP. BP fungerer dermed ved at beregne gradienten af tabsfunktionen med hensyn til hver vægt. Gradienten beregner et layer ad gangen, og iterere baglæns fra det sidste layer for at undgå overflødige beregninger af mellemliggende termer i kædereglen. Hele det matematiske bevis for BP synes at ligge uden for specialets emnefelt, og derfor stoppes forklaringen af begrebet her, mens tabsfunktion kan skrives op således (Mueller and Massaron, 2019).

$$tab = y * log(h_W(X)) + (1 - y) * log(1 - h_W(X)) \tag{3.3}$$

h_W(x) udtrykker den forudsagt output ud fra W (samlet sæt af vægtninger i matrix) og X (samlet sæt af inputs i matrix), og dermed er tabet differencen mellem det forventet fra testsættet (y) og det forudsagte (fra træningssættet) output. I tilfælde af store forudsigelsesfejl, udsender tabsfunktionen en høj værdi, hvilket optimeringsalgoritmen ønsker at minimere igennem ændring af parametrene. Problemet er kompleksiteten hurtigt vokser, jo dybere netværket (flere hidden layers) bliver. De mange layers gør, at parameterkoefficienterne (forbindelserne) forholder sig til hinanden, da forbindelserne i et layer afhænger af, hvordan forbindelserne i de foregående layer kombinerede inputtet. Løsningen på dette problem er BP. BP er en måde at kortlægge fejlene tilbage i netværket (fra output til input) og få hver forbindelse til at justere dens vægt og bias i overensstemmelse med at minimere tabet. Det første trin i reducering af fejl i netværket afhænger af multiplikationen fra forrige layer. Da værdierne, der bidrager til outputlayer, er lavet ud fra input, der er proportionale med vægtning i deres forbindelse i forrige layer og bias, så kan fejlene omfordeles i henhold til hvert layers bidrag. F.eks. er vektoren af fejl (angivet med δ) i et hidden

layer n i netværket, resultatet af følgende formel (Mueller and Massaron, 2019):

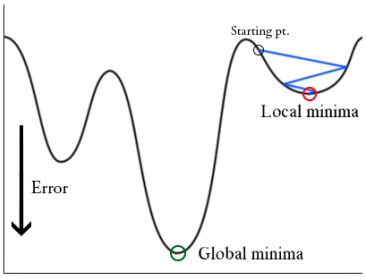
$$\delta_{(n)} = W_n^T * \delta_{(n+1)} \tag{3.4}$$

Denne formel fortæller at, ud fra det endelige delta (δ), hvilket også kaldes gradienten. Herfra kan man gå bagud i netværket med de vægtninger, som blev brugte til at "skubbe" input frem, og partitionere fejlen til de forskellige neuroner. På denne måde kan BP beregne en mere passende vægtning eller bias-led for hver netværksforbindelse for at minimere fejlen. For at opdatere vægten W af layer 1 bruges følgende formel (Mueller and Massaron, 2019):

$$W = W + \eta * \delta * g'(z) * X \tag{3.5}$$

Funktion g' er det første derivat af aktiveringsfunktionen (g) evalueret af inputværdierne (z). Ligning 3.5 refererer også til *the gradient descent method*, der bestemmer hvordan fejlmålingen skal reduceres ved de mulige værdikombinationer. η angiver læringshastigheden, som over tid reducerer marginal effekten af nye vægtninger, der er foreslået af derivatet. Den angivne retning, som determinerer hvorvidt den enkelte vægtning skal reduceres eller øges, kan kun være delvist korrekt. Derfor skal der tages flere små trin i metoden (lav η-værdi), således at algoritmen kan tage en mere præcis retning mod det globale minimum, som tidligere nævnt er det overordnet mål for NN (Mueller and Massaron, 2019).

Figure 3.6: Gradient descent (Saiful, 2020)



Ligesom i figur 3.6 kan problemet opstå, at der findes lokale minimummer for *loss*, og dermed skal den rigtige indstilling af η-værdien findes. En metode kunne være at sætte η-værdien højt som reducerer optimeringsprocessen, men tilgengæld risikeres det at modellen bevæger sig væk fra det globale minimum. En anden metode sætter η-værdien lavt således opleves konstante reduceringer af forudsigelsesfejlen, men samtidig hersker risikoen for at blive fanget i et lokalt minimum. Således er den mest optimale metode, en mellemvej mellem de to førnævnte metoder, hvor η-værdien øges eller formindskes afhængig af forbedringerne opnået ved hjælp af algoritmen. Opdatering af vægtningene kan ske på tre forskellige måder:

1. *Online-tilstand*: Vægtopdateringen sker, når hvert eksempel gennemgår netværket. På denne måde behandler algoritmen læringseksemplerne som en "strøm", hvorfra algoritmen kan lære i realtid. Imidlertid er denne metode følsom over for outliers, så indlæringshastighed skal være lav, og derved konvergerer algoritmen langsom mod en løsning.

2. *Batchtilstand*: Vægtopdateringen sker efter behandling af alle eksemplerne i træningssættet, og dermed overvejer *back propagation* de summerede gradienter af alle eksempler. Denne teknik gør optimeringen hurtigt og mindre genstand for, at varians viser sig i "strømmen".

3. *Mini-batch tilstand*: Vægtopdateringen sker, når netværket har behandlet et undereksempel af tilfældige eksempler i træningssættet. Denne tilgang blander fordelene ved online-tilstand (lav hukommelsesforbrug) og batch-tilstand (en hurtig konvergens), mens der introduceres et tilfældigt element for at undgå, at gradientafstamningen sidder fast i et lokalt minimum (Mueller and Massaron, 2019).

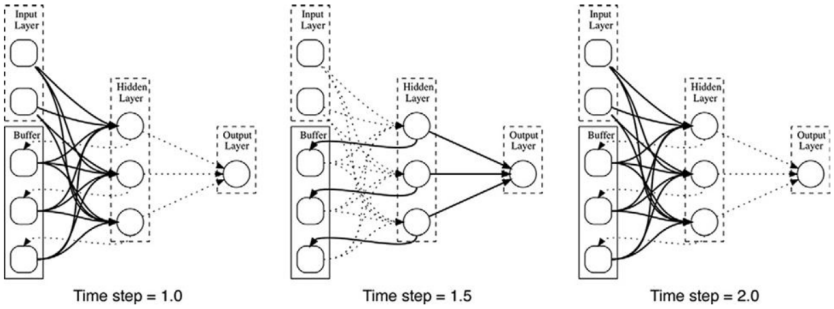
Før tankerne om opdatering af vægtning var fremme, var de første NN, som kan dataeres tilbage til 1800-tallet, blot en simpel variant af datidens lineære regressionsmetoder (Schmidhuber, 2015). Det skal i dette sammenhæng siges, at disse NN ikke kunne lære, da dette først kom i spil med (McCulloch and Pitts, 1943) der lagde fundamentet for AI. Derfra kunne Roger David Joseph i sit ph.d. afhandling (*Contributions to perceptron theory*) i 1961, fremføre den første NN, der kunne lære at vægte forskelligt, med et hidden layer. Efterfølgende kom folk som (Ivakhnenko and Lapa, 1965), (Ivakhnenko et al., 1967) og (Ivakhnenko, 1968) der kunne lave de første NN med flere hidden layers, hvilket refererer til Deep Learning (DL). DL blev dog først et term i (Dechter, 1986), DL er en kompleks udgave af et simpelt NN, der konstant lærer af egne prædiktioner eller træk igennem flere layers. Dermed er DL således en særlig gren indenfor ML med reference til nogle særlige typer af NN (Schmidhuber, 2015). Eksempler på disse typer er med tiden blevet mange, således findes i skrivende stund mere end femogtyve forskellige modeller, som alle har forskellige fordele og ulemper. I specialet er Recurrent Neural Network valgt som det mulige NN til forecasting, da RNN adskiller sig fra de mere traditionelle neurale netværks, hvor alle inputs og outputs er uafhængige af hinanden. Denne egenskab er fordelagtig, når der f.eks. arbejdes med tekst, hvor formålet er at forudsige det næste ord i en sætning (Amidi and Amidi, 2019). Dette kræver, at modellen på forhånd kender de tidligere ord i sætningen, før den kan forudsige det efterfølgende ord. Dette kan samtidig tages og bruges i en økonomisk sammenhæng som i dette speciale, da RNN er et meget nyttigt værktøj i de rigtige applikationer på grund af dets høje automatiseringsniveau i læring og genkendelse af mønstre fra data.

3.4 Recurrent Neural Network

For at recurrent neural netværk (RNN) kan forudsige det næste inputs output, kræves en inddeling af data i sekvenser eller med andre ord en tidsdimension. En RNN er en model, der kan identificere autoregressive egenskaber i data af vilkårlig længde. De første RNN modeller kom frem i 1980'erne, men efter udviklingen af BP fra (Rumelhart et al., 1986) kunne (Elman, 1990) fremføre et simpelt RNN (*Elman network*), der har dannet fundamentet for viderearbejde med RNN. Et vigtigt kendetegn ved NN er modellens manglende hukommelse, der sørger for at hvert input behandles uafhængigt, uden at der holdes status mellem inputene (Kelleher, 2019). Dette er ikke tilfældet for RNN, da de har et *delay* komponent eller *memory buffer*, der gør det muligt for tidligere sekvensers output at påvirke netværket (Hagan et al., 2014). Dermed forbedres RNN's evne til at inkorporere information fra tidligere sekvensers output og/eller andre aktiveringsfunktioner, og bidrager i høj grad til ydeevnen i prædiktionen, da tidligere sekvenser kan bære relevant information om fremtidig udvikling. Dette faktum gør RNN interessant i forhold til en række forskellige problemstillinger, såsom forudsigelse af det næste ord i en tekst eller aktiekurser (Amidi and Amidi, 2019). RNN er samtidig kendetegnet ved kun at have et hidden layer, som både får information fra nyt input og fra tidligere sekvensers output (Kelleher, 2019).

"*In a sense, RNNs are the deepest of all NNs - they are general computers more powerful than FNNs, and can in principle create and process memories of arbitrary sequences of input patterns.*" (Schmidhuber, 2015)

Figure 3.7: RNN med input og memory buffer (Kelleher, 2019))



I figur 3.7 illustreres et simpelt RNN og ligner dermed et typisk NN, men blot med tilføjelse af *memory buffer*. RNN er således en efterkommer af NN, hvor tidligere output anvendes igen i hidden layers, i stedet for en enkelt gang. I tidstrin 1.5 vises, hvordan aktiverings flowet foresætter videre til output layer, men samtidig sendes samme information tilbage til *memory buffer*, hvor det overskriver alle tidligere informationer. *Memory buffer* ændrer eller transformerer ikke informationerne, og udover er der ingen særlige vægtninger fra det hidden layer til *memory buffer*. Der er dog forskellige vægtninger på alle andre forbindelser og dermed også fra *memory buffer* til det hidden layer. I tidstrin 2 modtager netværket det næste input, og dette overføres til det hidden layer sammen med den information, der er gemt i *memory buffer*. Denne gang indeholder bufferen de aktiveringer, der blev genereret i det hidden layer på det første input i tidstrin 1. Dermed husker RNN tidligere output og anvender dette i funktion med næste input (Kelleher, 2019).

Back-propagation Through Time (BPTT) skal bruges for at kunne propagere tilbage i netværket og reducere et givet tab. Samtidig regnes en *loss* fremfor *cost* da dette skal udregnes pga. en sekvens og ikke hele sættet. I et træningsæt startende med t_o og afsluttet med t_p, kan gradienten for outputlayer, j, i tiden, t, udregnes på følgende måde ved brug af z fra ligning 3.1:

$$\delta_j(t) = \frac{\partial loss}{\partial z_j(t)} \tag{3.6}$$

Mens *loss* funktionen med t_o og t_p kan udtrykkes således:

$$loss = \frac{1}{2} \sum_{t=t_o}^{t_p} \sum_{j=\wedge} e_j^2(t) \tag{3.7}$$

hvor $e_j = y(t) - y_j(t)$, som er differencen mellem det forventede og forudsagt output.

$\wedge$ udtrykker et indeks for en neuron i output layer i specifikt tidstrin. Afhængig af placeringen af neuronen (output eller hidden layer), så afhænger den lokale gradient i den tilbagevendende konfiguration også af iterationen. Dermed kan gradienten for t_n skrives op på denne måde:

$$\delta_j(t_o) = g'(z_j(t_o))e_j(t_o) \tag{3.8}$$

Mens gradienten for alle observationer mellem t_o og t_p kan skrives op på følgende måde:

$$\delta_j(t) = g'(z_j(t))[e_j(t) + \sum_{k=\wedge} W_{jk}\delta_k(t+1)] \tag{3.9}$$

Hvis det antages at samme aktiveringsfunktion er valgt for alle hidden layers (k), kan tilpasning af vægtning (W) når fejl propageres tilbage med $t_o - 1$, skrives op på følgende måde:

$$\Delta W_{jk} = \eta \sum_{t=t_o+1}^{t_p} \delta(t) x_m(t-1) \tag{3.10}$$

Vægt og bias er stationært under en sekvens, og dermed reduceres kompleksiteten af modellen, grundet afhængighed mellem inputs. Dette kunne antages som en fordel ved RNN, men dette er faktisk det komplet modsatte. Dette skal forståes igennem *the vanishing gradient problem*[5], da RNN ikke kan ændre sine vægtninger igennem BPTT. Dette skyldes, at træning af et RNN til at behandle en sekvens kræver, at der anvendes back propagation igennem hele netværket og hele længden af sekvensen. Dette involverer gentagende multipliceringer af fejlen med vægtene på forbindelserne, der giver aktiveringen fra et layer til det næste. Dermed opstår problemet, da det er det samme sæt vægte (stationære over tid), der bruges på alle forbindelserne mellem de hidden layers. Som konsekvens af dette så involverer tilbagepropagationen af en fejl gennem k-tidstrin (bl.a.) multiplicering af fejlgradienten med det samme sæt vægte k gange. Dette svarer til at multiplicere hver fejlgradient med en vægt opløftet i k. Hvis denne vægtning er mindre end 1, når den opløftes, så formindskes denne eksponentielt, og følgelig har fejlgradienten også en tendens til at falde med en eksponentiel hastighed med hensyn til længden af sekvensen - og dermed "forsvinder". Samme problem ses i andre netværker end i RNN, da store NN med flere hidden layers oplever at distancen mellem output og input bevirker at fejlen "forsvinder", ved anvendelsen af back propagation i netværket med vægtninger på forbindelser på mindre end 1. Samtidig kan problemet med *the exploding gradient problem*, ligeledes opstå, men det ses sjældent i RNN og derfor ikke noget dette speciale vil komme ind på (Kelleher, 2019).

Denne problemstilling blev derfor en udfordring for RNN, og således syntes en udvikling inden for RNN nødvendig. Derfor vil efterfølgende afsnit være en gennemgang af (Hochreiter and Schmidhuber, 1997) omhandlende deres præsentation og introduktion af LSTM, som netop tog hånd om *the vanishing gradient problem* i RNN. Derudover vil der være tilføjelser fra andre artikler med vigtigt kommentarer og inputs til LSTM, da der siden 1997 er blevet videreudviklet på LSTM og dermed kommet flere varianter af LSTM[6] (Zhang et al., 2018).

[5] Algoritmen lærer for hurtigt, og overser dermed vigtig information i efterfølgende sekvenser.

[6] Mange varianter af LSTM er blevet skabt, men de tre mest udbredt må siges at være:
1. (Gers and Schmidhuber, 2000) med *Extending LSTM with "Peephole Connections"*, som er en videreudvikling af LSTM med flere forbindelser.
2. (Cho et al., 2014) med *Gated Recurrent Unit* eller GRU, som er en forsimplet udgave af LSTM og har derfor vundet meget popularitet.
3. (Yao et al., 2015) med *Depth-Gated Recurrent Neural Networks*, som introducerer en *depth gate*.

3.5 Long Short Term Memory

LSTM er en bestemt udgave af en RNN-model, som indeholder en kæde af operationer, hvor den memorerer og anvender tidligere outputs. LSTM indeholder en mere kompliceret struktur end RNN, hvor LSTM er opbygget med en *memory* som opbevarer information over længere perioder end RNN gør. Denne hukommelse består af fire komponenter: Constant Error Carousel (CEC) og tre gates: Forget-, input- og outputgate. Den oprindelige artikel med (Hochreiter and Schmidhuber, 1997) præsenterede ikke en forgetgate, da denne først bliver tilføjet i (Gers et al., 1999).

CEC-delen har egenskaben til at løbe igennem hele kæden, uden at skulle igennem gates og kan dermed bibeholde dens gradient. CEC'er er forbundet til flere ikke-lineære adaptive enheder (nogle med multiplikative aktiveringsfunktioner), hvilket er nødvendigt for at lære ikke-lineær sammenhænge. Ved en omskrivning af ligning 3.9 for en enkelt neuron i det hidden layer, opnås følgende (Hochreiter and Schmidhuber, 1997):

$$\delta_k(t) = g'_k(z_k(t))W_{kk}\delta_k(t+1) \tag{3.11}$$

For at holde fejlene konstante i stedet for at de eksploderer eller forsvinder, så kræves det følgende:

$$g'_k(z_k(t))W_{kk} = 1 \tag{3.12}$$

Integralet af aktiveringsfunktionen g bliver således,

$$g_k(z_k(t)) = \frac{z_y(t)}{W_{kk}} \tag{3.13}$$

og dermed skal aktiveringen være linear og konstant:

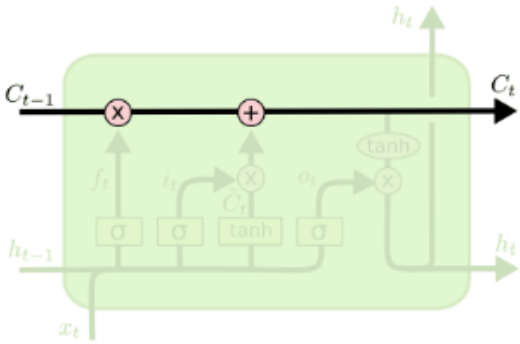
$$y_k(t+1) = g_k(z_k(t+1)) = g_k(W_{kk}y^k(t)) = y^k(t) \tag{3.14}$$

CEC er således placeret i neuronen så det kan aktiveres og hjælpe til at reducere netværkets samlede fejl (Hochreiter and Schmidhuber, 1997). CEC-delen er hovedårsagen til, at LSTM kan lære at opdage vigtigheden af (og huske) begivenheder. Dermed kan begivenheder der skete for tusindvis af diskrete tidstrin siden, huskes og bruges, mens RNN'er allerede mislykkedes i tilfælde af minimale tidsforsinkelser på 10 trin (Schmidhuber, 2015). Dermed kan forudsigelsesfejl progageres tilbage i netværket gennem en CEC, uden at "forsvinde" eller "eksplodere", men forblive som de er. Således grunden til, at LSTM-modeller har haft stor succes med bl.a. at lære langsigtet afhængigheder i forhold til RNN, da informationen nemmere kan flyde igennem CEC uden ændringer (Zhang et al., 2018).

(Hochreiter and Schmidhuber, 1997) udpegede selv problemet med at en neuron med CEC, forbundet med flere neuroner med samme vægt, vil modtage forskellige signaler om enten at opdatere eller ignorere signalerne. Dermed skal neuronen lære enten at huske eller ignorere, da informationen i CEC er fanget og ikke kan ændres. Dette problem kan dermed både påvirke input og output, og tilføjelse af en anden neuron er derfor ikke uden problemer. I en økonomisk sammenhæng giver CEC, netværket mulighed for at huske lagede afhængigheder ved at lagrer dem, ligesom vi kender det fra inkluderingen af en eller flere lags i almindelige tidsserie-økonometri.

Grundessensen i en LSTM er *cell state* (C_t), der er markeret frem i figur 3.8. C_t kan sammenlignes med et transportbånd, der løber igennem hele kæden med nogle mindre lineære interaktioner. Det er derfor nemt for information at passere igennem C_t uden større ændringer i informationen.

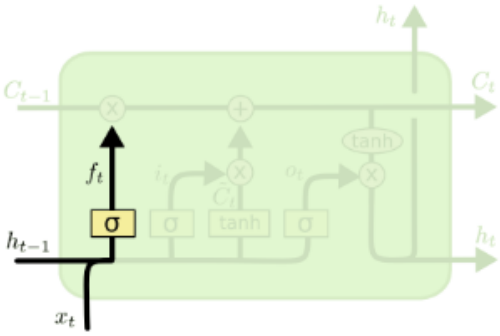
Figure 3.8: Cell State



LSTM besidder evnen til at tilføje eller fjerne information fra C_t gennem såkaldtes *gates*. *Gates* er således en måde for modellen valgfrit at lade information passere igennem. De er sammensat af en sigmoid NN layer, hvor outputtet har en værdi mellem 0 og 1. Værdien 0 indikerer at intet information vil blive lukket igennem den given *gate*, modsat vil en værdi på 1 betyde at alt information kommer igennem. LSTM er således opbygget med en C_t og tre *gates*, der kan lære hvilken information der er vigtig og frasortere hvad der er irrelevant.

Det første step i en LSTM, er at bestemme hvilken information der skal fravælges fra C_t, der kan ses i figur 3.8. Beslutningen omkring hvilke information der skal fravælges er bestemt i et sigmoid layer, kaldet *forget gate layer*. Her anvendes h_{t-1} og x_t og udsender et tal mellem 0 og 1, for hvert tal i *cell state* C_{t-1}.

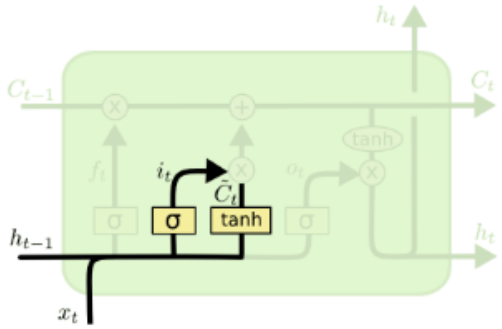
Figure 3.9: Forget Gate



$$f_t = \sigma(W_f * [h_{t-1}, x_t] + b_f) \tag{3.15}$$

Det næste step i LSTM er at beslutte hvilket nyt information der skal opbevares i C_t. Dette foregår i to trin, som figur 3.9 illustrerer. Første trin er et sigmoid layer kaldet *input gate layer*, der bestemmer hvilke værdier der skal opdateres. Efterfølgende vil en tanh layer kreere en vektor af de ny kandidatværdier, $\tilde{C}_t$, der kan tilføjes til C_t. Næste trin er en kombination af *input gate* og tanh, der beslutter hvilke værdier der skal opdateres i C_t.

Figure 3.10: Input Gate del 1

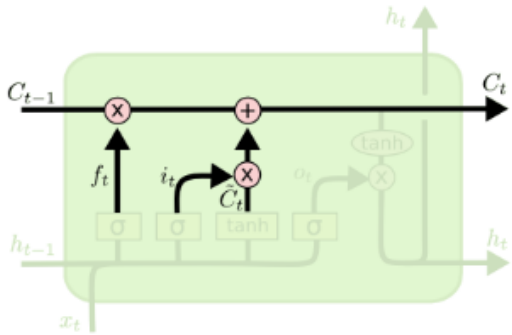


$$i_t = \sigma(W_i * [h_{t-1}, x_t] + b_i) \tag{3.16}$$

$$\tilde{C}_t = tanh(W_C * [h_{t-1}, x_t] + b_C) \tag{3.17}$$

Næste step i LSTM er at opdatere den gamle *cell state*, C_{t-1}, til den nye *cell state* C_t illustret i figur 3.11. Det foregående trin, besluttede allerede hvad LSTM-modellen skulle gøre, nu skal beslutningen dog udføres. Dette kommer til udtryk i ligning 3,18, hvor den gamle **cell state**, C_t, multipliceres med f_t. Derefter tilføjes $i_t * \tilde{C}_t$, der er de nye værdier, skaleret efter hvor meget LSTM modellen opdaterede hver værdi.

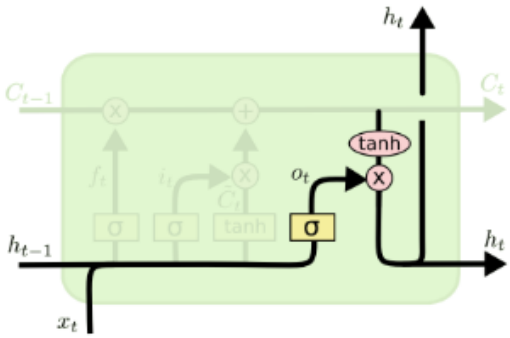
Figure 3.11: Input Gate del 2



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{3.18}$$

Sidste trin er at afgøre hvad output skal være. Outputtet er baseret på C_t, men dog en filtreret version. Første step er lade informationen komme igennem et sigmoid layer, der bestemmer hvilken del af C_t der bliver til output. Derefter bliver C_t kørt gennem en *tanh* funktion, for at skalere værdierne mellem -1 og 1, og derefter at multiplicere med output fra sigmoid gaten. Dette er illustreret i figur 3.12.

Figure 3.12: Output Gate



$$o_t = \sigma(W_o * [h_{t-1}, x_t] + b_o) \tag{3.19}$$

$$h_t = o_t * tanh(C_t) \tag{3.20}$$

Dette var således afrundningen på gennemgangen af LSTM-modellen i sig selv, men før en delkonklusion vil næste afsnit behandle teknikken bag LSTM med et Rolling Window

3.6 Rolling Window

I empirisk analyse inden for økonomiske og finansielle emnefelter, er rolling window[7] vidt brugt til test af modelstruktur og validering. I dette speciale anvendes rolling window til prædiktering af tidsseriedata. Et problem relateret til tidsserie er vanskeligheden ved udvælgelse af et testsæt til modeludvælgelse. I almindelige indstillinger for ML opdeles dataene i et træningssæt og et testsæt ved tilfældighed. Specialet kan ikke blot anvende valideringsmetoden som ved andre problemer, da al data er sekvensopdelt. Rolling window kan hjælpe med at løse problemet med valg af model i tidsseriedata. I stedet for at bygge en enkelt model til forudsigelse, fokuserer rolling window på at opbygge en modelstruktur til forudsigelse af tidsserier. I tilfældet af, N, sekvenser i dataet, og rolling window bruger sekvenslængden på p. I dette tilfælde vil hele modellen bestå af N-p modeller, der forudsiger N-p tidsserier, og disse modeller har ikke samme vægte. Mange undersøgelser anvender lineær regression med Rolling Window. Imidlertid er eksisterende modeller med Rolling Window ikke i stand til at bruge funktionen af sekventielle data, da f.eks. lineær regression eller Bayesian neuralt netværk ikke kan afspejle denne sekventielle karakteristika. Derfor foreslår specialet at anbruge af Rolling Window i LSTM til præditering af aktier.

3.7 Opsumering

Det kan nu konkluderes at forskellige typer af neurale netværk findes, og alle med deres fordele og ulemper. FNN kunne være nyttigt i enkelte klassificeringsproblemer, og mens RNN kunne håndtere tidsserier. Uanset type af netværk så består et neuralt netværk af neuroner, der interagerer med hinanden gennem vægtforbindelser. Hver neuron er konstrueret ud fra en aktiveringsfunktion, der ændrer input til et output. Vægtene justeres i henhold til fejl beregnet ud fra forskellen mellem netværkets output og targetoutputtet fra træningsdataet. Hver tilsluttet neuron stemmer således efter, for at ændre vægt og reducere fejlen. I RNN har dette resulteret i eksponentiel udvikling, der fører til forsvindende eller eksploderende gradienter. RNN trænes i denne forstand gentagne gange, men uden at lære de langsigtede afhængigheder. LSTM løser dette ved at introducere en hukommelse men samtidig en rettidig forglemmelse (forgetgate). Dette gør det muligt for netværket at gemme information på lang sigt, men samtidige at glemme irrelevante sammenhænge. LSTM med sin forgetgate starter ved at modtage input fra et træningsdatasæt og sender det igennem et hidden layer, der producerer et output. I det hidden layer findes hukommelsen der modtager inputtet og ændrer, hvilket resulterer i output. Input til hukommelsen ganges med en gate-værdi for at bestemme, hvor vigtigt inputtet er. Det føres derefter gennem neuronens kerne CEC, som gemmer en kopi af træningsindgangen til senere brug. Inputtet ganges derefter med output-værdien, idet det bestemmes, om outputtet er berettiget til at forlade cellen eller ej. Outputtet der passerer til det endelige outputlayer, sammenlignes med den targetoutputtet, der også leveres fra træningsdataet. Fra dette beregnes fejl som tilbagesendes til det tidligere layer og ankommer til outputgaten. Målet er, som sagt, her at reducere neuronens samlede fejl. Da neuronets output afhænger af input- og forgetgaten, justerer begge deres egne vægtninger, altså en gradientafstamning. Ved at justere vægtene gentagne gange og lade forgetgatens vægt ændre sig med tiden, glemmer netværket langsomt de oplysninger, det har lagret, afhængigt af hvor meget det bidrager til netværkets samlede fejl på det aktuelle tidspunkt. Netværket ser effektivt på et par input- og outputværdier fra træningsdata og lærer datamønstrene.

[7] Rolling window kan også benævnes med silding eller moving window.

Litteraturreview 4

Forecasting af finansielle aktiver har givet anledning til meget forskning, idet en universal løsning til foreudsigelse af bl.a. morgendagens aktiekurs ikke eksisterer. Koblingen mellem at bruge lineære og ikke-lineære modeller til forecasting er derfor ikke noget nyt emne. Der er adskillige videnskabelige artikler, der har undersøgt dette grænseområde. Flere artikler vedrørende de ikke-linære ML-modeller, kan dateres tilbage til start 90'erne, hvor de første DL-modeller vinder indpas. I begyndelsen var de fleste DL-modeller ikke udviklet specifikt til forecasting af aktiekurser og måtte se sig slået af de mere traditionelle modeller. Over tid har udviklingen med DL-modellerne taget fart og modellerne er blevet mere præcise i deres mønstergenkendelse. Derfor finder nyere litteratur oftest ud af, at de tradionelle linære modeller ikke kan hamle op med de ikke-linære modeller.

Arbejdet med data og preprocessing har stor betydning for hvordan de forskellige modeller klarer sig. F.eks. finder Namini og Namin i deres artikel: "A Comparison of ARIMA and LSTM in Forecasting Time Series" fra 2018, ud af at deres LSTM er 84-87 pct. bedre end deres ARIMA ud fra reduktion i RMSE. De tester deres to modeller både for finansielt og økonomisk data. Det finansielle data består i forecasting af daglige observationer fra følgende indekser: Nikkei 225, NASDAQ, Hang Seng indeks, S&P 500 og Dow Jones industrial average indeks med ugenlige og månedelige observationer. Mens det økonomiske data består af forecasting af "Medical care commodities for all urban consumers", "Housing for all urban consumers", "the trade-weighted U.S. dollar", "Food and Beverages for all urban consumers", "M1 Money Stock" og "Transportation for all urban consumers" (Namin and Namin, 2018).

De to forfattere bag artiklen finder ud af, sammen med Tavakoli i 2019, at LSTM i gennemsnit er 88,1 pct. bedre end ARIMA. Her forecaster de aktieindekser med både daglige, ugenlige og månedlige observationer med overordnet samme resultater uagtet tidsramme, dog klarer ARIMA sig en smule bedre ved månedlige observationer fremfor daglige. Således kan udvælgelse, dataets natur og måden hvorpå dataet behandles være altafgørende for resultatet. Derfor har specialet valgt at søge inspiration i disse artikler men vil ligeledes afvige fra disse, da specialets udvalgte data kræver en individuel vurdering ift. fremgangsmåde og behandling (Namin et al., 2019).

Derudover findes der litteratur der ikke vil mene at ARIMA overhovedet burde overvejes i sammenhæng med atkiekurser, men istedet en GARCH-model. Her kan nævnes Kim og Won's bidrag fra 2018, hvor de to rigtig nok påpeger at volatilitet spiller en afgørende rolle på det finansielle marked, og deres belæg for at bruge en GARCH-model fremfor ARIMA. Dog må de traditionelle GARCH-typer[1] se sig slået af LSTM med daglige observationer fra 2 forskellige aktieindekser. LSTM opnår både lavere MAE, MSE, HMAE og HMSE end GARCH-typerne. Samme konklusion opnåes af Bao et al. i 2019 og dermed går

[1] GARCH, the exponential GARCH (EGARCH) og the exponentially weighted moving average (EWMA).

begge artikler videre med at konstruere en hybrid mellem de to modeltyper (Kim and Won, 2018). Kim og Won kombinerer således de tre typer GARCH med deres LSTM og når endnu bedre resultater og de absolut laveste fejl, mens Bao et al. kombinerer en ARIMA med ANN (et andet NN) og opnår ligeledes bedre forecasts end de to seperate modeller kan, målt på MAE og RMSE (Bao et al., 2019).
Dette speciale har dermed ikke kunne finde litteratur på en kombination af ARIMA og LSTM til forecasting af finansielt data. Dog skal det ikke være usagt, at (Deorukhkar et al., 2019) i deres forsøg på forecaste morgendagens priser på en række aktier, bruger en hybrid af LSTM, RNN, ARIMA og Sentiment Analysis model, som slår alle seperate modeller med en forklaringsgrad på 94.49 pct.

Dette var således en kort gennemgang af tidligere værker skrevet på området af NN og ARIMA forecasting aktieindekser. Dermed kan specialet herefter begynde sin egen analyse og således vil næste afsnit præsentere data, samt opbyggelse og udvælgelse af ARIMA- og LSTM-model.

Databehandling og forecast 5

Afsnittet vil indeholde en beskrivelse af det anvendte data til analysen. Der vil forekomme grafisk afbildning, beskrivelse af karakteristika af data samt forarbejdelse. Til sidst vil der diskuteres fordele og ulemper ved forecastingprocesser og valget for specialets proces.

5.1 Deskriptiv statistik

Specialet vil anvende fem udvalgte aktier til forecasting:

Johnson & Johnson(JNJ) er et amerikansk multinational medicinalkonglomerat. Virksomheden fremstiller udover medicin også medicinsk udstyr og husholdningsprodukter.

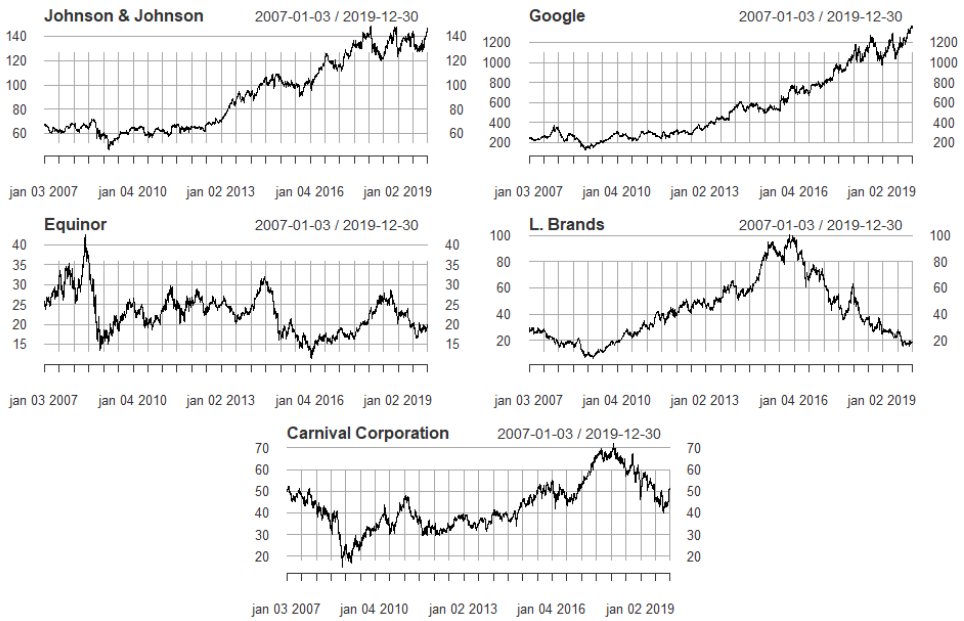
Google(GOOG) eller Alphabet Inc. leverer onlineannonceringstjenester i hele verden og tilbyder ydelser og annoncering. Virksomheden opererer gennem segmenterne Google og andre indsatser.

Equinor ASA(EQNR) er et energiselskab, der udforsker, producerer, transporterer, raffinerer og markedsfører olie- og petroleumsproducerede produkter og andre former for energi i særligt Norge (Statoil) og internationalt.

L Brands, Inc.(LB) er en specialforhandler af beklædningsgenstande, skønhedsprodukter, personlige plejeprodukter og duftprodukter til hjemmet.

Carnival Corporation & Plc.(CCL) er en britisk-amerikansk krydstogtsoperatør.

Figure 5.1: Udvikling i de fem udvalgte aktier



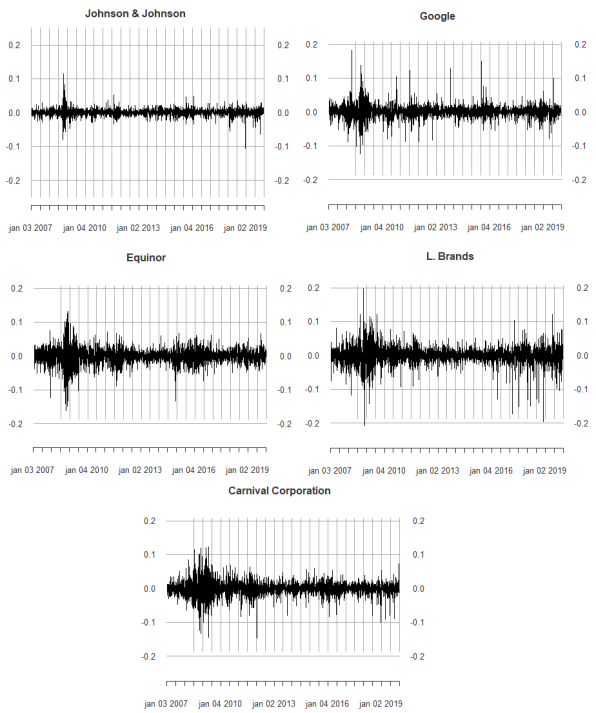
Aktierne er udvalgt for at skabe spredning i porteføljen, der normalt ønskes af en investor. Da samlingen af alle æg i samme kurv, udsætter investeringer for større risiko. Derfor er aktieudvælgelsen i sig selv relativ tilfældigt, men valgt grundet deres forskellige sektortilhørsforhold. Ligeledes afviges der fra at arbejde med indeks som i kilderne fra kapitel 4, og dermed fås et mere sandfærdigt billede af prædikteringsevner

hos ARIMA og LSTM. Dette skal forstås på den måde, at aktier varierer i deres natur, da nogle oplever store fluktueringer, mens andre er mere stabil i deres kurs over tid. Eksempelvis ses det i tabel 5.1, at GOOG har en relativ stor standard afvigelse og varians end de fleste gængse indekser oplever.

Den gældende periode for dataet strækker sig fra primo januar 2007 til ultimo december 2019, der er grafisk illustreret i figur 5.1. Perioden består i alt af 3271 observationer, der indikerer den daglige lukningspris for aktierne. Figur 5.1 indholder både perioderne før, under og efter finanskrisen i 2008, hvor alle aktier oplever næsten eller mere end en halvering i deres aktiepris i perioden fra 2007 til 2009. Aktiemarkedet var på daværende tidspunkt kendetegnet som et bear-marked, mens perioden efter 2009/10 er kendt som et bull-marked med stigende aktiepriser. Dog med undtagelse af EQNR, som 'kun' går fra en kurs på omkring de 15 USD i ultimo 2008 til omkring 20 USD i ultimo 2019, og LB's kurs som vender omkring ultimo 2016, mens de tre andre mere end tredobles.

Udviklingen i de fem aktier tyder på trend og drift i lukkekurser. Dette var en af de nævnte problemstillinger i kapitel 2 ved tidsserie data. Ved håndtering af problemet med en trend og drift, kan differentiering anvendes og således kan ændringen i kursen opnås svarende til det daglige afkast. En augmented Dickey-Fuller test på det differenceret data er ligeledes blevet udført, og kan aflæses i tabel 5.1, og sikrer stationaritet i dataet.

Figure 5.2: Dagligt afkast af de fem udvalgte aktier



I figur 5.2 fremstår tidsserierne efter at være differentieret af første orden, som værende stationær. I figur 5.2 er det differenceret data eller dagelige afkast omregnet til procent. Derved fremgår, at årene omkring 2008 og 2009 var præget af høj volatilitet, særligt for GOOG, CCL og LB, som igen kan spores tilbage til finanskrisens påvirkning. Hvilket rent intuitivt giver god mening, at de største ændringer i kursen forekommer i sådan en periode, hvor usikkerhed, spekulation, konkurs osv. indtræder. GOOG er ligeledes den aktie med højst varians, mens EQNR har den laveste (se tabel 5.1), og dermed mindre prisfølsomme. Dette er endnu en begrundelse for deres tilvalg i dette speciale, da en investor oftest søger

aktier som reagerer forskelligt på f.eks. konjunkturudviklingen. Dette er også i overensstemmelse med, at resten af perioden ikke oplever samme store ændringer, dog med enkelte undtagelser tilfældigt spredt i tidsserierne. Dette kunne hænge sammen med virksomhedsspekfikke nyheder, der sender aktiekursen ud i enten en større stigning eller fald. Tidsperioden 2007-2019 er valgt for at opnå så mange observationer som muligt til forecasting.

Specialets formål er forecasting af aktier, og inden forarbejdning af dataet påbegyndes, vil der gennemgås nogle af tidsseriens karakteristikaer igennem deskriptiv statistik. Deskriptiv statistik er en kombination af forskellige metoder, der har til formål at beskrive og opsummere det anvendte data. Deskriptiv statistik beskriver kun det indsamlede data og generaliserer dermed ikke på baggrund af dataet. Det er derfor ikke muligt at kunne lave forudsigelser på baggrund af statistikken.

Table 5.1: Deskriptiv statistik

	JNJ	GOOG	EQNR	LB	CCL
Middelværdi	90.51	550.26	22.82	42.65	43.41
Standard afvigelsen	28.97	333.22	4.96	23.24	11.26
Varians	839.51	111038.8	24.63	540.10	126.70
Maksimum	148.14	1361.17	42.47	100.22	71.94
Minimum	46.60	128.24	11.38	6.26	15.02
Skewness	0.41	0.77	0.47	0.63	0.39
Kurtosis	-1.32	-0.73	0.24	-0.49	-0.30
Jarque Bera p-værdi	2.2e-16	2.2e-16	2.2e-16	2.2e-16	2.2e-16
Augmented Dickey-Fuller p-værdi	>0.01	>0.01	>0.01	>0.01	>0.01

I tabel 5.1 er der en oversigt over nogle karakteristika af de udvalgte aktier, som middelværdi, standard afvigelse og varians. Middelværdien eller gennemsnitsværdien kan ikke sige så meget, da forskellige trends forekommer i tidserierne. Varians og standard afvigelse er udtryk for det samme, og er som tidligere nævnt særligt høj for GOOG, grundet dens store positive aktiekursudvikling.
Maksimum og minimum værdierne giver ligeledes et indtryk af hvor stor spredning der forekommer og således tidseriernes forskellighed.
Jarque Bera testen anvendes til at tjekke om hvorvidt fordelingen i et datasæt er normalfordelt. For at definere Jarque Bera, udregnes først skewness (S) og kurtosis (K). Skewness af en tilfældig variable Y er defineret som:

$$S = \frac{E[(Y-\mu)^3]}{E[(Y-\mu)^2]^{\frac{3}{2}}} \tag{5.1}$$

Dermed er skewness med andre ord en indikator for symmetrien i fordelingen. Med en skewness værdi lig nul vil fordelingen af observationerne være distribueret symmetrisk omkring middelværdien. Med en negativ værdi vil fordelingen være venstreskæv og med en positiv værdi tilsvarende højreskæv. I det anvendte data er skewness positiv for alle, hvilket betyder at dataet er højreskæv og indikerer dermed at særligt store positive afkast forekommer i data og særligt GOOG (0,77), som kan hænge sammen med den positive udvikling i kursen. Kurtosis af en tilfældig variabel Y er defineret som:

$$K = \frac{E[(Y-\mu)^4]}{E[(Y-\mu)^2]^2} \tag{5.2}$$

Kurtosis variablen er en indikator for om 'halen(-erne)' i skewness er bred eller smal. En standard normal fordeling har en kurtosis lig tre, hvilket også kaldes mesokurtisk. De fem udvalgte aktier har alle

en negativ eller en kurtosisværdi lavere end tre, og indikerer at halerne er smalle (platykurtic). Dette fortæller samtidig at der ikke forekommer mange outliers i tidsserierne. Modsat er en positiv kurtosisværdi (leptokurtic) en indikation om brede haler, der indikerer at der forekommer outliers i tidsserien. Data præget af platykurtic er som regel derved også lettere at prædiktere, da data som sagt er udsat for færre 'ekstreme' observationer.

Jarque Bera testen er indenfor stastistikkens verden en *goodness-of-fit* test, hvor formålet er at teste hvor godt de observerede værdier stemmer overens med en almindelig normalfordeling. Til formålet opstilles en nulhypotese, som værende at dataet er normalfordelt. Jarque Bera testen er defineret som følgende:

$$JB = \frac{n}{6}\Big(S^2 + \frac{(K-3)^2}{4}\Big) \tag{5.3}$$

hvor n er antal af observationer, S er skewness og K er lig kurtosis (Lovric, 2011). Med en p-værdi på 2.2e-16 for alle variabler kan nulhypotesen forkastes om at tidsserien er normalt fordelt. Dette stemmer ligeledes overens med skewness og kurtosis værdierne.

Den sidste karakteristika er Dickey-Fuller test, som blev beskrevet i kapitel 2.4.2. Testen er modsat de andre tal, udført på det differenceret data og ikke de originale data. Formålet med testen er kort fortalt at tjekke for stationaritet i tidsserien. Nulhypotesen bygger på at der forekommer en unit root i tidsserien og dermed gør at tidsserien ikke er stationær. Med p-værdi på >0.01 afvises nulhypotesen om at tidsserien indeholder en unit root, og dermed konkluderes det at tidsserien er stationær.

Den deskriptiv statistik giver et godt overblik over dataets natur, men fortæller ikke noget nærmere omkring hvordan dataet bedst kan gengives og prædikteres, og dette vil næste afsnit derfor tage fat i.

5.2 Udvælgelse af ARIMA-model

For at kunne udvælge den passende ARIMA-model til de 5 aktier, har specialet valgt at anvende AIC eller Akaike information criterion. Dette er ligeledes den gængse udvælgelsesmetode, der ligeledes benyttes i litteraturen fra kapitel 4.

$$AIC = 2k - 2\ln(L_{max}) \tag{5.4}$$

Hvor L_{max} er log-likelihood og ud k er antallet af estimeret parametre. Andre muligheder kunne ligge i Schwarz information criterion (SIC)[1] eller Hannan-Quinn (HQ) der ligeledes fungerer som indikator på hvor godt et fit en given ARIMA-model har på dataet. Alle tre har det tilfælles at den laveste værdi udtrykker det bedste fit. Ved tilføjelse af flere paramtre (k) opnås oftest lavere værdier da likelihood eller forklaringsgraden stiger i takt med dette. Dog kan det ske at tilføjede parametre reelt ingen værdi har for forklaringsgraden, og dette løses i de tre indikationer ved indførelse af et 'straftillæg' på antallet af parameter. BIC og HQ er dog hårdere end AIC, og vælger dermed oftest modeller med færre parametre. AIC tager modsat de andre to, ikke højde for antallet af observationer, n. SIC og HQ kan skrives op følgende måde:

$$SIC = \ln(n)k - 2\ln(L_{max}) \tag{5.5}$$

$$HQ = -2L_{max} + 2k * \ln(\ln(n)) \tag{5.6}$$

[1] Kaldes også Bayesian information criterion eller BIC.

Derudover beskæres de 5 tidsserier fra 3271 til 3241 observationer, da specialet ønsker at forecaste de sidste 30 observationer, og således undlades disse i modeludvægelsen. Hvilket har givet følgende resultater:

Table 5.2: Modeludvælgelse ud fra AIC for de udvalgte aktier

	JNJ	GOOG	EQNR	LB	CCL
ARIMA(1,1,0)	9026.17	23920.87	4706.89	**8624.92**	8624.92
ARIMA(1,1,1)	9025.53	23917.23	**4706.68**	8626.92	**8626.92**
ARIMA(1,1,4)*	9008.53	**23903.44**	4713.66	8633.74	8633.74
ARIMA(2,1,2)	**9003.56**	23907.68	4711.13	630.96	8630.96

* with drift

Bilag A.1 er en udvidet udgave af de forskellige ARIMA-modeller som specialet har testet de 5 tidsserier for. Derfor er en ARIMA(0,1,0) eller random walk også blevet testet. Der er således ingen af aktierne som kan understøtte Eugene Fama's teori om ECM eller random walk theory som i 1973 blev udviklet og udgivet i bogen *"A random walk down wall street"* af Burton Malkiel. Bogen skabte en del debat i både den finansielle og akademiske verden, og bragte "the efficient market hypothesis" tilbage i fokus. Den kortsigtet tilfældighed der kan forekomme ved aktier tilsiger dermed at investorer burde være mere passive i forvaltingen af deres veldiversificeret portefølje. Burton Malkiel kunne i sin bog, konkludere følgende populær sætning:

"*A blindfolded monkey throwing darts at a newspaper's financial pages could select a portfolio that would do just as well as one carefully selected by experts.*"

Dette er ligeledes grunden til at dette speciale udover sit prædiktering af aktier, vil forsøge at sammensætte en portefølje ud fra det. Dermed tilføje endnu en dimension i specialet ift. at sætte ARIMA op mod LSTM. ARIMA-modeller skal kunne prædiktere uden seriel korrelation og testes med en Ljung-Box test med nulhypotese lig tidsserien ikke lider af seriel korrelation.

Table 5.3: Ljung-Box test

	JNJ ARIMA(2,1,2)	GOOG ARIMA(1,1,4) with drift	EQNR ARIMA(1,1,1)	LB ARIMA(1,1,0)	CCL ARIMA(1,1,1)
Ljung-Box test P-value	0.84	3.783e-05	0.64	0.95	0.68

Det kan aflæses ud fra tabel 5.3, at både JNJ, EQNR, LB og CCL ikke lider af seriel korrelation, og herfra kan prædikteres. Modsat lider GOOG af seriel korrelation, og skal bearbejdes ydereligere for at kunne prædikteres. Hertil findes der flere metoder, som f.eks. tilføjelse af flere AR-lags der kan eliminere seriel korrelation. Dette er således forsøgt med GOOG op til lag 40[2], hvor en p-value for ARIMA(34,1,4) (med drift) for GOOG ligger på 0.19, og dermed kan hulhypotesen om ingen seriel korrelation ikke forekastes. Ved denne metode opnås der ligeledes en lavere AIC værdi for GOOG på 23856. En anden metode ville være at føre tidsserien igennem en GARCH-model for at komme af med seriel korrelation ligesom Kim og Won i Kapitel 4.

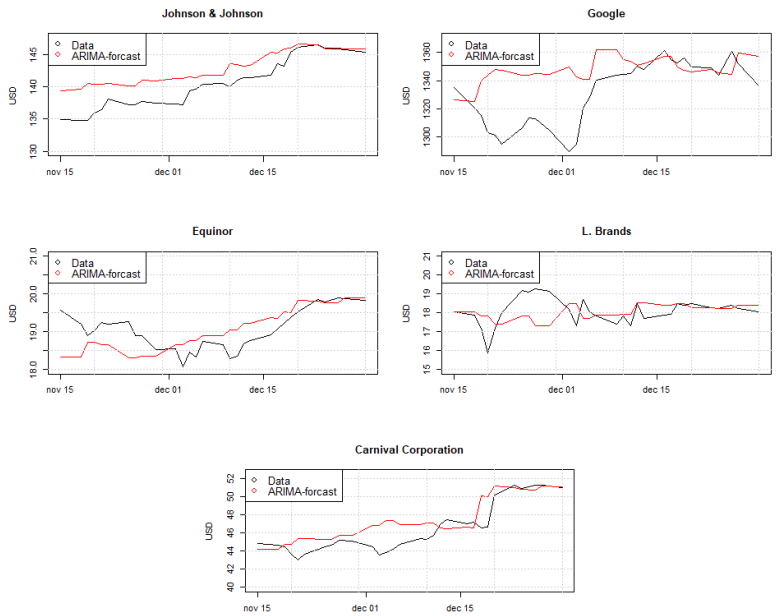
[2]Grundet manglende computerkraft kan specialet ikke teste tidsserier med mere end 40 lags.

Specialet har derfor valgt at udføre en ARIMA(2,1,2) for JNJ, en ARIMA(1,1,1) for EQNR, en ARIMA(1,1,1) for CCL, en ARIMA(1,1,0) for LB og en ARIMA(34,1,4) med drift for GOOG grundet at tidsserien består af mere end 3000 observationer og derved ikke lider noget større tab af observationer, ved kreere 34 lags.

5.2.1 Forecasting ARIMA

Det findes en række forskellige processer der kan anvendes til forecasting, og i dette afsnit vil der fokuseres nærmere på to typer: *one-step ahead forecast* og *multi-step ahead forecast*. Metodernes tilgang til forecast ligger i navnet på dem. Hvor one-step ahead forecaster en enkelt periode hen og multi-step forecaster flere perioder frem. Specialet har valgt at gå med one-step ahead metoden, idet multi-step ahead forecast kan formindske kvaliteten af forecastet, da disse modeller oftest konvergerer mod en ligevægt. Samtidig bliver hver forecast opdateret fra hver forhenværende periode ved brug af one-step ahead med Rolling Window, og dermed prædiketeres mere akkurate forecasts ift. at specialet ønsker at analysere og forecaste de daglige kursændringer. Derudover er dette også metoden, som kilderne fra kapitlet 4 benytter sig af.

Figure 5.3: Rullende forecast for 15/11 til 30/12-19 med ARIMA



Prædiktionerne for de fem tidsserier er forskellige fra de sande værdi. For JNJ starter de to linjer langt fra hinanden, men det forekommer over tid, at de konvergerer mod samme værdi i slutningen af perioden. GOOG slutter med relativt de samme værdier, men oplever i starten et stort gab mellem forcastet og dataet, men konvergerer igen i starten af dec. EQNR er ligeledes i starten, præget af store forskelle mellem prædiktering og data, men får rettet op efter start december og prædikterer en fin stigning der bliver genspejlet i dataet. LB er relavtivt konstant i sin kurs og oplever ikke de store udsving, dog kunne det syntes at prædiktering ikke opfanger udsvingene særligt godt, da den tendenserer en vandret linje igennem dataets udsving. Prædikteringerne for CCL er derimod bedre ift. udsving, da de to linjer ligger forholdvis tæt op af hinanden under hele forecastingsperioden.

Disse foreløbige resultater fra ARIMA vil kort blive lagt til side, til fordel for opsætning og resultater for LSTM, og herefter vil sammenligningen mellem de to modeller begynde.

5.3 Opsætning af LSTM

I jagten på at finde den rette LSTM-model til specialets data, skal det gøres klart at opsætning og kombination af konstrueringer af et LSTM-netværk kan udføres på mange måder. Antallet af layers, neuroner i hvert layer, evt. dropout mellem hvert layer samt antallet af dense, type af optimizer, andel af træningsæt kontra testsæt og størrelsen på batch og antallet af epochs er alle ting som kan justeres i denne jagt på det bedste forecast. Mulighederne for en LSTM-model er derved endeløse, og det bedste forecast kan således være svært at opnå. Problemet ved alle disse mange kombinationsmuligheder er ligeledes, at metoden hvorpå den bedste model er ikke fuldt beskrevet og derved kan kun enkelte tommelfingerregler følges. Derfor har specialet bl.a. søgt inspiration i kilderne fra Litteraturreview, da disse gav specialet et fundament at stå på og arbejde ud fra. Kilderne fra kapitel 4 beskriver dog ikke til fulde hvordan deres LSTM-model er sat op, men tabel 5.4 præsenteres et kortfattet sammendrag:

Table 5.4: LSTM-opbygning - litterturreview

	S. S. Namin & A. S. Namin	S. S. Namin & A. S. Namin & N. Tavakoli	H. Y. Kim & C. H. Won	T. Q. Bao & co. (bruger en ANN)	O. S. Deorukhkar & co.
Trænings-/testsæt (i %)	70/30	70/30	U	98/2	U
Layers	1	1	5	U	1
Neuron i hvert layer	4	4	10, 4, 1, 5, 1	U	500
Epochs	1	1	150	U	100
Batch	U	42	U	U	U
Type af optimizer	Adam	Adam	Adam	U	U
Dropout (i %)	Ingen dropout	Ingen dropout	3 (0.3, 0.8, 0.8)	U	U

U - uoplyst.

Som det kan aflæses i tabellen er opbygningerne i de diverse kilder meget forskellig (og tildels ufuldkommen da ikke alle oplysninger er til rådighed). Dette handler i høj grad om at dataet er forskelligt fra kilde til kilde, og dermed ikke behandles på samme måde. Således har specialets resultat fra disse LSTM-opbygninger ikke kunne bruges, da fejlprædikteringerne blev for uacceptabelt store. Specialet har som sagt, således med afsæt i disse kilder opbygget sin egen LSTM-model.

Jagten på den rette LSTM-model kan på mange måder minder og jagten efter den ARIMA-model. Istedet for at finde den laveste AIC, bruges RSME til at finde den rette LSTM-model. Herfra blev de første modeller kørt med 32 batch(s) og 25 epoch, som har sikret lavere computertid. Samtidig tog specialet belslutningen om at anvende en Adam optimizer med en relu-aktiveringsfunktion samt en 80/20 fordeling ift. til træning- og testsæt som standard, og vil ikke prøve at forbedre modellerne ved ændring af disse parametre. Adam optimeringsalgoritmen er en videreudvikling af den stokastiske gradient descent, der har haft større evne til at håndtere DL (Brownlee, 2017). En videre forklaring af hyperparameterudvælgelse forefindes i appendix A, hvor udvælgelsen bliver gennemgået. Specialet vil derfor i næste afsnit kort opsummere og præsentere de udvalgt LSTM-modeller for de 5 tidsserier.

5.4 Udvælgelse af LSTM-modeller

Specialet vil prøve at opstille den bedste mulige model for at kunne prædiktere *one-step-ahead* forecasting af fem aktier. Dermed skal *hyperparameterne* optimeres, for at modellerne får den maksimale performanceevne. Hyperparameter kan oftest forveksles med model parameter, men det er vigtigt at kende forskellen på disse to termer:
Model parameter er parameter, der er determineret gennem træningssættet, disse er tilpasset parametre og kan udgøre vægte og bias i et NN.
Hyperparameters er egenskaber der er med til at styre hele processen under træningen i modellen. Parametrene indeholder variabler, der er bestemmende for NN strukturen f.eks. skjulte units. Dermed er hyperparameter noget der bliver optimeret inden træningen af modellen kan begynde.
En række af disse parameter der kan blive optimeret for at højne modellens performance evne er som følgende:

- Layers og neuroner
- Antal af epochs og batch
- Læringsrate

Hyperparameter er derfor meget vigtige, da de direkte kontrollerer adfærden i træningsalgoritmen og dermed modellens performance. Udvælgelsen af de rigtige hyperparameter er derfor en stor faktor i arkitekturen i et NN med fokus på læringen. Specialet har således tunet disse førnævnte hyperparametre i jagten efter den optimale model.

Layers og neuroner
Det er en gængs metode at anvende et større antal af hidden input end den reelle input størrelse. Modellen har et input dimension på (2557, 60, 1). 2557 kan forklares som de 80 pct. af 3241 som træningsættet består af. Værdien 60 er selve størrelsen på inputtet og da det forekommer at algoritmen i computeren nemmere kan håndtere tal der i potens af to. (2, 4, 8, 16, 32 osv.). Af den årsag og tommelfinger reglen fra Jojo Moolayil B, er første hidden layer i modellerne sat til at være 128.
Antallet af layers i modellen spiller en stor rolle for prædikteringen af forecastet. Det er oftest at tre layers modeller vil outperform to layers modeller. Det forekommer at der ved mere end tre layer, vil marginal effekt enten falde eller blive negativ, idet kompleksiteten vokser for meget i modellen. Dette blev ligeledes testet i modellerne for forecasting af de fem finansielle aktiver. Her fremgik det at modellerne var mest akkurate ved modellerne i besiddelse af to og tre layer, over modeller med fire layer.
Dense layer er det regelmæssigt forbindende layer i et NN. Dense layer udfører følgende operation for at omdanne input til output.

$$output = activation(dot(input, kernel) + bias)$$

Med de ovenstående parameter, blev der udvalgt i alt fem forskellige modeller til anvendelsen af forecast gennem LSTM på de fem udvalgte aktier. Alle modeller er bygget ud fra specialets hyperparametertuning afsnit i appendiks B. Ud fra de fem forskellige modeller er der lavet 10 forskellige varianter med forskellig antal layers og neuroner, hvoraf to af ti mulige modeller viste sig bedst. Begge modeller besidder to

layers, dog med forskellige hyperparameter værdier:

Figure 5.4: LSTM opbygning for JNJ

Figure 5.5: LSTM model opbygning for GOOG

Figure 5.6: LSTM model opbygning af EQNR

Figure 5.7: LSTM model opbygning af LB

Figure 5.8: LSTM model opbygning af CCL

Modellerne for JNJ og CCL følger begge strukturen fra model 2, hvorimod de tre (GOOG, EQNR og LB) andre følger model 1. Forskellen mellem de to modeller er at model 1 benytter sig af 128 neuroner i første layer og efterfulgt en halvering til 64 i andet layer. Hvorimod model 2 holder antallet af neuroner i begge layer konstant. Hver model på nær JNJ og LB har en forskellige dropout værdi. Et dropout layer, betyder at noderne i modellen tilfældigt vil blive deaktiveret. Dette anvendes for at styrke træningen af modellen, da det tvinger modellen til at lære med forskellige noder under uændret forhold, som sænker lærings tempoet.

Epochs

For at vælge det optimale sæt af *epochs* rettes fokus hen mod tabsfunktionen. Det fremkommer at der vælges det antal epochs indtil modellens tabsfunktion ikke længere er faldende. Da det forekommer at tabsfunktionen er minimalt faldende på forskellige tidspunkter, for de fem finansielle aktiver. Ud fra dette har specialet valgt at teste forskellige antal af epochs i alle modeller forskelligt med 50 epochs for JNJ og CCL, 25 for GOOG og LB og 500 for EQNR. For mere deskriptiv information refereres der til tabel B.10 i appendiks.

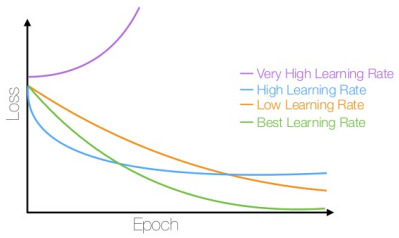
Batch size

Størrelse af batch er valgt til at være mellem 16 og 64, og dette blev bestemt i specialets hyperparametertuning afsnit i appendiks B. Dette blev således påvist at JNJ's, GOOG's, EQNR's og CCL's prædiktioner var bedst med kun 16 batch, mens LB var bedst med 64. For nærmere uddybelse af hvad batch size er, refereres der tilbage til kapitel 5.3.

Læringsrate

Vægtene i et NN, kan ikke beregnes ved brug af en analytisk metode tilgang, men i stedet skal vægtene afkodes gennem en empirisk optimeringsprocedure kaldet *stokastisk gradient descent*. En stokastisk gradient descent er en optimeringsalgoritme, der har til formål at estimere fejlgradienten for den aktuelle status i modellen, ved hjælp af eksempler fra træningssættet. Derefter kan vægtene ændres gennem *back propgagation*, der blev forklaret i kapitel 3.3.1. Læringsraten er således en hyper-parameter der er bygget til at kontrollere vægtene i det neurale netværk med hensyn til tabsgradienten. Den bestemmer hvor hurtigt, eller langsomt, modellen bevæger sig mod de optimale vægte. Figur 5.9 illustrerer en række forskellige scenarier for læringsrater. En lav læringsrate medføre at modellen i mødet med et lokalt minimum vil tage længere tid at konvergere. Det er ønskværdigt at opnå en læringsrate der både lærer relativt hurtigt og konvergerer mod et lokalt-/globalt minimum. Dermed skal tabsfunktionen formindskes i passende rate, mens antallet af epochs stiger[3].

Figure 5.9: Learning rate



I figur 5.10 til 5.14 er læringsraten for hver finansielt aktiv illustreret. Alle modellerne udvikler sig relativt på samme måde og har en en blanding mellem en høj og god læringsrate. Ved GOOG fremkommer det at *test* til tider, er højere end selve *train*. Dette skyldes overfitting, hvor dette referer til at modellen træner dataet for godt. Dette forekommer når detaljerne og støjen i træningssættet bliver for bekendt for modellen, og således stopper med at lære nye detaljer, som har en negativ indflydelse på modellens performanceevne. Dermed forekommer støj eller tilfældige udsving i datasættet bliver opsamlet og inddraget som koncepter af den given model. Problemet opstår i at disse koncepter ikke gælder for det nye data og dermed har en negativ påvirkning på modellens performanceevne til at generalisere dataet.

Figure 5.10: Læringsrate - JNJ

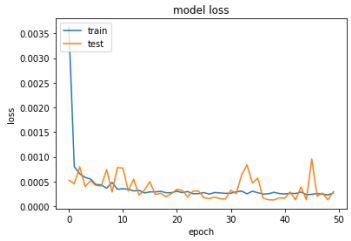


Figure 5.11: Læringsrate - GOOG

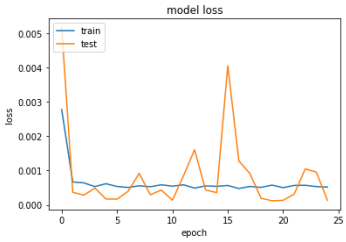


Figure 5.12: Læringsrate - EQNR

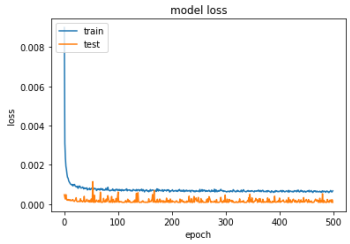
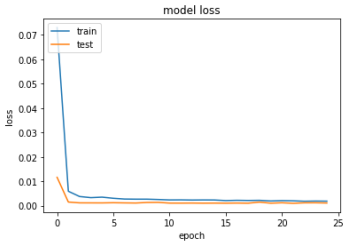
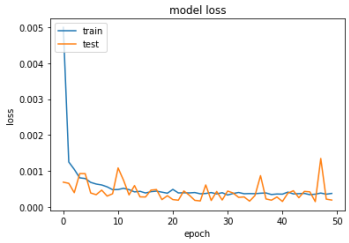


Figure 5.13: Læringsrate - LB



[3]Epoch er en hyperparameter, der definerer antallet af gange, hvor vidt algoritmen vil bearbejde hele træningssættet.

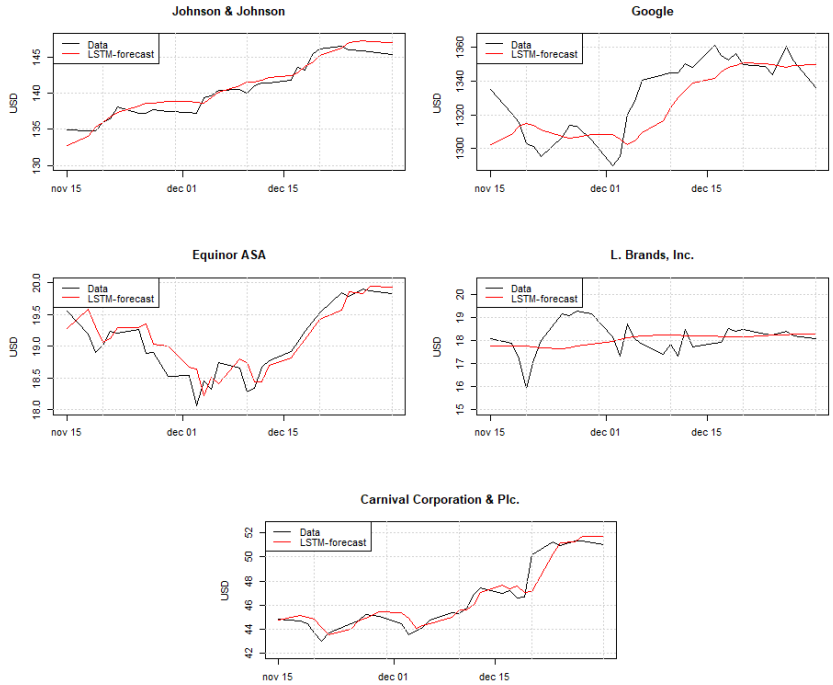
Figure 5.14: Læringsrate - CCL



Forhindrering af overfitting kan bl.a. være reducering af netværkets kapacitet, eller anvendelse eller forøgelse af *dropout layers*. Dette blev ligeledes forsøgt, dog uden ydereligere forbedringer. Samtidig kan termet overfitting også diskurteres, da dette ikke kan sættes op en matematisk formel. Derved kan man argumentere for at GOOG ikke overfitter, da det kun er enkelte spikes hvorpå test ligger højere end træning, og dermed overfitter modellen ikke systematisk. I forlængelse af dette, vurderes det at værdien på tabsfunktionen befinder sig på et meget lavt niveau (>0.005), at det ikke har den store effekt på modellens performance evne til at prædiktere fremtidige værdier. Således har specialet valgt ikke at gøre noget nærmere ved et evt. overfit for GOOG. Herpå kan specialet efter udvælgelse af LSTM-modeller, begynde prædikteringer af aktierne.

5.4.1 Forecasting LSTM

Figure 5.15: Rullende forecast for 15/11 til 30/12-19 med LSTM



I figur 5.15 kan det ses at LSTM er noget mere glidende i sin forecasting fremfor ARIMA som har nogen større udsving. Ved første øjekast fanger LSTM ligesom ARIMA JNJ relativ godt, mens forecastet for GOOG kunne syntes mere tvivlsomt. LSTM ses ud til at fange udsving og tendens bedre i EQNR. Forecastet af LB er en relativ lige linje, og det kan dermed være svært at sige om LSTM reelt fanger disse kursændringer. LSTM klarer det derimod bedre med forecasting af CCL og fanger næsten alle udsving. Indtil videre er disse prædikteringer kun kommenteret overfladisk og uden brug af statistiske tests. Derfor vil næste kapital være en mere dybdegående analyse af resultaterne fra de to modeller.

Sammenligning af ARIMA og LSTM 6

I vurderingen af ARIMA og LSTM har specialet valgt at opstille forskellige tests og bruge forskellige indikatorer for at opnå et så retvisende billede af prædikteringsevner blandt ARIMA og LSTM.
Det kan ses i tabel 6.1, at der ialt prædikteres 150 observationer, hvor ARIMA formår at forudsige 61 af observationernes retning mens LSTM kan forudsige 70. Denne måde at opgøre prædikteringsevner på, kaldes for "Sign ratio" (SR). Dette giver således ARIMA og LSTM en succesrate på henholdsvis 41 pct. og 47 pct. Således ville den hurtige konklusion være at LSTM er bedre end ARIMA, men at begge modeller er værre end hvis specialet havde valgt "A blindfolded monkey" til at gætte om kurserne ville gå op eller ned. ARIMA formår kun ved JNJ at prædiktere ligeså godt som tilfældighed ville gøre, med 50 pct., og klarer sig dårligst med CCL og rammer kun 8 ud af 30 rigtige. Det skal dog siges, at d. 27. dec. 2019 lukker CCL i samme kurs som dagen før, og dermed forekommer der hverken stigning eller fald. Ingen af prædikteringerne af både ARIMA og LSTM, tilskriver en stilstand i kurserne og foreudsiger dermed ikke samme pris som dagen før. Således tæller denne observation som en fejl ift. prædikteringsretning. Modsat ARIMA så oplever LSTM kun at prædiktere dårligere end ARIMA ved LB med 8 mod 12 korrekte prædikteringer. Istedet for at kigge på fortegn mellem data og forecast, så kan Theil's U indeks bruges. Indekset beskriver hvor høj præsitionen er i prædikteringerne, med en skala fra 0 til 1, hvor 0 er lig med en præcis gengivelse af data, mens 1 er det komplet modsatte (Bliemel, 1973). Dette viser at LSTM i gennemsnit er tættere på 0 og dermed en smule bedre end ARIMA. Det fremkommer igen at LB er en svær tidsserie for ARIMA og LSTM at prædiktere, men LSTM opnår en lavere værdi end ARIMA.
Ligeledes understøtter RMSE og MAE, LSTM og tilsiger at ARIMA i gennemsnit laver flere fejl end LSTM, og dermed har ringere forudsigelsesevne. LSTM laver færre fejl ved prædiktering af alle aktier målt på både RMSE og MAE, men igen er de to modeller meget tæt ved forecasting af LB. Samlet set går dette en smule imod hvad (Namin and Namin, 2018) og (Namin et al., 2019) finder frem til hvor RMSE i gennemsnit reduceres med mere end 80 pct. ved brug af LSTM frem for ARIMA på finansielle indekser. Dette blev ligeledes understøttet af (Bao et al., 2019), som også finder at ARIMA laver flere fejl ift. RMSE end en ANN. Specialet finder kun en gennemsnitlig reduktion af MAE på 39.79 pct., og 44.25 pct. gennemsnitlig reduktion i RMSE, når der kigges på det akkummelerede RMSE og MAE for de 5 aktier.

Table 6.1: Sammenlignsresultater for ARIMA og LSTM

	JNJ		GOOG		EQNR		LB		CCL		I alt	
	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM
Antal forecasts	30	30	30	30	30	30	30	30	30	30	150	
Korrekt retning	15	19	12	12	14	14	12	8	8	17	61	70
I pct.	50%	63%	40%	40%	47%	47%	40%	27%	27%	57%	41%	47%
Theil's U index	0.020	0.007	0.020	0.011	0.026	0.013	0.045	0.041	0.037	0.018	0.030*	0.018*
RMSE	2.87	1.09	26.06	14.96	0.50	0.25	0.82	0.72	1.73	0.81	31.98	17.83
MAE	2.41	0.90	19.60	12.41	0.40	0.20	0.57	0.55	1.32	0.57	24.3	14.63
Persaran & Timmermann P-value	0.56	0.20	0.87	0.74	0.62	0.78	0.87	0.99	0.99	0.30		
Anatolyev & Gerko P-value	0.13	0.93	0.56	0.96	0.77	0.50	0.65	0.99	0.63	**0.04**		

* udregnet som et gennemsnit.

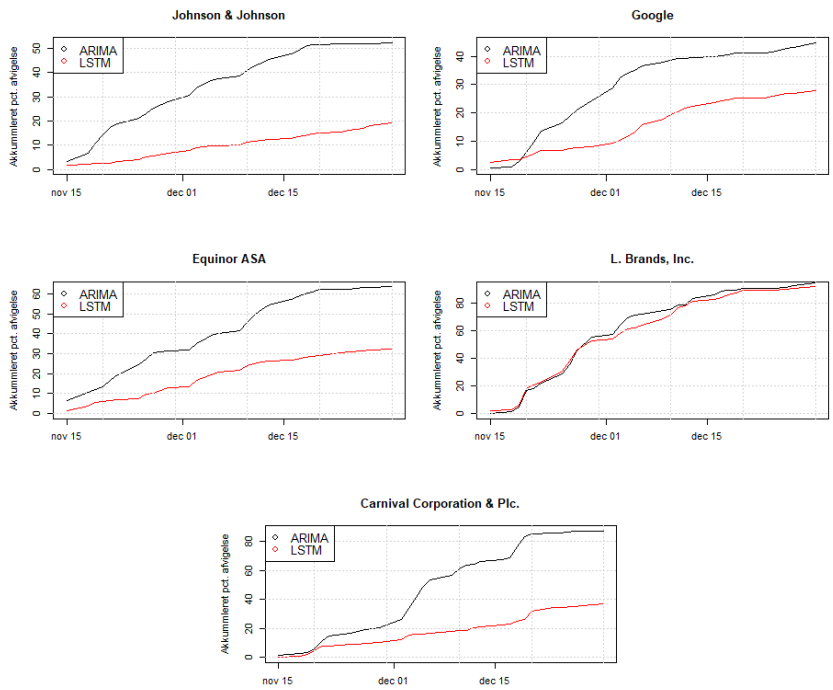
Table 6.2: Diebold-Mariano test

	JNJ LSTM	GOOG LSTM	EQNR LSTM	LB LSTM	CCL LSTM
M1=M2	>0.00	0.02	>0.00	0.22	0.01
M1<M2	1	0.99	0.99	0.89	0.99
M1>M2	>0.00	0.01	>0.00	0.11	>0.00

ARIMA-modellerne er benchmark-modellen (M1).

De akkumulerede fejlforcast opgjort i procent er ligeledes grafisk vist i figur 6.1 (bemærk at y-aksen ikke er den samme for alle 5 figurer). LSTM slår igen ARIMA på denne parameter med lavere værdier for alle aktier. LSTM klarer sig bedst ved JNJ med kun 19.2 procent i akkumuleret fejl, mens ARIMA forecaster forkert med 52.24 pct. LB giver de største akkumulerede fejl for de to model med 94.14 for ARIMA og 91.59 for LSTM. Samlet set vinder LSTM over ARIMA i denne parameter med 208.01 pct. mod 342.12 pct.

Figure 6.1: Akkumuleret fejlforcast i pct. mellem ARIMA og LSTM



En Pesaran & Timmermann test kan også benyttes til at finde signifikant gode forecasts. Testen kigger på hvorvidt om værdierne fra SR er signifikant forskellige fra hvad der kunne forventes, hvis fortegnene fra data og forecast var uafhængig fra hinanden. Dermed er nulhypotesen lig med en uafhængig fordeling mellem data og forecast, og vice versa med den alternative hypotese. Dermed kan specialet fastslå at ingen prædikteringer er så signifikant gode, at disse kan betragtes som afhængige fordelt og et bevis for stærke prædikteringsevner (Pesaran and Timmermann, 1992). LSTM's prædiktioner for JNJ var dog dem, der kom tættest på med en p-værdi for 0.2.

Anatolyev & Gerko test minder meget Pesaran & Timmermann test, men deres nulhypotesen tager her udgangspunkt i, at den betinget middelværdi i forecastet er uafhængig. Resultatet er ligeledes ej heller forskelligt fra Pesaran & Timmermann testen, dog med undtagelse af CCL prædikteret af LSTM som viser sig signifikant med P-værdi på 0.04, og således kan betragtes som et signifikant stærkt forecast (Anatolyev and Gerko, 2005).

Tabel 6.2 er den sidste udvalgte evalueringsmetode, som består af en Diebold-Mariano test som sætter de to forecasts op mod hinanden. Diebold-Mariano test mener ligeledes at LSTM er signifikant bedre end ARIMA, når det kommer til at forecaste JNJ, GOOG, EQNR og CCL, og derved er disse forecasts også signifikant forskellige fra hinanden. I de resterende tests kan nulhypoteserne ikke afvises, og dermed kan ingen ydereligere konklusioner drages ud fra denne test.

6.0.1 Opsumering

LSTM var således bedst til at prædiktere retning, og fange den volatilitet, som forekom i data med 47 pct. korrekte retninger, hvilket dog kun førte til en enkelt signifikant p-værdi i Anatolyev & Gerko test. MAE og RMSE pegede ligeledes i LSTM retning, og at denne model var bedre i sin prædiktering. Dette blev ligeledes grafisk vist i figur 6.1, som viste at LSTM akkumuleret set lavede færre fejl omregnet i procent. Disse blev ligeledes brugt i Diebold-Mariano test, som i fire ud af fem tilfælde, tilskrev LSTM som den signifikant bedste model.

Når man har denne snak, skal man holde sig for øje at lavere prædikteringsfejl ikke nødvendigvis er lig med flere korrekte prædikteringer. Specialet har således afvist modeller (både ARIMA og LSTM) som havde flere korrekte præditkeret retninger, men altså havde alt for høje fejlprædikteringer målt på RSME og MAE. Dermed handler dette om sammenligningsgrundlag, da man normalt i videnskabelige artikler går efter at reducere fejlprædikteringer, således at forecastet kommer så tæt på data som muligt. Dog når man vender blikket mod finansielt data eller rettere aktiepriser, må det siges at en korrekt prædikteret retning er vigtigere end lave fejl, når man investere. Således selvom LSTM lavede færre fejl i forcasting af LB end ARIMA, så var ARIMA bedre til prædiktere retning med 12 mod 8. Denne problemstilling mellem lave fejl ikke nødvendigvis er lig med korrekt retning vil der blevet taget hånd om i Mean-Variance analyse, som vil blive gennemgået i næste kapital.

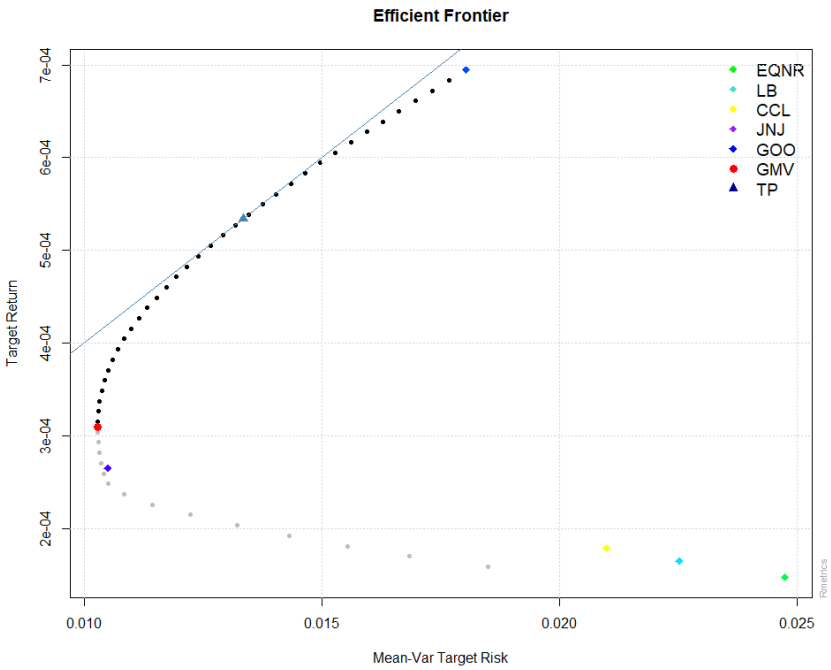
Porteføljestrategier 7

Et nærmere blik på modellernes performance kan vises i kontekst med aktieinvestering, således vil dette speciale ikke blot kigge på matematikken bag modellerne og sammenligne dem med de traditionelle metoder som bl.a. RMSE og MAE. Derimod vil specialet inddrage en mean-variance analyse for rent praktisk at teste om de beregnet forecastingsevnver blandt ARIMA og LSTM kommer til udtryk ift. aktievælgelse og afkast.

7.0.1 Mean-Varaince analyse

En mean-variance analyse er et kendt værktøj indenfor den finansielle verden til at porteføljesammensætte, hvor indragelse af gennemsnit fra tidligere afkast i procent bruges, og en kovarians-matrix mellem de udvalgte aktier udregnes ud fra tidligere afkast. Mean fungerer således som en indikation på hvad der kan forventes af afkast på aktien, mens Variance udtrykker den risiko, der følger aktien. Ud fra det global minimum varians (GMV) og tangencyporteføljen (TP) kan den efficiente rand udregnes. Dette er afbilledet i figuren nedenfor.

Figure 7.1: Mean-Variance analyse



I figur 7.1 er de fem aktier fra observation 1 til 3241 (observationerne før forecastingsperiode), afbilledet ud fra deres varians og middelværdi, som tilsammen danner den efficiente rand, som er alle yderpunkter

i en mulig porteføljesammensætning. Det kan hermed ses at GOOG og JNJ ligger højest opad y-aksen, da de ifølge figur 5.1 ligeledes har det højeste mean ift. de andre udvalgte aktier. Det kan overraske at både CCL, LB og EQNR vurderes som mest risikofyldte aktier på trods af at de alle har lavere varians end JNJ og GOOG ud fra tabel 5.1. Variansen i afkast i procent er dog langt højere og således vurderes disse som risikofyldt.

Randen udregnes som sagt ud fra GMV og TP igennem mean og kovariansmatrixen. Middelværdierne for afkastene fra de 5 aktier indsættes i en matrix[5,1] og krydses med den inverse kovariansmatrix[5,5] (i omvendt rækkefølge, jævnfør matrix-regneregler), som tilsammen giver vægtningerne for TP. I udregningen for GMV erstattes mean-matrix med et-taller og krydses igen med den inverse kovariansmatrixen. Således opnås to punkter i randen og derfor kan man udregne resten af punkterne. GMV er kendetegnet ved at være den porteføljesammensætningen med den mindste varians, som navnet angiver. Tangency-porteføljen er defineret som den portefølje med det højest mulige Sharpe-forhold. Sharpe-forhold(S) er mellem porteføljens afkast(R) minus den risikofrie rente(R_f) og porteføljens varians(σ):

$$S = \frac{E(R - R_f)}{\sigma} \tag{7.1}$$

Dermed ligger TP på den øvre halvdel af randen, og opnås når det marginale afkast ikke længere overstiger den marginale varians eller risikoen ved at gå efter mere afkast. Dette er således punktet eller porteføljesammensætningen, som dette speciale anvender ved udregning af tabel 7.1. Dermed spiller lave fejlprædikteringer også en vigtig rolle selvom den prædikteret retning er forkert. Da prædikteringen bliver omregnet til fremtidigt afkast og brugt i M-V analysen til at finde den rette vægt for hver aktie.

Det risikofrie afkast kan være en svær ting at effektuere, da alle har muligheden for at placere opsparingen helt uden risiko i banken med enten meget små eller ingen rentetilskrivninger. Investoren kan ligeledes benytte muligheden for at placere sig opsparing en kortløbende stats- eller realkreditobligation. Det er meget muligt, at afkastet er nul, hvis opsparingen skal placeres uden risiko – især i en tid, hvor renteniveauet er lavt – men ikke desto mindre indgår det risikofrie alternativ normalt i beregningen. Dette speciale har dog valgt at udenlade dette element[1].

7.0.2 Antagelser

Specialet har valgt at udregne 7 forskellige porteføljer med en 1 million. dollars i hver:

1. Den første porteføljestrategi er simpel, og vælger at placere en femtedel i hver aktie, og således kaldes denne portefølje blot Strategi 1.

Porteføljestrategi 2. og 3. er baseret på de samme beregninger, nemlig en mean-variance analyse ud fra data fra observation 1 til 3241. Dermed er forskellen mellem de to porteføljer, at den ene tillader shorting.

Porteføljestragi 4. og 5. kan ligeledes slås sammen, da disse er baseret på prædikteringerne fra ARIMA og LSTM for observation 3242 til 3271, som indsættes som mean for mean-variance analyse, og derefter

[1]Specialet har udført estimatorer med renten for den amerikanske t-bill med en løbetid på en måned, som den risikofrie rente. Den 14. nov. 2019, som er dagen inden prædiktering begynder, kunne en investor få en t-bill med 1,59 pct. i rente. Dog giver inddragelsen af t-bill rente ikke andre resultater for portefølje 1, 2 og 3, og for portefølje 4 til 7 ændres porteføljen dagligt og dermed skal en daglig t-bill rente udregnes, hvilket giver et meget lav estimator, som ligeledes ikke giver andre resultater for porteføljesammensætningen. Derfor udenlades inddragelsen af den risikofrie rente i dette speciales brug af mean-variance analyse.

sammensættes en portefølje. De to porteføljer ændrer sig således hver dag i de 30 handelsdag, modsat de tre første som fastholder samme vægt hver dag igennem hele perioden.
6. og 7. følger samme prædikteringer fra 4 og 5, men blot med muligheden for shorting.
1 million. dollars som startkapital, er blot et fiktiv tal, som specialet har taget udgangspunkt i, og dermed kunne dette tal erstattes med alle mulige andre og stadig give samme resultat. Dog med den bemærkning at mindre afvigelser kan forekomme, da specialet har afrundet aktiekøbene, således at en portefølje ikke kan bestå af f.eks. 51.57 aktier i GOOG, men kun 51, hvilket efterlader et restbeløb til næste periode. Specialet har også gjort sig følgende antagelser for at opnå nedenstående beregninger ift. porteføjlesammensætning:
- Der antages at kurtage omk. beløber sig til 29 USD hver gang, der købes og sælges.
- Det antages at det alle dage kan svare sig at ændre porteføljesammensætningen på trods af kurtages omk. Samtidig er der ingen dage, hvor det ikke kan svare sig, grundet at porteføljen kun kan bestå af 5 aktier og forholdet mellem kurtagesomk. (29 USD) og porteføljens startværdi (1 million. USD) er relativt lille. Dermed kunne det have større indflydelse på afkast, hvis forholdet var sat anderledes.
- Det antages at der kan sælges og købes til de præcise lukkepriser som der prædikteres efter.
- Ved amerikanske aktier skal danske investorer tage højde for valutakurserne der udvikler sig samtidig med ens aktiekurser og spiller derfor også en rolle i virkeligheden. Dette forhold har specialet dog valgt at holde konstant og således ikke komplicere denne del af analysen ydereligere.
- Vægtningerne for opnåelse af TP kan til tider være abnormale f.eks. ved dag 22 i ARIMA's prædiktering opnås TP med følgende vægtninger: 12621% i JNJ, -3148% i GOOG, 18758% i EQNR, -89060% i LB og 60928% i CCL. Disse vægtninger må vurderes som urealistiske og alt for dumdristige til at en investor ville foretage disse positioner. Samtidig giver disse vægtninger meget store udsving i porteføljerne, og derfor har specialet valgt vægtningerne som disse, vil blive omregnet således at porteføljerne maksimalt kan gå +/- 300% i hver aktie. Dermed kommer vægtningerne for dag 22 til at se ud som følgende: 56% i JNJ, -10% i GOOG, 83% i EQNR, -300% i LB og 271% i CCL. Hvis dette ikke gøres ville Buy & sell med ARIMA gå i minus med omkring 500.000 USD. fra dag 10 til dag 22 og til sidst ender med et samlet afkast for porteføljen på 628% på dag 30.

7.0.3 Resultater

Førnævnte antagelser har herved givet følgende resultater, som er opsummeret i tabel 7.1. De samtlige 7 porteføljer med deres sammensætninger og udvikling i forecastingsperioden er grafisk præsenteret i bilag A.1 til bilag A.7.

Table 7.1: Porteføjlresultater

	Strategi 1	M-V portfolio	M-V portfolio with Shorting	Buy & sell		Buy & sell with Shorting	
				ARIMA	LSTM	ARIMA	LSTM
Afkast	6.71%	5.26%	4.84%	9.69%	9.59%	-20.26%	28.03%
Std. afv.	26639	20852	22076	56624	28762	139055	79705
Maks. afkast	1077487	1064751	1081478	1108024	1103787	1117523	1287741
Mini. afkast	981615	1000000	1000000	937580	995503	608486	978528
Antal handler	10	4	10	123	119	155	155
(Køb & salg)	(5 - 5)	(2 - 2)	(5 - 5)	(59 - 64)	(53 - 66)	(75 - 80)	(77 - 78)
Trans.omk.	290	116	290	3567	3451	4495	4495
(% af afkast)	(0.0003%)	(0.0001%)	(0.0003%)	(0.003%)	(0.003%)	(0.006%)	(0.004%)
Indbringende køb				35	29	41	38
% af køb				(59.32%)	(54.72%)	(54.67%)	(49.35%)
Indbringende salg				29	22	36	32
(% af salg)				(45.31%)	(33.33%)	(45%)	(41.03%)
Samlet succesrate				64	51	77	70
(%)				(52.03%)	(42.86%)	(49.68%)	(45.16%)

Det kan aflæses i tabel 7.1 strategi 1 giver et afkast på 6,71 pct. på dag 30, hvor alle aktier i porteføljen bliver solgt ud. Alle aktier oplever positiv aktiefremgang i denne periode fra dag 1 til dag 30, og derved fungere strategi 1's porteføljesammensætning godt. Mean-variance porteføljen uden shorting ville give et afkast på 5.26 pct. og oplever samtidig lavere standard afvigelse grundet at strategien kun investerer i JNJ (37.45 pct.) og GOOG (62.55 pct.). Dette bevirker samtidig at den laveste værdi, som denne portefølje oplever, er startværiden på 1 million., mens strategi 1 bevidner at komme under startværdien på 1. million. Mean-variance porteføljen med shorting leverer ligeledes positiv afkast på 4.84 pct., og når i perioden faktisk en højere maksimum værdi end de føromtalte porteføljer. Mean-variance porteføljen med shorting går således lang i JNJ (63.26 pct.) og GOOG (80.77 pct.), og kort i ENQR (15.63 pct.), LB (9.53 pct.) og CCL (18.87 pct.). Disse tre omtalte porteføljer er ligledes kendetegnet ved lavere transaktionsomkostninger, da de foretager få handler og da kurtages kun er på 29 USD. pr. transaktion, og spiller dermed en meget lille rolle ift. afkast.

Buy & sell er ligesom M-V porteføljen uden shorting, da porteføljerne kun kan købe og sælge uden at kunne gå kort i positionerne. ARIMA kan ud fra sine prædikteringer leverer et afkast på 9.69 pct., mens LSTM giver 9.59 pct. og begge slår dermed de tidligere nævnte porteføljer. Standard afvigelsen og maksimunsværiden er ligeledes højere for ARIMA end for LSTM. Den høje standardvigelse kan være forårsaget af de flere handler som ARIMA fortager med 123 mod 119. Det giver naturligvis en anelse flere omkostninger til ARIMA, men rykker ikke ved den overordnede konklusion.
Et indbringende salg defineres som et salg af en aktie foretaget dagen før at aktiekursen falder. Dermed reagerer porteføljen efter hensigten ved enten at sælge ud eller gå kort i aktierne, som er prædikteret til at falde og ligeledes gør det. Dermed er et indbringende køb udtryk for et aktiekøb som efterfølgende

oplever stigning, hvilket man igennem mean-variance model køber ind i. I buy & sell kan det se at ARIMA faktisk foretager flere indbringende salg og køb end LSTM, og samlet set opnår en højere succesrate på 52.03 pct. mod 42.86 pct. Således forøges ARIMA's korrekte prædikteringsretninger med 10 pct., (fra tabel 6.1) ved brug af mean-variance analyse. På trods af færre indbringende salg og køb, så formoder LSTM at levere næsten samme afkast som ARIMA.

De dårlige prædikteringsevner hos ARIMA kan ses i Buy & sell med shorting, som leverer et negativ afkast på -20.26 pct. Dette på trods, leverer ARIMA igen en højere succesrate med 49.68 pct. mod LSTM's 45.16 pct. Dog bliver fejlprædikteringerne mere afgørende for porteføljen end de korrekte prædikteringer, som derved giver en høj standard afvigelse og den absolutte laveste miniumsumsværdi af alle porteføljer. Omvendt klarer LSTM sig langt bedre og giver et afkast på 28.03 pct., og når således den højeste maksimumsværdi af alle strategier. ARIMA har igen en højere succesrate end LSTM, og foretager flere indbringende salg både norminalt og procentvis.
Buy & sell med shorting kræver derved langt mere af prædikteringerne fra ARIMA og LSTM, hvilket kan bevidnes ved den store afkastsforskel mellem ARIMA og LSTM. Overordnet set klarede både ARIMA og LSTM bedre end de passive porteføljesammensætninger, og således på trods af lave korrekte prædikteringer så leverer både ARIMA og LSTM gode resultater med undtagelse af ARIMA med shorting. Næste kapitel vil tage et nærmere kig på resultater og diskutere underliggende faktor bag de opnåede resultater samt evt. forbedringer.

Diskussion 8

I overstående analyse er modellerne testet ud fra statistiske evalueringer og til brugen af porteføljesammensætning. Dele af resultaterne, fra analysen, var forventlige dog med enkelte overraskende elementer. Dette efterlader flere interessante spørgsmål i specialet, som i dette afsnit vil blive nærmere analyseret og diskuteret.

På stort set alle parametre opnåede LSTM bedre resultater end ARIMA. Den prædikterede i flere tilfælde korrekt og opnåede således lavere MAE og RSME, hvilket altsammen var gældende for porteføljesammensætningen ift. afkast. ARIMA klarede sig over middel, forstået på den måde at de passive porteføljesammensætninger ikke kunne slå ARIMA. Dog viste LSTM sig at være ARIMA overlegen målt på afkast, hvor shorting var tilladt.
Dette var således en test udført i en periode, som specialet har kendetegnet som værende under kursopgang (bull), og således kan man spørge sig selv, om samme resultater havde været opnået, hvis perioden havde været kendetegnet ved et bear-markedet. Dataet viser dog imidlertid kun 80 stigninger mod 70 fald, og dermed kan ARIMA synes at være mere passende til data. Dette er afdækket i tabel 8.1, hvor det i bunden kan aflæses, at LSTM faktisk hovedsageligt laver sin fejl (51 ud af 80) når kursen reelt falder, men hvor den prædikterede en stigning. Det kunne dog dokumenteres i tabel 6.1 at ARIMA prædikterer kun rigtig med 41 pct. mod LSTM 47 pct. og således var begge modeller værre end tilfældighed. ARIMAs fejl er mere ligeligt fordelt med 40 fejlprædikteringer ved fald og 49 ved stigning, mens LSTM prædikterer flere stigninger og dermed rammer mere ved siden af. Det kunne tyde på, at særligt LSTM's forecasting af JNJ kunne trænge til mere parameterturning, da LSTM prædikterer 26 stigninger mod 4 fald.

Table 8.1: Fejlprædikteringer

		Prædikteret stigninger	Prædikteret fald	Forkerte prædikteringer	Prædikteret stigning men reelt fald	Prædikteret fald men reelt stigning
JNJ	ARIMA	18	12	15	8	7
	LSTM	26	4	11	10	1
GOOG	ARIMA	14	16	18	9	9
	LSTM	20	10	18	12	6
EQNR	ARIMA	14	16	16	6	10
	LSTM	16	14	16	7	9
LB	ARIMA	15	15	18	10	8
	LSTM	21	9	22	15	7
CCL	ARIMA	11	19	22	7	14
	LSTM	20	10	13	7	6
I alt	ARIMA	72	78	89	40	49
	LSTM	103	47	80	51	29

Dataet for JNJ viser dog kun 17 stigninger og 13 fald, og således prædikterer LSTM forkert 13 gange. Samme mønster tegner sig for LB, hvor LSTM forecaster 21 stigninger mod 9 fald. Dataet for GOOG viser sig også særligt svært for både ARIMA og LSTM med henholdsivs 18 og 18 fejl hver. De to modeller

prædikterer ligeledes også meget forskelligt, da ARIMA forudsiger 9 fald mod LSTM med kun 6 fald, mens der i virkeligheden var 16. Derved var samme tidserier næsten lige svært for ARIMA og LSTM, men de to modeller lavede forskellige fejl ved samme data. LSTM er dermed mere tilbøjelig til at forecaste en stigning (103 ud af 150), og således kan der sættes spørgsmålstegn ved, om denne model reelt er bedre til at fange udsvingene i data, og er i stand til at tilpasse sig et modsat scenarie.

8.1 Data generering

Hvorvidt LSTM eller ARIMA er bedst til at fange udsving i data, kan anskues i variansen i data kontra prædiktioner i tabel 8.2. Det kan her aflæses, at variansen i LSTM's prædikteringer af JNJ, GOOG og CCL er tættest på dataets sande varians, mens vice versa er tilfældet for EQNR og LB. LB viser sig igen som en svær størrelse for modellerne at prædiktere, da begge modeller er relativt langt fra at have samme varians som data. GOOG har ligeledes en høj varians, som ARIMA fanger meget lidt af, mens LSTM er noget bedre med.

Table 8.2: Varians i data

	JNJ		GOOG		EQNR		LB		CCL	
Varians	14.43		492.99		0.275		0.496		7.03	
Forecast	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM	ARIMA	LSTM
Varians	6.08	16.17	74.50	354.73	0.295	0.231	0.152	0.051	5.32	6.07

Den mulige volatilitet, som skal rummes af modellerne, er blevet et særligt varmt emne under specialets udarbejdelsesperiode i form af Corona-krisen. Corona-krisen har i perioden skabt stor uro og derved megen volatilitet på de finansielle markeder grundet usikkerheden. Eksempelvis har S&P 500 indekset (hvor specialets aktier står listet) ikke oplevet større udsving i priser siden børskrakket 1929. Derudover har mange lande valgt enten at lukke helt eller delvist ned, hvilket ligeledes har ramt den generelle økonomi, der afspejles i aktiemarkederne. Skiftet fra bull til bear marked kunne uden tvivl have påvirket specialets modeller med et negativt fortegn. Større fluktueringer og nye mønstre i data ville have sat sine spor i modellerne og nedsat succesraten for korrekte prædikteringer. Dog ville konklusionerne være de samme, med LSTM som værende en smule bedre end ARIMA, da ARIMA i særlig grad ville have svært ved skiftende volatilitet. Dette kunne dog have været løst med GARCH-model, som ville kunne håndtere disse udsving bedre.

Overordnet må det siges, at LSTM er bedre end ARIMA, men at begge modeller ikke viser sig overlegne ift. aktieforecasting. ARIMA var en lettere modeltype at justere, og har et større teoristisk grundlag, som derved gør metoden og fremgangsmåden til opbygning nemmere at gå til. Det store teoritiske grundlag kan omvendt ligeledes være en udfordring, da tidsserierne ikke må lide af seriel korrelation og skal være stationære, før det videre arbejde med forecast kan begynde. LSTM er mere i opposition med et manglende teoritisk grundlag i arbejdet med forecasting. LSTM-modeller kan dermed blive en lang og tidskrævende jagt på optimering af modellen, hvilket dog i sidste ende viser sig at være en smule mere nøjagtig end ARIMA's prædikteringer. Dette er således også blevet et af fokuspunkterne for litteraturen vedrørende LSTM, hvilket har ført til udviklingen af LIME.

8.2 Local Interpretable Model-agnostic Explanations

I nyere tid er der kommet mere fokus på fortolkning af DL-modeller. Når det kommer til LSTM så kendes algoritmen bag outputtet, men realtidstilstanden af det udvalgte NN er skjult. Dermed er grunden til at der produceres et specifikt output ukendt. Det kræver derfor en stor tillid til modellen og hvorvidt der kan stoles på prædiktionen? For at åbne op for den sorte boks, der forekommer i DL modeller, blev Local Interpretable Model-agnostic Explanatons (LIME) modellen præsenteret i 2016. LIME er en forklaringsteknik, der er udviklet til at kunne forklare prædiktionen af enhver forudsigelse. LIME trænes ikke på en global surrogatmodel[1], men derimod lokale surrogatmodeller til at kunne forklare de individuelle forudsigelser. LIME ændrer en enkelt observation i datasættet af gangen, igennem finjustering af funktionsværdier og derefter observerer dets påvirkning af output. Dermed åbner LIME op for sammenhænge i dataet, ved at justere enkelte observationer en af gangen (Ribeiro et al., 2016). Dermed kunne LIME anvendes på specialets LSTM-modeller for at skabe et større overblik og dermed øge LSTM præcision i forecasting. Dog kunne andre muligheder også overejes ift. at øge nøjagtigheden hos modellerne.

8.3 Kombination af Machine Learning og Økonomi

Sammenspillet mellem økonomi og ML giver incitament til at inddrage NN i økonometri og kreere modeller der kan anvende fordele fra begge genre, til bl.a. forecasting. NN's håndtering af store mængde ustruktureret data kunne således bidrage til økonometriens mål om beskrivelse af den økonomiske virkelighed kvantitativt. Sammenspillet mellem de to kan være med til, at verificere økonomiske teorier eller afdække kausale sammenhænge, og derved udvikle kvantitative modeller til forudsigelse af den økonomiske aktivitet eller udførelse af konsekvensberegninger af økonomisk-politiske tiltag.
Det har fra start været specialets mål, at skabe en hybrid mellem ARIMA og LSTM, hvilket i sig selv lykkes. Resultaterne fra hybriden var dog af en kvalitet som må betegnes som mangelfuld, grundet Spyder (Python) ikke kunne beregne alle ønskede estimator. Det lykkes kun i enkelte tilfælde at få de rette estimator, dog var forecastingsfejlene (RMSE og MAE) så høje og derved meningsløse for specialet at præsentere. Denne hybrid mellem ARIMA og LSTM er, som sagt efter specialets overbevisning, ikke en del af nogen videnskabelig artikler. Derved er hybriden en model, som kræver viderearbejde og større teoretisk fundament for at kunne fungere.
Generelt er formålet med anvendelsen af økonometri at finde relationerne i data. Hvor formålet med ML er at opnå en nøjagtigt forudsigelse. Med en kombination af de to redskaber der bygger på en relativ ens filosofisk ide om at beskrive data genereringsproces, kan der muligvis opnås mere akkurate prædiktioner end hidtil (Tan et al., 2017). ML er ikke nødsaget sig til at beskæftige sig med økonomisk data. Billeder er noget der bl.a. anvendes når der skal laves økonomiske analyser med AI. I en artikel "Measuring economic growth from outer space" skrevet af tre økonomer, anvendte forfatterne billeder til at forudsige økonomisk aktivitet og BNP vækst. Her blev satellit fotos anvendt til at spotte udledt lys, hvor intensiteten af lysstyrken blev analyseret i syv forskellige lag og holdt op mod densiteten af

[1] En surrogatmodel er en teknisk metode, der bruges, når et resultat af interesse ikke let kan måles direkte, så i stedet bruges en model af resultatet

den indkomstbesiddende befolkning. Områder hvor densiteten var lav, forekom det at de oftere have færre oplyste pixels, end områder med høj densitet. Dermed fandt forfatterne ud af at områder med høj densitet og lav intensitet oplever lavere aktivitet og vækst end område med høj densitet og høj intensitet (Henderson et al., 2011).
Billeder er ikke det eneste element der kan anvendes til forecasting. I en artikel "Text-baed crude oil price forecasting: A deep learning approach" anvendes metoden Natural Language Processing (NLP) der er en underkategori til AI. NLP er en metode, der håndterer interaktionen mellem computere og mennesker ved hjælp af det naturlige sprog. Formålet for NLP er at læse, afkode og forstå sprog så det bliver værdifuldt. I artiklen blev NLP anvendt sammen med text mining til at opsamle overskrifter fra diverse nyhedsmedier, der kunne have indflydelse på olie priser, som bl.a. politik og naturkatastrofer. Det forestærkede modellens evne til at opfange fluktueringer i olie prisen. Specialets empiriske resultater antyder, at LSTM-modellerne ville performere bedre ved tilføjelse af text mining, (Li et al., 2019).
Dermed kunne en videre udvikling af LSTM og ARIMA være en tilføjelse af billeder eller tekst vedrørende nyheder på aktiemarkedet.

8.4 Kan data fortælle alt?

Tilføjelse af flere forklarende variabler vil altid kunne øge nøjagtigheden i modellerne, dog med dette speciale for øje kunne det generelt tyde på, at både LSTM og ARIMA alene ikke helt overlegent kan forudsige aktiepriser, da modeller prædikterede under 50 pct. Aktiepriserne er således for tilfældige og kunne syntes ikke at være afhængig af fortiden. Dog kunne specialet afvise igennem tests med ARIMA at de 5 tidserier bedst kunne forklares ud fra en random walk. Dermed kan det synes svært at forbedre ARIMA-resultater mere, dog kunne man prøve flere forskellige model end gjort i bilag A.1.
Forbedring af LSTM's resultater er mere åben, da mere hyperparametertuning selvfølgelig fører til færre fejl i prædiktering. Når dette er sagt, kommer dette også til en pris af mere computertid, og således kan jagten efter marginalt bedre resultater blive lang. Samtidig arbejder specialet med et dagligt tidsperspektiv, som i virkelighedens verden, ville sætte en tidsmæssig begrænsning for hvor lang tid, der kan anvendes efter jagten på den perfekt LSTM-model, når en investor skal day-trade. Selvfølgelig kan den rette model efterfølgende bruges flere dage i træk, men over tid ville data eller adfærden på aktiemarkedet måske ændre sig og kræve yderligere hyperparametertuning.
Da dette speciale er datadrevet og søger at afdække to modellers evner, er det også nødvendigt at gå tilbage og reflektere over brugen af data på det finansielle felt. ML har som sagt visse love og tommelfingerregler, og derved en vis forudsigelighed som gør LSTM anvendelig på dette område. Det giver derfor mening at jagte ekstra nøjagtighed i ens prædikteringer igennem hyperparametertuning. Forudsigeligheden er dog lidt mere diffus i økonomiens og økonometriens verden. Modelleringen af rationelle agenter (investorer) som agere på et aktiemarked, medbringer nogle naturlige adfærdsmønstre, der er afhængige af personens individuelle nyttefunktion og ikke nødvendigvis konstant over tid. Brug af data til at forecaste er således noget lig med påstanden om, at tidligere adfærd er lig med fremtidig adfærd; ex-post er lig med ex-ante. Et keynesisk argument ville være, at dette ikke nødvendigvis er sandt, da ingen situationer nogensinde er nøjagtig den samme. Erfaringen indsamlet fra en given situation ville

have indflydelse på den samme agents tankeproces. Hvis en fremtidig identisk situation ville opstå, ville tidligere erfaringer fra investorer gøre situationen anderledes og derved give et andet udfald i kurserne. LSTM er ingen undtagelse fra denne antagelse om lighed, da et overfittende scenarie trækker paralleller til denne idé. NN kan kun lære mønstre, det har set før. Hvis NN lærer tidligere mønstrene for godt, begynder forudsigelsesevnen at blive dårligere. Selvom forudsigelser er nyttige til at reducere usikkerhed, bør resultaterne af en hvilken som helst model stadig betragtes som en retningslinje, uanset mængden af "intelligens", der imiteres. Især hvis der ikke præsenteres nogen indsigt i begrundelsen bag forudsigelsen.

Konklusion 9

Dette speciale havde sat selv til opgave at besvare følgende problemformulering:

Kan forecasting performance af aktiepriser i udvalgte lineære og ikke-lineære modeller jævnføres i forskellige aktive porteføljestrategier?

Specialet har mere præcist kigget på ARIMA (traditionel lineær model) og LSTM (ikke-lineær model). De to modeller er således langt fra ens, da ARIMA har stærke rødder inden for økonometriens verden, mens LSTM er afstedkommet fra Machine Learing og kunstig intelligens. Samtidig har ARIMA en lang fortid med et stærkt teoritisk fundament om korrelationer og parameterudregning, mens LSTM er en nyere opfindelse til håndtering af store mængde af data og mønstergenkendelse. Specialet er sat op om aktieforecasting, der har været notorisk svært at prædiktere. En del af litteraturen på området mener, at aktier er så uafhængig af fortiden og derved komplet tilfældige, og dermed stærkt præget af nutidens nyhed og trends, at prædiktering ikke giver mening.
Det var dermed ingen let opgave for de to modeller på trods af klare fordele ved begge modeller. Det blev klargjort, at LSTM i særlig grad har vist sig stærk i læring af mønstre i billeder og tekst. LSTM består af tre gates og en cell state (Constant Error Carousel (CEC)), som gør det muligt for forudsigelsesfejl at progageres tilbage i netværket, uden at "forsvinde" eller "eksplodere", men forblive som de er. Dette er således grunden til, at LSTM-modeller har haft stor succes med bl.a. at lære langsigtet afhængigheder ift. den tidligere anvendte RNN. ARIMA-modellen tager, modsat LSTM, højde for det faktum, at data over tid, har en intern struktur som bl.a. autokorrelation, trends eller er præget af sæson. Det er således ikke en ting i LSTM's verden. ARIMA er derfor nødt til at sikre stationæritet og ingen seriel korrelation i data før en forecasting kan foretages.
Det udvalgte data til forecasting, var fem tilfældige amerikanske aktier inden for hver sin sektor (JNJ, GOOG, EQNR, LB og CCL). Den passende ARIMA model blev fundet ud fra den laveste AIC værdi efter data blevet gjort stationært og fremkom uden seriel korrelation. Modelopbygningen til LSTM for de fem aktier følger på samme måde ikke en direkte metode, men rettere nogle tommelfingelregler, hvilket førte til hyperparametertuning af 10 forskellige modeller og test for overfitting, før de rette modeller blev fundet til forecasting af hver aktie.
Resultaterne fra forecasting var relativt ensidet med LSTM, der samlet set havde flest korrekte prædikteringer ift. retning, samt lavere Theils U indeks-, MAE- og RMSE-værdi, dog med enkelte undtagelser for de individuelle aktier. Specialet opnåede mindre reduktioner i RMSE og MAE (39.79 pct. og 44.25 pct.) ved brug af LSTM fremfor ARIMA, men ikke samme størrelsesorden som tidligere udført litteratur på området ((Namin and Namin, 2018) og (Namin et al., 2019)). Diebold-Mariano

test kunne ligeledes se signifikant forskel mellem de to modeller i 4 ud af de 5 aktie prædikteringer. Prædikteringerne for EQNR var således den eneste, hvor LSTM kunne fastslås signifikant bedre end ARIMA.

På trods lave succesrater ift. korrekt prædikteret retning (41 pct. og 47 pct.), viste begge modeller at kunne slå passive porteføljer ved brug af Mean-variance analyse. ARIMA kunne levere afkast på 9.69 pct. mod LSTM 9.59 pct. uden brug af shorting, mens LSTM skabte mere afkast end ARIMA med 28.03 pct. mod -20.26 pct. ved brugen af shorting.

De høje afkast for porteføljer ledte specialet på sporet af hvorvidt samme billede ville tegne sig, hvis forecastingsperioden havde været mindre gunstig med flere kursfald inden stigninger. Herfra kunne det dokumenteres, at LSTM faktisk var bedre til at tilpasse sig data med den passende varians i prædiktering ift. dataet, samt at LSTM prædikterede flere stigninger end fald generelt set. Igen kunne enkelte undtagelser nævnes med f.eks. JNJ, hvor LSTM prædikterede 26 stigninger mod kun 19 stigninger i virkeligheden. Der er således en del teoritisk fundament, der mangler bag LSTM før denne modeltype til fulde kan anvendes i sammenhænge med finansielt data. Der skal således en bedre forståelse af LSTM og dens sorte boks, hvilket en større introduktion og praktisereing af LIME kunne afhjælpe med i fremtiden. ARIMA er således målt på denne parameter langt mere brugervenlig og struktureret, mens anvendelsen af LSTM kan opnå mere nøjagtighed i forecasting og derfor må siges at være en bedre model til forecasting af finansielle aktiver.

Dog kan denne sammenligning af modeller trækkes op på større perspektiv af hvorvidt aktieforecasting er muligt med konstante ændringer i adfærd blandt investorer og andre aktør på markedet. Dermed stilles der fra specialets side spørgsmålstegn ved forecasting af aktier kun baseret på aktiekurserne er en realistisk metode og om der skal inddrages andre faktorer, når det kommer at vurdere fremtidige aktier i lineære og ikke-lineære modeller.

Bibliography

Amidi, A. and Amidi, S. (2019), *Recurrent Neural Networks cheatsheet*, Stanford.
https://stanford.io/2Plv7Xc, besøgt d. 24-02-2020.

Anatolyev, S. and Gerko, A. (2005), *A trading approach to testing for predictability*, Journal of Business Economic Statistics.
https://bit.ly/3adhZuT, besøgt d. 16-04-2020.

Ayodele, T. O. (2010), *Types of Machine Learning Algorithms*, University of Portsmouth. S. 19-48.

Bao, T. Q., Tan, L. N., An, L. T. T. and My, B. T. T. (2019), *Forecasting stock index based on hybrid artificial neural network models*, Science Technology Development Journal – Economics - Law and Management.
https://bit.ly/3al33uJ, besøgt d. 21-04-2020.

Bliemel, F. (1973), *Theil's Forecast Accuracy Coefficient: A Clarification*, Journal of Marketing Research, Vol. 10, No. 4. S. 444-446.

Brownlee, J. (2017), *Gentle Introduction to the Adam Optimization Algorithm for Deep Learning*, MachineLeaningMastery.
https://bit.ly/35NsV1X, besøgt d. 11-05-2020.

Brownlee, J. (2018), *How to configure the number of layers and nodes in a neural network*, Machinelearningmastery.
https://bit.ly/37Gmixt, besøgt d. 20-02-2020.

Childs, M. (2011), *John McCarthy: Computer scientist known as the father of AI.*
https://bit.ly/38vdI62, besøgt d. 17-02-2020.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., Bahdanau, D. and Bengio, Y. (2014), *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.*
https://bit.ly/2PIp9Qv, besøgt d. 03-03-2020.

Copeland, M. (2016), *What's the Difference Between Artificial Intelligence, Machine Learning and Deep Learning?*, Nvidia.
https://bit.ly/2uPMqZL, besøgt d. 19-02-2020.

Danish, I. (2019), *Introduction to Deep Learning*, Medium.
https://bit.ly/37Gmixt, besøgt d. 20-02-2020.

Dechter, R. (1986), *Learning While Searching in Constraint-Satisfaction-Problems*, Artificial Intelligence Center. S. 178-183.

Deorukhkar, O. S., Lokhande, S. H., Nayak, V. R. and Chougule, A. A. (2019), *Stock Price Prediction using combination of LSTM Neural Networks, ARIMA and Sentiment Analysis*, International Research Journal of Engineering and Technology.
`https://bit.ly/2KiACTI`, besøgt d. 21-04-2020.

Det Etiske Råd (2007), *Kunstig intelligens.*
`https://bit.ly/2HEHJV7`, besøgt d. 19-02-2020.

Dreyfus, S. (1973), *The computational solution of optimal control problems with time lag*, IEEE Transactions on Automatic Control. S. 383-385.

Elman, J. L. (1990), *Finding Structure in Time*, Vol. 14, Cognitive science.
`https://bit.ly/2HMiNvb`, besøgt d. 24-02-2020.

Enders, W. (2014), *Applied Econometrics Time Series*, Wiley.

Expert System Team (2017), *What is Machine Learning? A definition*, Expert System.
`https://bit.ly/2M4EMzC`, besøgt d. 18-04-2020.

Fadlalla, A. and Lin, C.-H. (2001), *An Analysis of the Applications of Neural Networks in Finance*, Informs. S. 112-122.

Gers, F. A. and Schmidhuber, J. (2000), *Recurrent Nets that Time and Count*, IDSIA.
`https://bit.ly/2x7bITR`, besøgt d. 03-03-2020.

Gers, F. A., Schmidhuber, J. and Cummins, F. (1999), *Learning to Forget: Continual Prediction with LSTM*, Proc. ICANN'99 Int. Conf. on Artificial Neural Networks (Edinburgh, Scotland).
`https://bit.ly/38iDr0H`, besøgt d. 03-03-2020.

Grover, P. (2018), *5 Regression Loss Functions All Machine Learners Should Know*, Heartbeat.

Hagan, M. T., Demuth, H. B., Beale, M. H. and Jesús, O. D. (2014), *Neural network design*, 2nd edn.

Hall, J. S. (2011), *Algorithmic probability and friends - bayesian prediction and artificial intelligence*, Springer. S. 174.

Hayes, A. (2019), *Econometrics Definition*, Investopedia.
`https://bit.ly/2LZZvEA`, besøgt d. 18-04-2020.

Henderson, J. V., Storeygard, A. and Weil, D. N. (2011), *Measuring Economic Growth from Outer Space*, NBER.

Hochreiter, S. and Schmidhuber, J. (1997), *Long short-term memory*, Neural Computation. S. 1735-1780.

Hsieh, D. A. (1991), *Chaos and Nonlinear Dynamics: Application to Financial Markets*, The Journal of Finance. S. 1839-1877.

IDC (2019), *Worldwide Spending on Artificial Intelligence Systems.*
`https://bit.ly/2HATJqC`, besøgt d. 19-02-2020.

Ivakhnenko, A. (1968), *The group method of data handling—a rival of the method of stochastic approximation*, Soviet Automatic Control. S. 43-55.

Ivakhnenko, A. and Lapa, V. G. (1965), *Cybernetic predicting devices*, CCM Information Corporation.

Ivakhnenko, A., Lapa, V. G. and McDonough, R. N. (1967), *Cybernetic predicting devices*, American Elsevier.

Johnson, J. and Whinston, A. (1998), *Advances in Artificial Intelligence in Economics, Finance, Management*, Jai Pr. S. 84-86.

Kelleher, J. D. (2019), *Deep learing*, The MIT press.
`https://bit.ly/2w05sg9`, besøgt d. 24-02-2020.

Kim, H. Y. and Won, C. H. (2018), *Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models*, Elsevier.
`https://bit.ly/2wZwhls`, besøgt d. 21-04-2020.

Kolb, D. P. (2019), *In Cloud We Trust.*
`https://bit.ly/39QN8o3`, besøgt d. 19-02-2020.

Li, Z., Shang, W. and Wang, S. (2019), *Text-based crude oil price forecasting: A deep learning approach*, ELSEVIER.

Linnainmaa, S. (1970), *The representation of the cumulative rounding error of an algorithm as a Taylor expansion of the local rounding errors*, University of Helsinki (Master's thesis).

Lovric, M. (2011), *International Encyclopedia of Statistical Science*, Springer Berlin Heidelberg. S. 701.

Malkiel, B. G. (2003), *The Efficient Market Hypothesis and Its Critics*, Journal of Economic Perspective. S. 59-82.

McCarthy, J. (1959), *Programs with common sense*, Her Majesty's Stationary Office. S. 75-91.

McCarthy, J. (1980), *Circumscription: A form of non-monotonic reasoning*, Artificial Intelligence. S. 27–39.

McCarthy, J. and Hayes, P. J. (1969), *Some philosophical problems from the standpoint of artificial intelligence*, Edinburgh: Edinburgh University Press. S. 463–502.

McCulloch, W. S. and Pitts, W. (1943), *A logical calculus of the ideas immanent in nervous activity*, Bulletin of mathematical biophysics.
`https://bit.ly/3bVXaq8`, besøgt d. 21-02-2020.

McDermott, D. and Doyle, J. (1980), *Non-monotonic logic I*, Artificial Intelligence. S. 41–72.

Minsky, M. (1974), *A framework for representing knowledge*, Artificial Intelligence Laboratory. S. 69-72.

Mitchell, T. M. (2006), *The Discipline of Machine Learning*, Carnegie Mellon University. S. 3.

Molnar, C. (2019), *Interpretable Machine Learning, A Guide for Making Black Box Models Explainable*, Machinelearningmastery.
https://bit.ly/37NunAt, besøgt d. 20-02-2020.

Moolayil, J. (2019), *Learn Keras for deep neural Networks - A fast-track approach to modern deep learning with python*, Apress. S. 142-59.

Mueller, J. P. and Massaron, L. (2019), *Deep learing for Dummies*, John Wiley Sons.
https://bit.ly/2HVbXn7, besøgt d. 25-02-2020.

Mullainathan, S. and Spiess, J. (2017), *Machine Learning: An Applied Econometric Approach*, Journal of Economic Perspectives.
https://bit.ly/3bUOamI, besøgt d. 21-02-2020.

Namin, S. S. and Namin, A. S. (2018), *Forecasting economic and financial time series: ARIMA vs. LSTM.*
https://bit.ly/34R2xUl, besøgt d. 21-04-2020.

Namin, S. S., Tavakoli, N. and Namin, A. S. (2019), *A Comparative Analysis of Forecasting Financial Time Series Using ARIMA, LSTM, and BiLSTM.*
https://bit.ly/2xCQpKB, besøgt d. 21-04-2020.

Pesaran, M. H. and Timmermann, A. (1992), *A simple nonparametric test of predictive performance*, Journal of Business and Economic Statistics 10.
https://bit.ly/3f0H7bY, besøgt d. 30-04-2020, S. 461-465.

Pyle, D. and José, C. S. (2015), *An executive's guide to machine learning*, McKinsey Co.
https://mck.co/38K8G5Y, besøgt d. 19-02-2020.

Reiter, R. (1980), *A logic for default reasoning*, Artificial Intelligence. S. 81–132.

Ribeiro, M. T., Singh, S. and Guestrin, C. (2016), *"Why Should I Trust You?": Explaining the Predictions of Any Classifier.*

Rumelhart, D. E., Hinton, G. E. and Williams, R. J. (1986), *Learning Internal Representations by Error Propagation*, Vol. 1, MIT Press. S. 318-362.

Russell, S. and Norvig, P. (2003), *Artificial Intelligence - A Modern Approach*, Pearson Education. S. 16-18.

Saiful, I. (2020), *Gradient Descent Stuck at Local Minima*, Researchgate.
https://bit.ly/2ZkzlUV, besøgt d. 20-05-2020.

Schmidhuber, J. (2015), *Deep learning in neural networks: An overview*, Elsevier.
https://bit.ly/2VluCR5, besøgt d. 21-02-2020.

Shumway, R. H. and Stoffer, D. S. (2011), *Time Series Analysis and Its Applications: With R Examples*, Springer. S. 83-154.

Stock, J. H. and Watson, M. W. (2007), *Introduction to Econometrics, 3rd Edition*, Pearson. S. 544-545.

Tan, Y., Zheng, E., Goes, P., Chellappa, R., Wu, D., Shaw, M., Sheng, O. and Gupta, A. (2017), *When Econometrics Meets Machine Learning*, De Gruyter.

Wang, P. (2019), *Predicting vs. Explaining*, towardsdatascience. https://bit.ly/3brjV4C, besøgt d. 20-04-2020.

Yao, K., Cohn, T., Vylomova, K., Duh, K. and Dyer, C. (2015), *Depth-Gated Recurrent Neural Networks.* `https://bit.ly/3cmSUAd`, besøgt d. 03-03-2020.

Zhang, J., Zhu, Y., Zhang, X. and Ye, M. (2018), *Developing a Long Short-Term Memory (LSTM) based Model for Predicting Water Table Depth in Agricultural Areas*, Journal of Hydrology. `https://bit.ly/2VppE5w`, besøgt d. 27-02-2020.

Zhang, S. Z. (2002), *Data Preparation for Data Mining*, Applied Artificial Intelligence. S. 375-381.

Bilag A

Table A.1: Modeludvælgelse ud fra AIC for de udvalgte aktier

	JNJ	GOOG	EQNR	LB	CCL
ARIMA(0,1,0)	9027.463	23918.73	4717.219	8628.515	8628.515
ARIMA(0,1,0) with drift	9028.108	23916.91	4719.168	8630.462	8630.462
ARIMA(0,1,1)	9025.463	23920.49	4709.369	8629.858	8629.858
ARIMA(0,1,1) with drift	9025.999	23918.74	4711.312	8631.806	8631.806
ARIMA(0,1,2)	9017.897	23910.47	4709.385	8631.856	8631.856
ARIMA(0,1,2) with drift	9018.264	23908.19	4711.325	8633.805	8633.805
ARIMA(0,1,3)	9010.046	23912.44	4711.369	8632.779	8632.779
ARIMA(0,1,3) with drift	9010.587	23910.19	4713.309	8634.725	8634.725
ARIMA(0,1,4)	9005.959	23913.57	4712.703	8634.685	8634.685
ARIMA(0,1,4) with drift	9006.351	23911.13	4714.644	8636.631	8636.631
ARIMA(0,1,5)	9007.827	23909.04	4714.064	8636.604	8636.604
ARIMA(0,1,5) with drift	9008.24	23906.09	4716.004	8638.55	8638.55
ARIMA(1,1,0)	9026.166	23920.87	4706.89	**8624.924**	8624.924
ARIMA(1,1,0) with drift	9026.751	23919.17	4708.848	8626.89	8626.89
ARIMA(1,1,1)	9025.53	23917.23	**4706.68**	8626.924	**8626.924**
ARIMA(1,1,1) with drift	9026.001	23915.46	4708.643	8628.89	8628.89
ARIMA(1,1,2)	9008.639	23912.84	4708.276	8628.924	8628.924
ARIMA(1,1,2) with drift	9009.071	23910.62	4710.246	8630.89	8630.89
ARIMA(1,1,3)	9008.091	23914.8	4710.286	8629.847	8629.847
ARIMA(1,1,3) with drift	9008.622	23912.61	4712.256	8631.815	8631.815
ARIMA(1,1,4)	9008.09	23907.11	4711.702	8631.778	8631.778
ARIMA(1,1,4) with drift	9008.532	**23903.44**	4713.659	8633.743	8633.743
ARIMA(2,1,0)	9018.543	23912.16	4707.242	8627.925	8627.925
ARIMA(2,1,0) with drift	9018.929	23909.99	4709.2	8629.891	8629.891
ARIMA(2,1,1)	9007.976	23913.8	4709.226	8629.925	8629.925
ARIMA(2,1,1) with drift	9008.428	23911.71	4711.183	8631.891	8631.891
ARIMA(2,1,2)	**9003.556**	23907.68	4711.133	8630.961	8630.961
ARIMA(2,1,2) with drift	9004.068	23904.03	4713.069	8633.823	8633.823
ARIMA(2,1,3)	9005.184	23909.49	4712.121	8632.078	8632.078
ARIMA(2,1,3) with drift	9005.771	23905.85	4714.071	8634.128	8634.128
ARIMA(3,1,0)	9012.245	23914.85	4709.865	8629.502	8629.502
ARIMA(3,1,0) with drift	9012.793	23912.75	4711.819	8631.47	8631.47
ARIMA(3,1,1)	9008.321	23916.85	4709.06	8631.503	8631.503
ARIMA(3,1,1) with drift	9008.84	23914.74	4713.746	8633.471	8633.471
ARIMA(3,1,2)	9006.604	23909.99	4710.372	8632.966	8632.966
ARIMA(3,1,2) with drift	9007.788	23906.16	4712.635	8634.957	8634.957
ARIMA(4,1,0)	9007.245	23917.25	4710.433	8632.268	8632.268
ARIMA(4,1,0) with drift	9007.621	23915.03	4712.398	8634.233	8634.233
ARIMA(4,1,1)	9008.776	23908.82	4712.231	8634.268	8634.268
ARIMA(4,1,1) with drift	9009.178	23904.97	4714.218	8636.233	8636.233
ARIMA(5,1,0)	9009.49	23912.05	4712.092	8635.083	8635.083
ARIMA(5,1,0) with drift	9009.911	23909.39	4714.061	8637.05	8637.05

Udarbejdet i R-studio.
Laveste AIC markeret med fed.

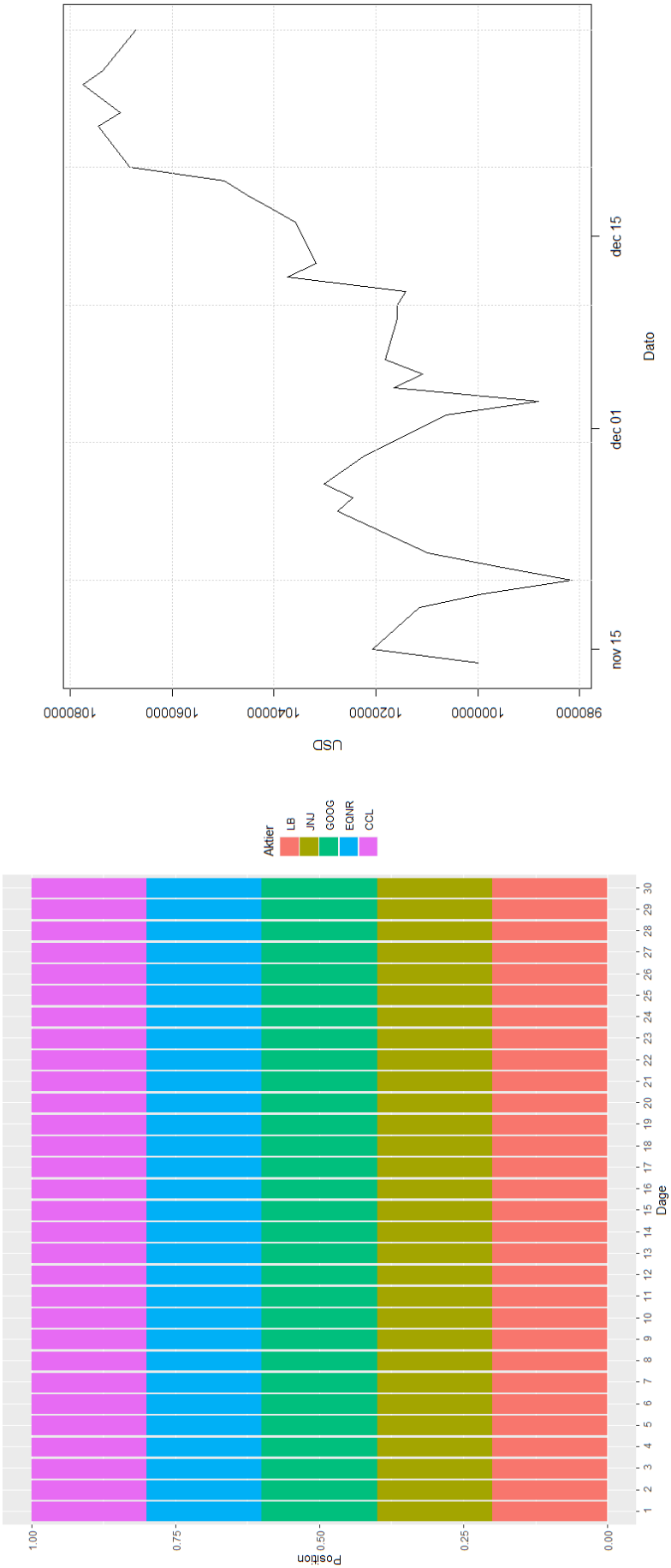


Figure A.1: Porteføljesammensætning og udvikling for strategi 1

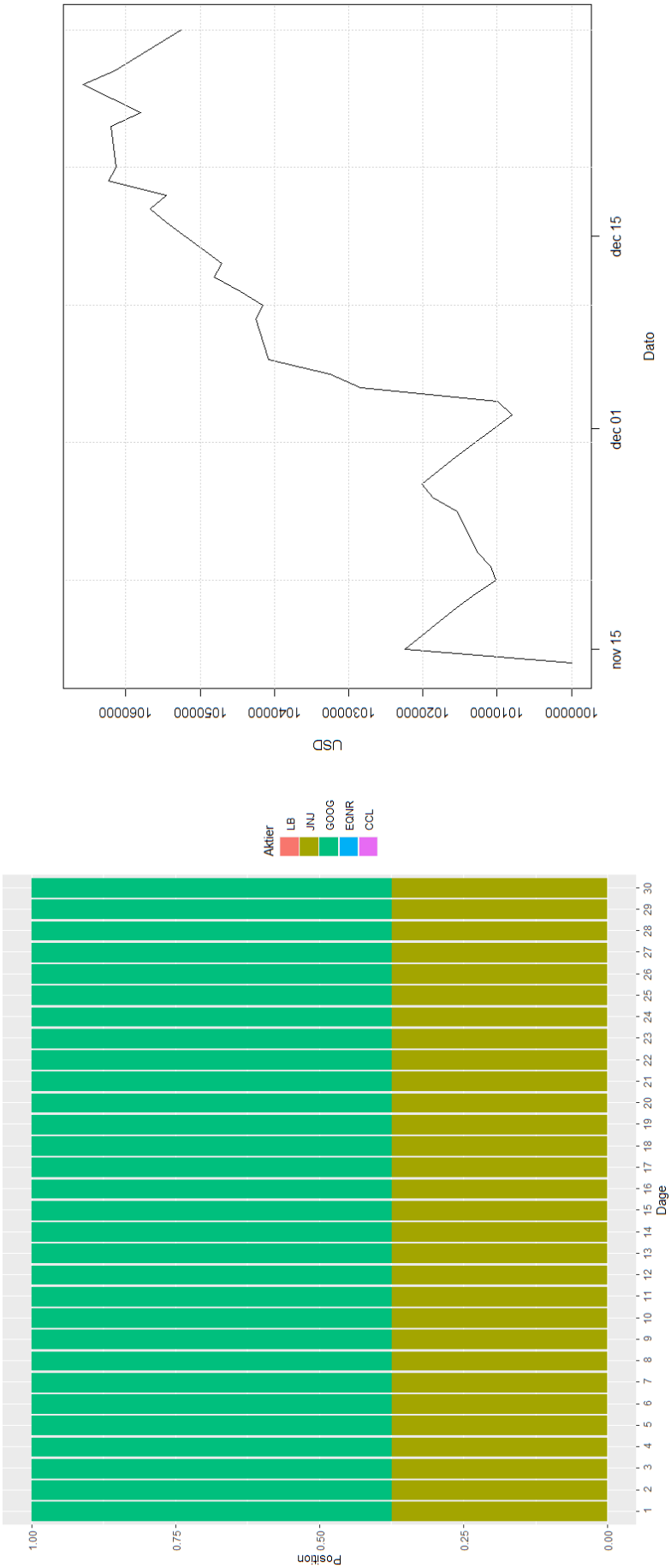


Figure A.2: Porteføljesammensætning og udvikling for MV-analyse uden shorting

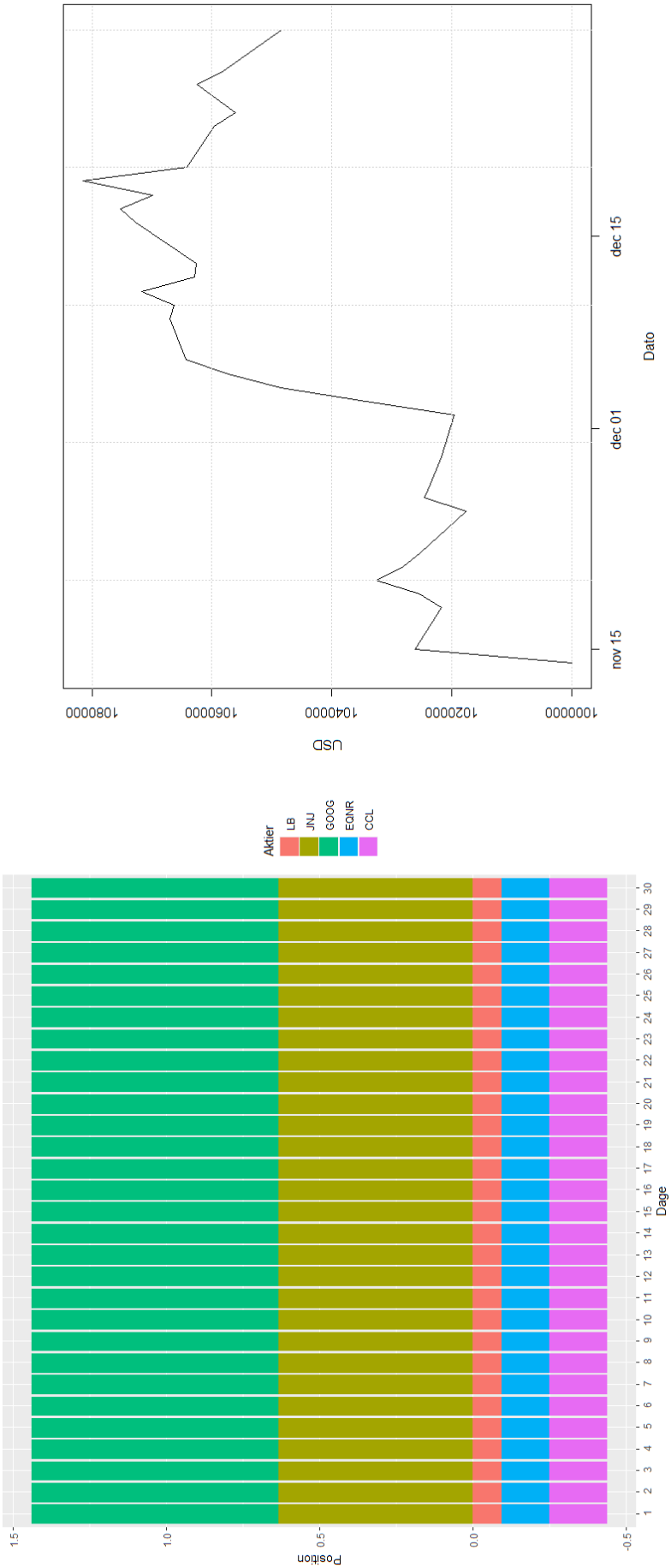


Figure A.3: Porteføljesammensætning og udvikling for MV-analyse med shorting

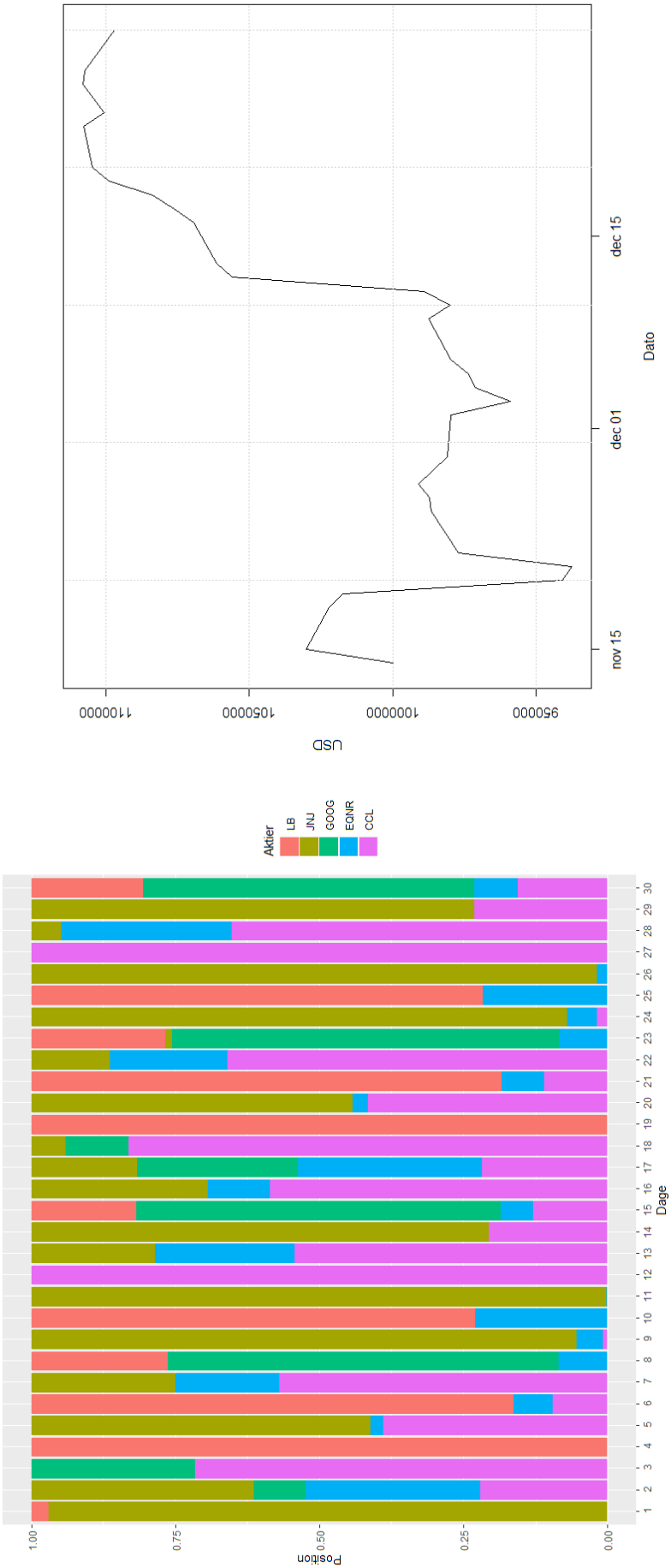


Figure A.4: Porteføljesammensætning og udvikling for Buy & sell med ARIMA-forecast

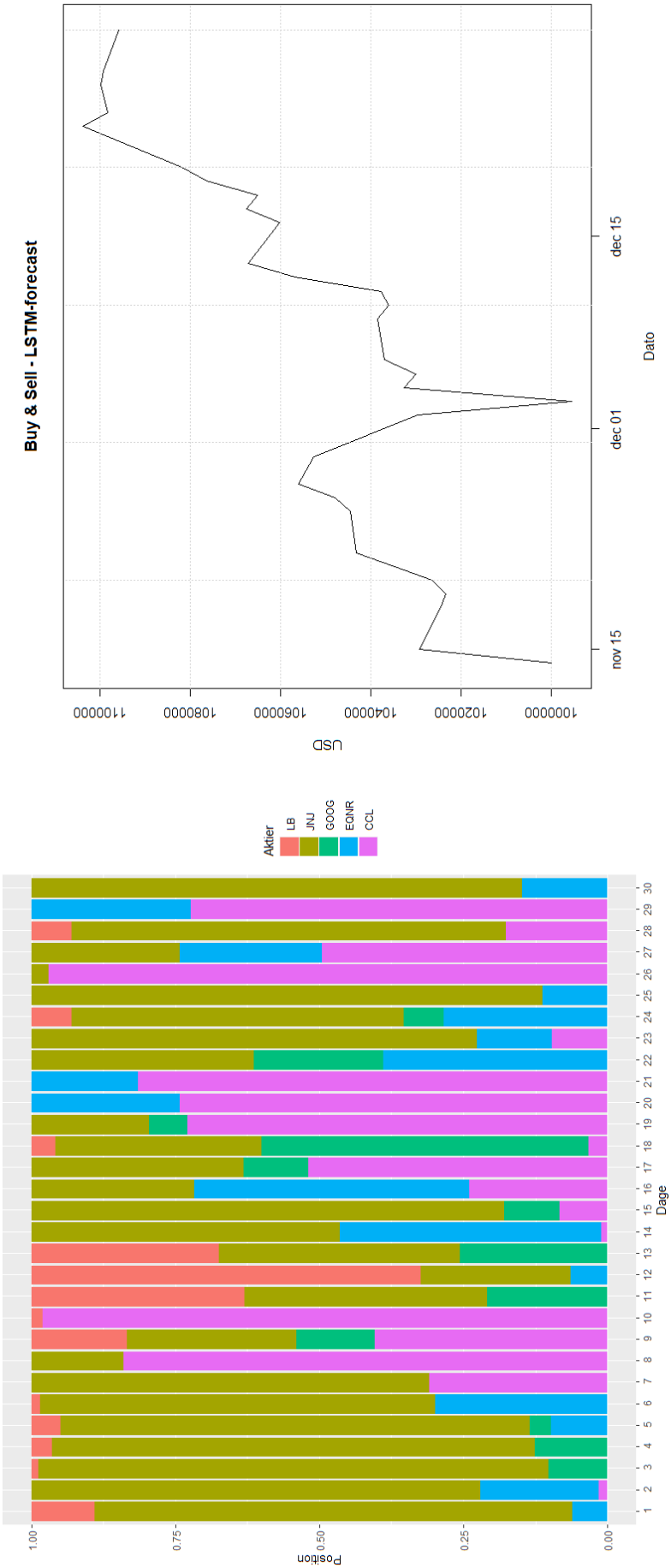


Figure A.5: Porteføljesammensætning og udvikling for Buy & sell med LSTM-forecast

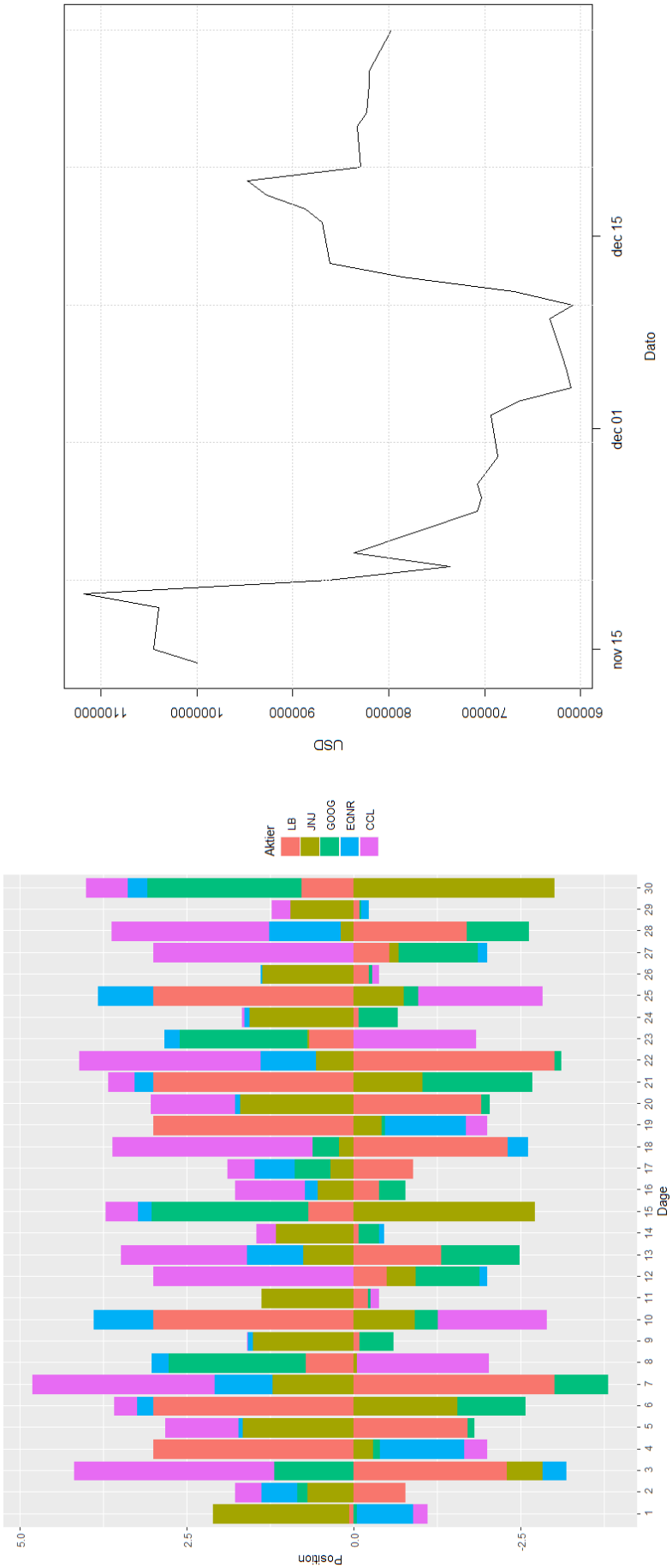


Figure A.6: Porteføljesammensætning og udvikling for Buy & sell with shorting med ARIMA-forecast

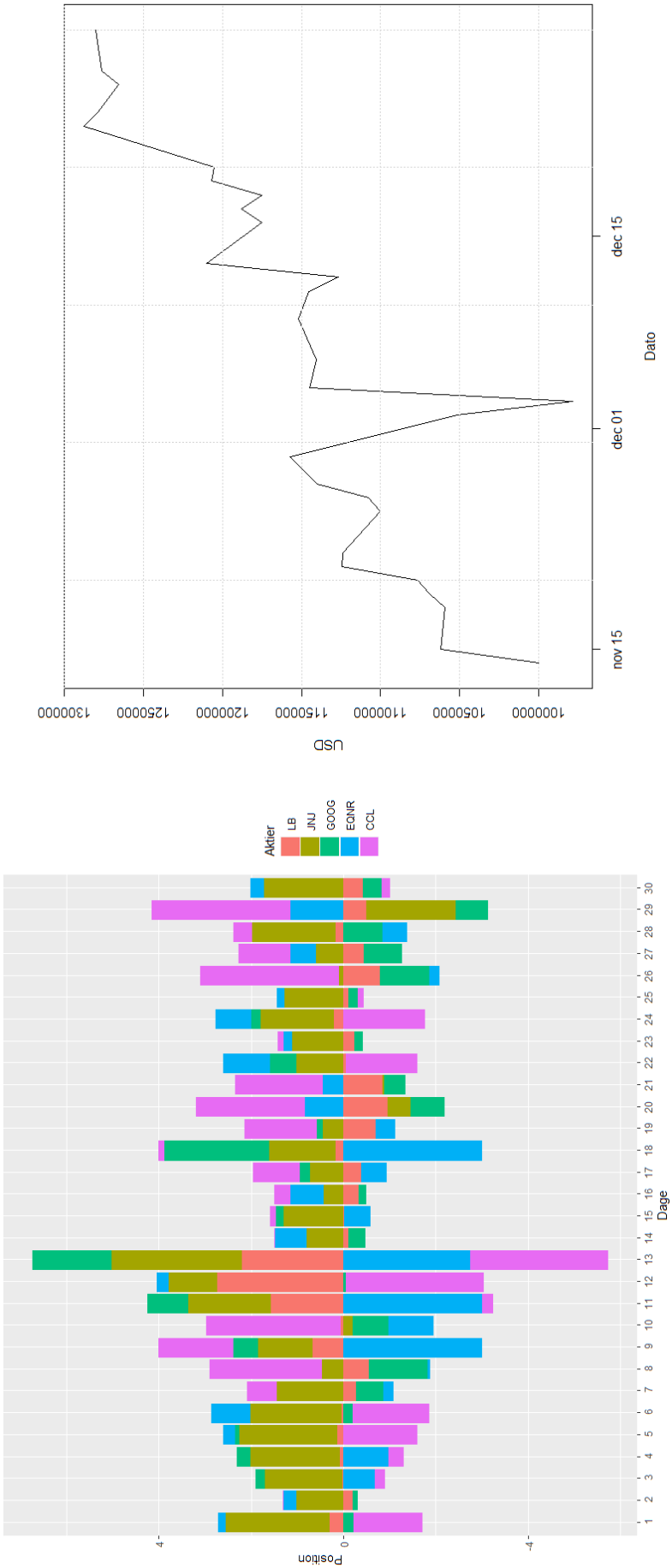


Figure A.7: Porteføljesammensætning og udvikling for Buy & sell with shorting med LSTM-forecast

Appendiks B

Specialet har i forbindelse med opbygning af LSTM valgt at følge Jojo Moolayil's bog fra 2019: "Learn Keras for deep neural Networks - A fast-track approach to modern deep learning with python". I sin bog beskriver Moolayil at der findes ingen gyldne regler hvorpå man teoretisk kan argumentere eller teste for om man f.eks. skal benytte 1, 2 eller 3 layers i sin NN. Derfor starter man oftest med simple og små NN, for at få en fornemmelse af dataet og således arbejder man ud fra det. Dette kunne også ses i kilderne fra litteraturreview, som fører ingen større argumentation for deres NN. Derfor vil dette speciale starte med at tune antallet af layers og neuroner. Hertil påpeger Moolayil enkelte tommelfingerregler:
- Afhængig af antal input dimension, så burde antallet af neuroner i første layer være en del af den eksponentielle udvikling i tallet 2: 4, 8, 16, 32, 64, 128 osv.
- For hvert layer man tilføjer burde antallet af neuroner falde eller være det samme.
- Et dropout på 0.3 (eller 30%) for hvert layer burde være nok til at undgå overfitting.

Da specialet bruger 60 observationer til at prædiktere en observation, så bliver antallet af neuroner i første layer sat lig 128 og herfra udvides netværket til flere og mere kompliceret modeller, som kan ses i tabel B.1 (Moolayil, 2019). Alle de forskellige netværker bliver sat op med et tilfældigt seed nummer (1337 for dette speciale). Dette betyder at initialværdierne for biasled og vægtningerne mellem de forskellige layers er de samme for alle modellerne, og således kan resultaterne reproduceres[1] og derved bedre sammenlignes da dette ikke vil være tilfældige hver gang en model testes.

Table B.1: LSTM-opbygning

	1. layer		2. layer		3. layer		4. layer	
	Neuroner	Dense	Neuroner	Dense	Neuroner	Dense	Neuroner	Dense
Model 1 (M1)	128	25	64	1				
Model 2 (M2)	128	25	128	1				
Model 3 (M3)	128	50	64	25	32	1		
Model 4 (M4)	128	50	128	25	64	1		
Model 5 (M5)	128	50	64	25	46	1		
Model 6 (M6)	128	75	64	50	32	25	16	1
Model 7 (M7)	128	75	128	50	64	25	32	1
Model 8 (M8)	128	75	128	50	64	25	64	1
Model 9 (M9)	128	75	64	50	64	25	32	1
Model 10 (M10)	128	75	64	50	32	25	32	1

I første omgang frygtede specialet et overfit, således startede tests med et dropout på 0.5 på hvert layer og derfra erfarede specialet, at dropout kunne falde mere, uden at resultere i et overfit. I efterfølgende

[1] Der kan dog stadig forekommer mindre udsving i resultaterne og det kan skyldes computertekniske indstiller af f.eks. GPU (Graphics Processing Unit) libraries som kan skabe en smule tilfældighed i resultaterne.

tabler fra B.2 til B.6 er RSME angivet og brugt til at udvælge de bedste LSTM-opbygninger. Samtidig er der enkelte tilfælde konstateret overfit, som betegnes med et * ved siden af RSME-værdiden.

Table B.2: LSTM-opbygning med dropout=0.5

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
JNJ	2.2	9,2	7,0	6,2	7,6	18,8*	18,0	4,4	16,7	21,1
GOOG	15.6	90,0	168,0	142,5	102,9	220,6	29,0	115,7	100,6	73,9
EQNR	0.5	0,5	1,0	1,0	0,4	0,7	0,4	0,7	0,7	0,4
LB	1.6	1,8	0,8	3,0	1,3	3,2	1,7	1,2	1,3	1,2
CCL	1.4	1,2	1,7	1,3	1,3	2,5	2,0	1,5	1,8	2,1

Table B.3: LSTM-opbygning med dropout=0.35

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
JNJ	4,2	6,6	5,2	3,7	5,5	7,7	8,7	5,4	9,2	11,6
GOOG	60,5	21,3	117,4	79,5	74,9	148,6	69,8	85,3	141,7	162,4
EQNR	0,3	0,5	0,3	0,9	0,4	0,4	0,6	0,4	0,5	0,5
LB	0,8	0,9	1,0	1,2	2,1	1,2	2,3	0,8	1,6	0,8
CCL	1,3	1,3	1,4	1,6	1,9	2,1	1,6	1,7	1,3	1,4

Table B.4: LSTM-opbygning med dropout=0.30

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
JNJ	3,6	3,5	2,0	7,5	2,2	12,8*	8,4	4,5*	4,1	6,9
GOOG	25,6	42,3	18,4	46,1	83,4	144,2	81,1*	47,0	84,9	168,2
EQNR	0,3	0,3	0,3	0,5	0,7	0,4	0,3	0,4	0,4	0,7
LB	2,0	1,2	1,6	2,1	0,8	1,3	1,4	2,3	2,2	2,2
CCL	1,7	1,4	1,9	1,3	1,2	2,3	2,6	1,7	1,6	3,0

Table B.5: LSTM-opbygning med dropout=0.25

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
JNJ	5,3	1,4	9,1	4,7	3,1	10,2	4,6*	5,1	2,0	7,2
GOOG	25,1	54,1	76,3	65,0	99,1	177,5	15,8	53,7	199,9*	61,2
EQNR	0,4	0,4	0,3	0,3	0,5	0,4	0,9	0,5	0,5	1,0
LB	0,8	0,8	2,2	1,0	0,8	1,0	0,8	2,3	2,2	2,3
CCL	1,5	1,1	1,7	1,7	1,6	1,3	1,4	1,9	1,4	1,4

Table B.6: LSTM-opbygning med dropout=0.2

	M1	M2	M3	M4	M5	M6	M7	M8	M9	M10
JNJ	5,4	2,5	5,8	2,6	6,9	10,0	5,2	3,4	6,1	6,6
GOOG	40,2	48,9	54,2	63,0	61,5	35,0	77,0	70,8	22,3	133,2
EQNR	0,4	0,30	0,3	0,3	0,3	0,4	0,3	0,4	0,3	0,4
LB	0,8	0,9	1,8	0,8	1,9	1,5	10,0	1,5	1,0	1,2
CCL	1,3	1,1	1,3	1,2	1,3	1,9	1,3	1,5	2,5	1,6

* - overfitter.

Ved test af 5 forskellige dropout-niveauer på 10 forskellige LSTM-modeller, kan det konkluderes at følgende modeller var bedste ud fra laveste RMSE og uden overfit (markeret med fed) i tabel B.7. Det kan hermed ses, at M1 og M2 som er LSTM-modeller med kun 2 layers faktisk er bedre til at minimere fejl i forecast, og herefter vil appendixet kun kigge på disse 2 typer af modeller. Dropout-niveauer for de

fem aktier er dog mere forskelligt med GOOG som opnår laveste RSME ved et dropout på 0.5, mens de 4 andre ligger noget lavere på henholdvis 0.2, 0.25 og 0.35.

Table B.7: Bedste LSTM-opbygning ved forskellige dropouts

	JNJ	GOOG	EQNR	LB	CCL
M2 med 0.25 dropout	**1.4**	2.2	4.2	5.3	2.5
M1 med 0.5 dropout	54.1	**15.6**	60.5	25.1	48.9
M1 med 0.35 dropout	0.4	0.5	**0.3**	0.4	0.4
M1 med 0.25 dropout	0.9	1.6	0.9	**0.8**	0.9
M2 med 0.2 dropout	1.2	1.4	1.3	1.5	**1.1**

Herefter vil specialet foretage en Grid Search som metode hvorpå hyperparametertuning kan foregå. Grid Search er forsøgt overskueliggjort i nedenstående tabel B.8. Specialet vil tune på 3 hyperparameter: Batch, Dense og læringsraten. Dette gøres ved indstille med modellerne med 3 forskellige værdier for batch med 32, 64 og 128, og 3 forskellige værdier for dense og læringsrate. Dense har i de tidligere modeller (M1-M10), været sat til 1 i sidste layer og 25 i andensidste layer osv. Dermed sagt at det første layer i M1 og M2, vil specialet fordoble fra 25 til 50, og fra 50 til 75. Ved denne metode eksperimenteres med forskellige kombinationer (markeret med x i tabellen) af de forskellige værdier for de forskellige hyperparameter. Dette kan således være tidskrævende proces hvis mere end 3 hyperparameter skal tunes, og således grunden at specialet blot har valgt 3 og dermed kun 27 forskellige kombinationer.

Table B.8: Grid Search

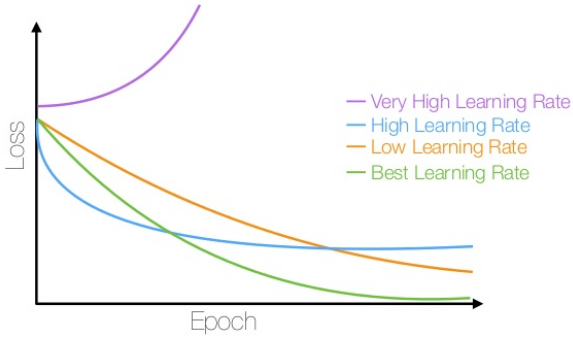
		Batch-størrelse			Dense			læringsrate		
		16	32	64	25	50	75	0.001	0.0005	0.0001
Batch-størrelses	16				x	x	x	x	x	x
	32				x	x	x	x	x	x
	64				x	x	x	x	x	x
Dense	25	x	x	x				x	x	x
	50	x	x	x				x	x	x
	75	x	x	x				x	x	x
læringsrate	0.001	x	x	x	x	x	x			
	0.0005	x	x	x	x	x	x			
	0.0001	x	x	x	x	x	x			

Hyperparametertuning af netop batch, dense og læringsrate skyldes at batch kan have store betydning for resultatet og ofte en hyperparameter, som ønskes justeres. Dette kunne også ses i kilderne fra afsnittet litteraturreview, som tunede denne hyperparameter i håbet om bedre resultater. Litteraturen og teorien bag indstilling af dense er ret begrænset efter specialets opfatning, og derfor en hyperparameter værd at undersøge nærmere og tune på. Læringsraten blev valgt grundet specialets ønske om en mere jævnt læringstempo for netværkene, hvilket er forsøgt afbilldet nedenfor i figur B.1.

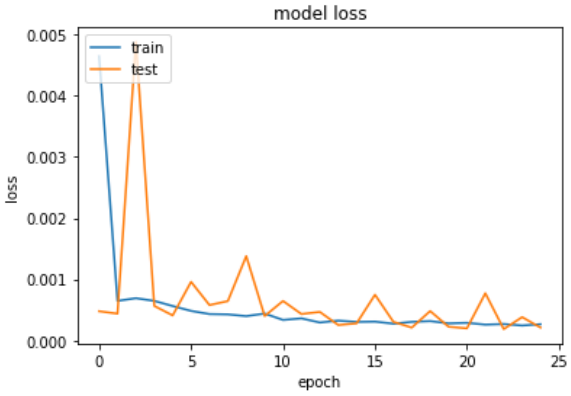
Til venstre (a) ses kurver med scenarier for forskellige læringsrater. Det mest optimale scenarie vil være den grønne kurve, hvor modellen konstant lærer og marginal effekten af en ekstra epoch aftager langsomt. Til højre (b) ses læringstempoet (blå linje) for specialets model for JNJ med M2 og 0.25 som dropout. Det kunne således tyde på at modellen lærer for hurtigt ift. det optimale, da marginal effekt af en ekstra

epoch er meget lille, efter 2 epochs. Derfor har specialet valgt at teste med lavere læringsrater, således en mere glidende udvikling opnås som i det grønne scenarie fra figur (a).

Figure B.1: Læringsrater



(a) Eksempler på læringsrater



(b) Læringsrate for JNJ ved M2 med 0.25 dropout

Ved udførelse af Grid Search opnås følgende resultater i tabel B.9.

Table B.9: Grid Search RMSE-resultater

Batch	Dense	lærings -rate	JNJ	GOOG	EQNR	LB	CCL
16	25	0.001	3.605*	37.787	0.564	0.913	1.025
16	25	0.0005	1.113	55.446	0.332	1.12	1.079
16	25	0.0001	2.083	58.243	0.447	0.796	1.289
16	50	0.001	3.122*	46.043	0.472	0.791	1.029
16	50	0.0005	**1.109**	62.758	0.302	1.733	1.042
16	50	0.0001	2.524	36.4	0.441	0.786	1.326
16	75	0.001	2.786*	**14.956**	0.388	0.779	**0.969**
16	75	0.0005	1.164	50.839	**0.282**	1.599	1.05
16	75	0.0001	2.098	54.016	0.431	0.862	1.377
32	25	0.001	1.4*	54.08*	0.36	0.8	1.11
32	25	0.0005	1.238*	81.082	0.339	0.789	1.15
32	25	0.0001	1.378	74.83	0.526	0.752	1.58
32	50	0.001	1.543	100.271	0.467	0.973	1.324
32	50	0.0005	1.474	76.191	0.344	0.784	1.159
32	50	0.0001	1.553	73.552	0.515	0.771	1.738
32	75	0.001	2.32*	50.466*	0.296	0.818	1.137
32	75	0.0005	1.174	64.734	3.709	0.787	1.17
32	75	0.0001	1.305	58.879	0.518	0.769	1.664
64	25	0.001	2.213	95.003	0.337	0.866	1.257
64	25	0.0005	3.205	84.458	0.416	0.808	1.508
64	25	0.0001	3.575	58.147	0.593	0.733	2.174
64	50	0.001	2.268	79.456	0.331	0.825	1.295
64	50	0.0005	4.107	88.144	0.44	0.896	1.574
64	50	0.0001	3.618	42.809	0.569	**0.727**	2.128
64	75	0.001	2.291	82.72	0.328	0.822	1.284
64	75	0.0005	3.627	92.629	0.435	0.937	1.426
64	75	0.0001	3.285	30.973	0.581	0.734	2.179

* - overfitter.

I tablen kan det ses, at 4 ud af 5 aktier prædikterer bedst ved en batch lig 16, mens LB bedst prædikteres ved 64 batchs. Samtidig opnås lavere RMSE-værdi ved at øge antallet af dense fra 25 til enten 50 (for JNJ og LB) eller 75 (for GOOG, EQNR og CCL). Således gælder det, at en lavere RMSE-værdi opnås ved ændring af læringsrate til henholdsvis 0.0005 for JNJ og EQNR og 0.0001 for LB.

Samtidig kan det ses, at overfitning kun forekomme ved en læringsrate på 0.001 med en enkelt undtagelse med 0.0005. En lavere læringsrate afhjælper således både overfitning og giver bedre resultater. Disse 5 modeller markeret med fed i tabel B.9, vil blive brugt videre til den sidste hyperparameter, som specialet vil justere på. Dette vil være antallet af epochs, altså hvor mange gange skal modellen trænes med træningssættet og justere parameter. Specialet har indtil videre brugt kun 25, men vil nu se nærmere på effekter af en forøgelse af epochs, hvilket er præsenteret i tabel B.10.

Table B.10: Justering af epochs

Epochs	JNJ	GOOG	EQNR	LB	CCL
25	1.109	**14.956**	0.282	**0.727**	0.969
50	**1.095**	52.885	0.334	0.749	**0.813**
75	1.253	129.642	0.364	0.837	0.992
100	5.607	96.497	0.466	0.760	0.953
500	4.106	254.242*	**0.249**	0.784	0.941

* - overfitter.

Resultaterne viser sig ikke bedre med flere epochs for GOOG og LB, som har lavest RMSE-værdi ved 25 epochs. Dog for de resternede 3 viser 50 epochs for JNJ og CCL at give de bedste resultater, mens for EQNR er 500 epochs det bedste resultat. Samtidig fås et overfit på GOOG på 500 epochs, hvilket også resulterer i et dårligt fit og en høj RSME-værdi.

B.0.1 Opsummering

De opnåede resultater efter LSTM-opbygning og hyperparmetertuning samt justering af epoch er opsummeret i nedenstående tabel:

Table B.11: Samlede resultater af hyperparametertuning opgjort i RMSE

Resultater efter:	JNJ	GOOG	EQNR	LB	CCL
LSTM-opbygning	1.4	15.6	0.283	0.79	1.10
Grid Search	1.11	14.96	0.282	0.73	0.97
Reduktion i %	-21%	-4%	-0.3%	-8%	-12%
Justering af epochs	1.095	14.96	0.25	0.73	0.81
Reduktion i %	-1%	-	-11%	-	-16%
Samlet reduktion i %	-22%	-4%	-11.8%	-8%	-26%

Opbygning af en LSTM-model er selvfølgelig af afgørende betydning og giver fundamentet til det videre arbejde med hyperparametertuning. På trods af et god fundament kan ydereligere justeringer føre til store og mindre forbedringer. JNJ opstår således en reduktion i RMSE på 21 pct. som den bedste forbedring, mens EQNR kun oplever en forbedring på 0.3 pct. ved af tuning af batch, dense og læringsrate. Ydereligere forbedringer kunne opnås ved ændringer af epochs, som særligt reducerer CCL's RMSE-værdi med 16 pct., og samlet set blev reduceret med 26 pct. GOOG var derimod tidsserien, som LSTM-modellen oplevede færrest procentmæssige forbedringer af hypermetertuning med kun 4 pct. Efter justering af epochs vil specialet ikke foretage ydereligere for at forbedre modellerne, og så vil de sidstnævnte modeller blive benyttet i specialet til at forecaste.