PROJEKT
ANALYS AV AKTIEKURSER MED RNN

Hannes Rabo , Victor Carle, *Civ.ing. Informationsteknik*, KTH

***Sammanfattning*—In this project we took it upon ourselves to create a stock price predictor using historical stock prices and sentiment analysis of texts regarding the company which is being looked at. These inputs are fed into a recurrent neural network with LSTM layers which predicts the stock price for next day. Using this technique for three different stocks we got much better result than what was achieved by just buying the stock in the beginning of the year. Best results were achieved using the stock TEVA which got a growth of 156.9% compared to the stock which decreased in price by 63.14 % during 2017. Least efficient was the analysis of AAPL which increased in price with 46.27 % during 2017. The model was able to achieve a growth of 78.19 % during the same time which still is much better.**

I. Introduktion

AKTIEMARKNADEN är en plats där mycket pengar finns att tjäna. Tidigare har man manuellt behövt göra undersökning om potentialen hos ett företag för att sedan kunna göra en gissning på om det är värt att investera i. Manuell analys leder såklart till att en väldigt begränsad mängd data behandlas och leder till osäkra approximationer. Den digitala eran har medfört ofantliga mängder data att analysera, alldeles för mycket för en människa. Därför är AI tekniker som kan analysera enorma mängder data av stort intresse för många företag.

Då kvarstår problemet: Hur skapar man ett program som analyserar en stor mängd data och skapar en rimlig approximation för framtiden.

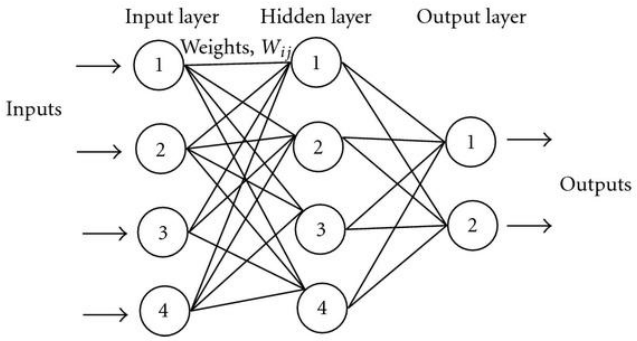
I det här projektet har målet varit att med hjälp av ett RNN (Recurrent Neural Network, se figur 2) skapa ett program som kan förutspå hur utvecklingen för aktievärdet av ett företag kommer se ut. Datan vi vill använda för att göra detta möjligt är först och främst tidigare aktievärden. Sedan vill vi även använda oss utav sentiment analysis av media. Det vill säga att i beräkningen ha med analyser ut av om det som skrivs på sociala medier om företaget är positivt eller negativt i snitt på dagen vi hämtat aktievärdet. Detta ger oss en större chans till att göra en bättre kalkylerad gissning utav företagets aktievärde, mycket negativ media leder ofta till att aktievärdet sjunker och vise versa.

II. Teoretisk bagrund

Under detta avsnitt presenterar de olika tekniker vi använt kortfattat.

A. RNN

Aktiemarknader är något som i sin natur är ett sekventiellt fenomen. Det innebär att man i en punkt inte kan se hur rörelsen kommer se ut utan att ha någon insyn i hur det sett ut tidigare. Om vi ser ett plötsligt fall i aktien kan vi anta att det är det större sannolikhet att fallet fortsätter även i en kort framtid. Detta gör att vi måste ta hänsyn till sekvensens utseende för att kunna avgöra beteendet i framtiden. För normala neurala nätverk ANN (Aritficial Neural Networks, se figur 1 så är uppbyggnaden sådan att varje inmatat värde ger ett utvärde. När nästa värde matas in så finns ingen information bevarad från föregående utan nätverket nollställs mellan varje omgång (efter att träningen avslutats) [1]–[3].

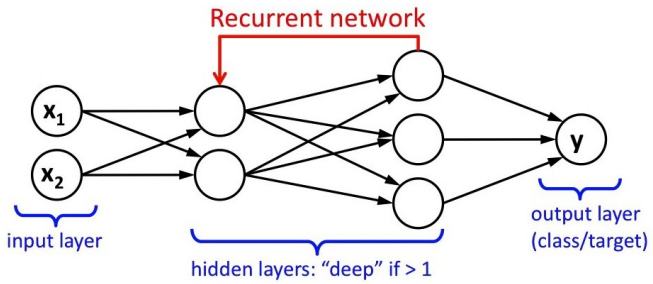


Figur 1. Artificial Neural Network

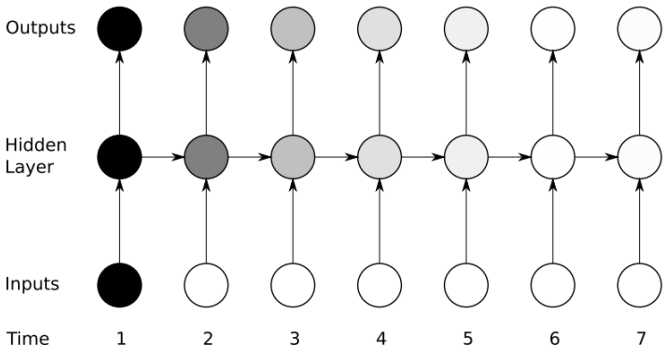
Det alternativ som passar bäst för den här typen av dataanalys är istället ett RNN (Recurrent Neural Network) som har ett dolt internt tillstånd. Det innebär att för varje inmatat värde så behåller nätverket en viss del av informationen från föregående steg och kan använda detta i nästa. För det exakta fallet kommer vi inte bevisa att marknaden har markovegenskaper vilket hade visat att modellen fungerar men vi antar att det är tillräckligt nära detta för att göra approximationen. Renner, Peinke och Friedrich har genomfört en studie av data av aktiemarknaden där de hittade bevis för att prisändringarna kan beskrivas som en markovprocess [4]. Detta innebär alltså att vi med hjälp av föregående tillstånd kan beräkna sannolikheten för nästkommande tillstånd oberoende av alla tidigare tillstånd. Varje tillstånd innehåller alltså rekursivt information från alla tidigare genom att föregående tillstånd alltid är baserat på det innan. Just denna egenskap är det som utnyttjas i RNN då det förenklar beräkningar och komplexitet.

B. Sentiment Analysis

Sentiment analysis är vetenskapen bakom att analysera känslorna bakom saker som folk har skrivit eller sagt. Ett av sätten att analysera känslorna bakom en text är att man



Figur 2. Struktur för Recurrent Neural Network



Figur 3. Vanishing Gradient problem.

helt enkelt använder sig av ett bibliotek där ord har givits en positiv eller negativ vikt. Sedan tar man ord för ord och summerar värdena för att avgöra om texten i helhet är positiv eller negativ. Detta sätt att tillväga gå är mycket begränsad på grund av det faktum att det finns saker såsom ironi. För att upptäcka sådana saker krävs mycket mer sofistikerad teknologi som utvecklas med hjälp av artificiell intelligens.

Sentiment analysis är nu ett enormt värdefullt verktyg tack vare den enorma mängd sociala medier data som finns. Används mycket inom marknadsundersökningar för att avgöra vad folk tycker om ett företag och kan därför användas för att förutse deras beteenden kring investeringar [5].

C. Problem

Att använda sig av RNN är inte helt problemfritt. Det finns många aspekter som påverkar resultatet och vi kan bland annat stöta på “Vanishing Gradient Problem” som gör att nätverket får väldigt dåliga inlärningshastighet. Problemet finns i nätverk som använder sig av backpropagation där mängden information för varje steg i nätverket har mindre och mindre värden att utgå ifrån allt eftersom vi kommer närmare starten på nätverket (se figur 3). Detta är inte något som är unikt för RNN men de lider ytterligare av egenskapen på grund av att de ofta blir djupa på grund av de steg som finns för de ytterligare steg som behövs för att hålla tillståndet hos den sekvens av data som vi analyserar. En av de lösningar som finns för det här är att använda sig av en kategori med neuroner som kallas för block (Long Short-Term Memory). Fördelen här att det finns en direkt väg mellan lagren där data kan överföras utan att minska enligt enligt de ekvationer som normalt appliceras på all data under justeringen av vikter vid backpropagation [6].

Det finns två stora problem med att använda sentiment analysis för att avgöra hur det går för ett företag. Först och främst så är teknologin inte tillräckligt sofistikerad ännu för att avgöra om en “negativ” åsikt av ett företag faktiskt är negativ, återigen förstår maskiner inte ironi. Detta leder uppenbarligen till felaktig indata.

Sedan finns det två problem med att förlita sig på folks åsikter. Om man inte använder sig utav pålitliga källor där man kan anta att det som skrivs är korrekt, ett bra exempel på en medioker källa är twitter. Där vilken person som helst med en telefon kan yttra åsikter. Det andra problemet med att kolla på folks åsikter är att de bara är relevanta om det är ett företag vars konsumenter är vardagliga människor. Om det inte är fallet spelar deras åsikter ingen roll.

III. Metod

Under detta avsnitt presenteras metoden för att skapa modellen och prototypen under arbetet.

A. Design

Under detta avsnitt behandlas designen av systemet. Detta innefattar hur det ska byggas med olika lager, vilken indata som behövs, hur data processas och övergripande arkitektur.

1) Invärden: För sentiment analysis delen vill vi ha en input bestående utav bland annat nyhetsartiklar eller saker folk har skrivit på sociala medier om det företag vi granskar. Vi behöver även veta datumen för när dessa saker skrev för att på ett bra sätt kunna använda datan i samarbete med börsvärden. För att få tillgång till all den här datan behöver man ha tillgång till API:n utav dessa media platformer.

För att kunna nyttja den här datan på ett sätt så att den assisterar i bedömning av företagets framgång måste vi transformera dessa texter till värden som avgör textens polaritet. Varje text får ett värde mellan -1 och 1, som representerar dess polaritet. Finns det flera texter tillgängliga för ett datum så summeras dessa och snittet används. Dessa värden matas in i en CSV fil där vi har ett polaritets värde för varje datum. I fallen där man inte har tillgång till texter på vissa datum så blir polaritets värdet 0, vilket egentligen betyder att det åsikten är neutral.

Enligt tidigare resonemang ska vi använda oss av flera olika källor till information för att bättre kunna förutsäga kommande värden. För att kunna relatera värden till marknadsutveckling ska vi också införa ytterligare ett värde som har ett hyfsat stabilt marknadsvärde i jämförelse med hur aktiekurser brukar röra sig. Det värde som valts ut för detta är guldpriset i dollar. Det finns en viss trend där guldpriset ökar då aktiemarknaden faller på grund av den ökade efterfrågan på stabila tillgångar. Denna ökning brukar dock vara mindre än de mer volatila aktiepriserna vilket gör det till ett lämpligt val. För att modellen med ett RNN nätverk med LSTM celler ska fungera är det också viktigt att vi kan koppla tidpunkten för varje invärde till andra kategorier. För att förenkla processen väljer vi att ha det som ett krav att det ska finnas lika många och samma

dagar på samma plats i de tabeller som används som invärden. Sammanfattningsvis har vi alltså två tabeller med ett invärde per dag (utom helger då aktiemarknaden är stängd vid dessa tillfällen).

Innan värden kan matas in i ett RNN måste datan dock processas. Invärden måste delas upp i delsekvenser av bestämd längd då LSTM celler i TensorFlow arbetar med fixerade storlekar i sina invärden. Värden måste plockas ut som en sekvens av efterföljande värden för att vi ska kunna träna nätverket i att hitta denna typ av mönster. Detta betyder däremot inte att vi inte kan använda flera block av sekvenser under träningen. Första steget i träningsprocessen ska därför bestå av en uppdelning av invärden i många sekvenser som sedan blandas för att öka möjligheten att kunskapen som nätverket får om datan kan generaliseras. För att få ut maximalt från datan ska blocken överlappas vilket gör att varje invärde kan användas flera gånger men fortfarande ingå i en ny sekvens. Detta på grund av den begränsade mängd data som vi arbetar med i det här fallet. I illustrationen är steget mellan början på varje block alltid ett men i kod skall detta värde gå att ändra.

Eftersom vi försöker använda den data vi har maximalt är det däremot viktigt att vi behandlar data som ska användas för test helt separat. Detta innebär att databehandlingsmodulen måste separera data för test och träning innan den bygger sekvenser. I annat fall hade många dataelement ingått i båda varianter vilket gett ett osäkert resultat.

2) Utvärden: När det gäller utvärden från nätverket så behöver vi förhålla oss till att det måste gå att förutspå mer än ett värde. Detta innebär alltså att nätverket måste kunna generera alla de värden som behövs för att göra nästa. Förutom detta krav så behöver vi även efterbehandla de värden som det neurala nätverket skapar. Detta för att göra det möjligt att tolka resultatet vilket annars kan vara nästa omöjligt då det består av en lång sekvens av värden som ligger i närheten av de värden som nätverket använt sig av.

Efterbehandlingen skall använda sig av de förutsagda värden på aktiekursen för att avgör ifall aktien är påväg upp eller ner. Med den informationen kan vi enkelt bygga en algoritm som köper maximalt antal aktier då aktien är påväg upp och istället säljer alla aktier om den är påväg ner. Det ger oss möjlighet att tolka utvecklingen i förhållande till hur aktien har gått.

3) Process och arkitektur: För att göra förutsägelser av aktiemarknaden behövs ett RNN nätverk enligt tidigare resonemang. Det kan bli problem och är ovanligt att endast använda sig av LSTM celler i den här typen av nätverk och vi kan genom test se att prestandan ökar markant genom att lägga till två lager med fullt anslutna celler i slutet av nätverket [6].

Vi har också speciella krav på nätverkets början eftersom vi vill ha flera parallella invärden till nätverket. Det är också en fördel ifall RNN-delen har minne från båda samtidigt. Detta gör att vi vill använda oss av två RNN efter varandra i början. På grund av hur arkitekturen ser ut accepterar LSTM celler endast en sekvens av värden till skillnad från många andra typer av lager för neurala nätverk som tar ett invärde åt gången. Detta gör att första lagret i det neurala nätverket måste skicka ut en sekvens av värden (i form av en tensor med ytterligare en dimension för detta). För att sedan kunna använda sig av båda invärden måste vi också lägga ihop värden från båda LSTM lagren som tar in rådatan. Till detta används ett lager som lägger datan efter varandra vilket syns i figur 5 som “concatenate”. För varje extra lager med LSTM celler finns det en ökad risk för brus och att nätverket över anpassas till datan och kunskapen inte går att generalisera. Därför väljer vi att använda oss av dropout layers som plockar bort en viss del av datan och kopplingar som skapas [7].

För att göra det enkelt att bygga den här typen av modeller har vi använt oss av Keras vilket är ett bibliotek som tillhandahålls tillsammans med Tensorflow [8].

4) Krash, data och kontroll: För att göra den här typen av verktyg mer användarvänligt skall de hjälpmedel som tillhandahålls av tensorflow att användas. Detta innebär att om man bygger programmet enligt de rekommendationer som finns för tensorflow så kommer det automatiskt att gå att övervaka hela processen med träning och informationsflöden med hjälp av verktyget tensor board som körs i webbläsaren. Förutom detta så sparas även mellansteg under hela träningsprocessen vilket gör att det finns möjlighet att återuppta en kraschad träningssession. Kod för återupplivning av denna typ har vi inte som primärt mål i denna undersökning då det är mindre relevant för så korta körningar som vi får i det här fallet. Vad som är mer relevant är däremot möjligheten att ladda in en färdigtränad modell från en disk vilket diskuteras mer senare i rapporten [8], [9].

B. Implementation

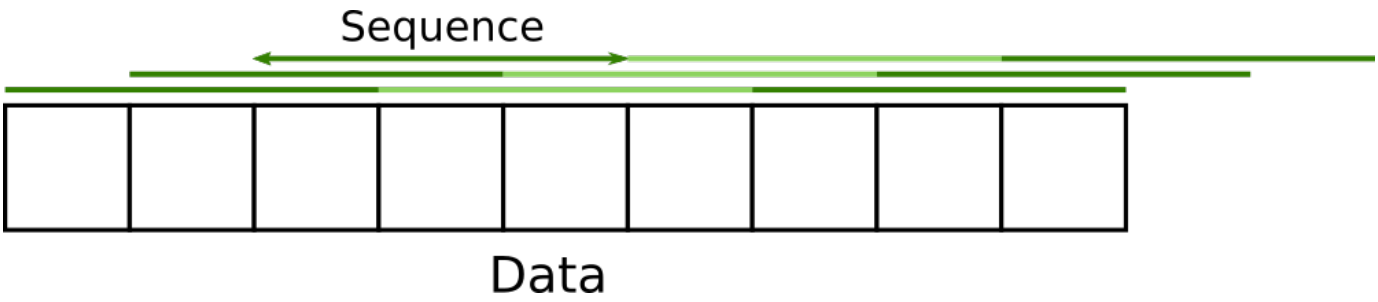
För implementation av detta nätverk med tillhörande dataprocessning är python det mest naturliga valet i och med googles utveckling av TensorFlow [10] och de många tillhörande hjälpverktyg som finns tillgängliga. Det ger möjlighet att på ett enkelt sätt bygga mycket avancerade neurala nätverk med hög prestanda som även går att skala upp och köras på kluster samt grafikkort. Detta är en omöjlighet att utveckla utan denna typ av verktyg på den tid som använts för detta projekt.

Projektet delas in i tre primära delar. Huvudfilen är stock_price.py vilken innhåller kod för att koppla ihop de andra modulerna men även hantera IO operationer. Den andra delen är model.py vilken skapar en modell för tensorflow och även kan hantera generering av data efter träning av nätverket. Sista delen är data_processing.py vilket är den modul som hanterar en del av det förarbete som krävs innan datan kan användas i nätverket. Här delas bland annat datan upp i sekvenser enligt tidigare beskrivning samt i delar för träning och test. En kort sammanfattning av hur filerna är uppbyggda finns i appendix A.

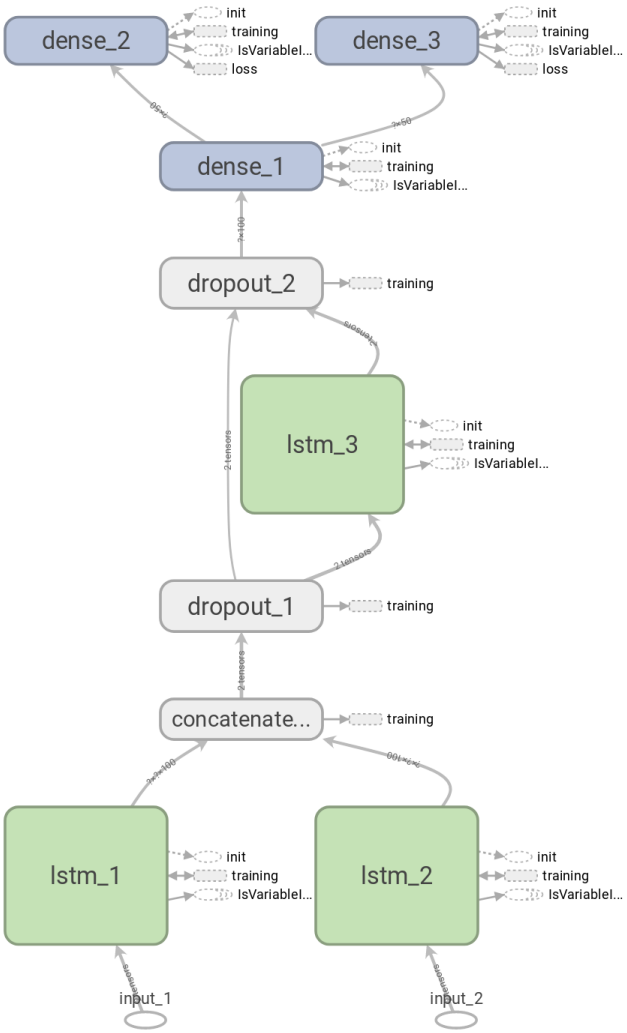
C. Test

De test som användes under arbetets gång plockade oftast bort ur koden för att göra den mer läslig i slutändan. Detta innebär att endast små delar av tester som genomförts för att säkerställa funktionalitet faktiskt finns med i den levererade prototypen.

För test av data_processing.py användes framförallt listor med nummer som kunde skickas genom modulen och kontrollera att beteendet stämmer väl överens med det förväntade. Dessa egenskaper presenteras i listan nedan.



Figur 4. Maximalt antal unika sekvenser uttagna ur lista.



Figur 5. Arkitektur och flöde av information i nätverket

För modulen som bygger tensorflow modellen behövdes inte så många tester eftersom tensorflow automatiskt validerar att modellen kommer fungera innan testet kan startas. Här gäller alltså enbart designval och restriktioner.

Sista modulen som kopplar ihop övriga delar av projektet är beroende av att övriga ger rätt data. Här har mycket rimlighets tester och kontroller av data genom plottning på graf använts då det är komplext att bevisa funktionalitet på många delar. Eftersom vi utgår från att de flesta komponenter som används här redan har korrekt funktionalitet så finns det inte lika mycket som kan gå fel i den här delen av programmet. Det som är intressant att testa här är framförallt hur beräkningen av utvecklingen går till för att vara säker på att det är korrekt.

Tabell I
TESTFALL FÖR DATAPROCESSNING

Namn	Krav
Test och träningsuppdelning	Antalet element som funktionen returnerar i de olika kategorierna skall stämma överens med den procentsats som anges i parametern test_ratio.
Testsekvenser	Inga data från träningen får ingå i testen
Start på block	Varje sekvens skall starta med det första elementet förskjutet lika många steg från förra starten som anges i parametern step_size
Storlek på block	Varje sekvens ska ha exakt den längd som anges i parametern input_size
Ordning på sortering	Om två block med samma storlek stoppas in skall sorteringen överensstämma för båda. Detta för att göra det möjligt att passa ihop datum när de används för träning.
Svar	Det värde som skall förutspås efter varje sekvens (svaret) får inte ingå i sekvensen.

Tabell II
TESTFALL FÖR HUVUDMODULEN

Namn	Krav
Utveckling ska följa aktien	Om aktien inte har någon utveckling och förblir på samma värde skall ingen utveckling av kapitalet ske trots upprepade köp och försäljningar.

D. Test

För användning av programmet behövs både prisutvecklingen för guldpriset samt den aktie som man har tänkt att analysera. Dessa finns tillgängliga som historiska data från bland annat yahoo på https://finance.yahoo.com. För att programmet ska fungera måste man enligt tidigare diskussion ha data för samma dagar och samma antal i filen. Detta för att göra det enklare (och möjligt) att passa ihop data från samma dagar tillsammans. Scriptet är inte helt automatiserat så för att välja den aktuella filen behöver man gå in i filen stock_price.py och redigera den textsträng som bestämmer vilken fil som laddas in. Värt att notera är att det tar ganska lång tid att träna nätverket på den data som finns (ca 1-2 minuter på en genomsnittlig persondator). Det finns därför också möjlighet att ladda in en färdigtränad modell som skapats under tidigare körning. Detta är ytterligare en variabel som går att ändra i huvudfilen.

Tabell III
RESULTAT FÖR CSCOS AKTIE

	Startvärde	Slutvärde	Utveckling
Modell	100	200.92	100.9 %
Aktie	30.37	37.69	24.8 %

Tabell IV
RESULTAT FÖR AAPLS AKTIE

	Startvärde	Slutvärde	Utveckling
Modell	100	178.19	78.19 %
Aktie	117.38	171	46.27 %

Tabell V
RESULTAT FÖR TEVAS AKTIE

	Startvärde	Slutvärde	Utveckling
Modell	100	256.82	156.9 %
Aktie	44.32	16.33	- 63.14 %

För att använda programmet behövs en python3 miljö finnas installerad på datorn. Eftersom ett antal bibliotek också används för att göra det möjligt att skapa komplexa och effektiva modeller behöver även dessa installeras. En lista finns i appendix B.

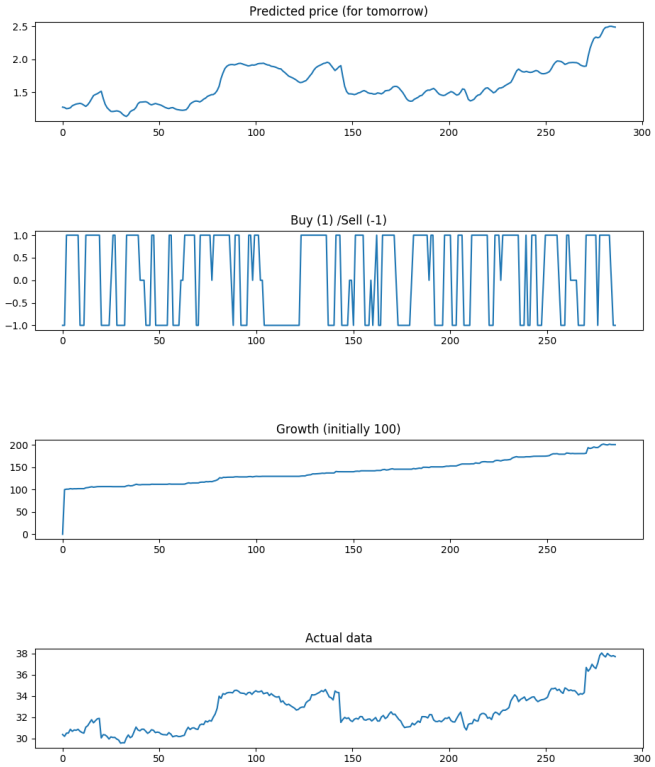
För att köra programmet används kommandot python stock_analysis.py vilket kommer att genomföra analysen enligt de inställningar som finns i form av variabler i huvudfilen. Resultatet är visas i en graf som visar utveckling för ett antal olika relevanta variabler (förutspått pris, köp/sälj, utveckling enligt nätverkets köpråd, aktiens utveckling). Relevant här är framförallt att observera hur utvecklingen blir på investerat kapital och jämföra detta med den utveckling som man fått om man bara köpt aktien direkt och inte gjort några förändringar. De exakta siffrorna för utvecklingen enligt modellen och aktien som den är skrivs också ut i konsolen efter att programmet har kört färdigt.

IV. RESULTAT

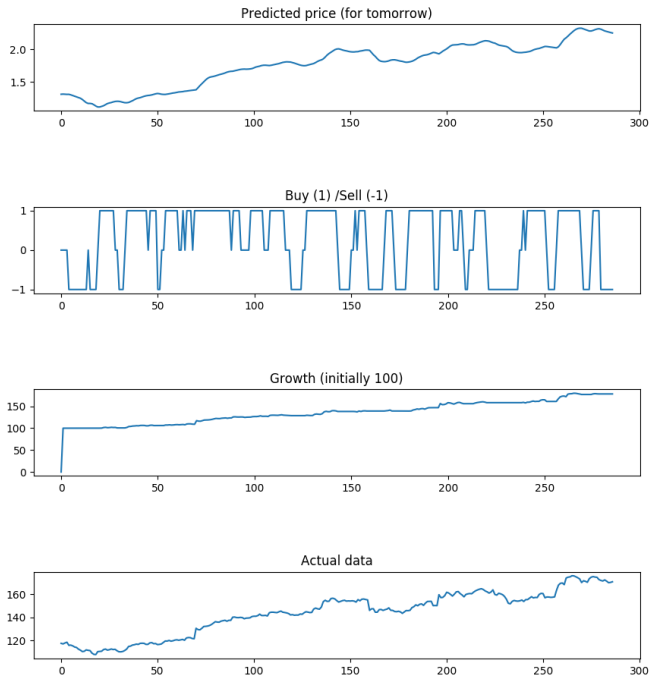
Med indata för CISCOs aktie mellan 30/7 - 2002 till 8/12 - 2017 där 10% av datan användes för test (ca ett års data) fick vi resultatet visas nedan. Resultatet efter 20 epokers träning illustreras i figur 6 samt i tabell III.

Vid en analys av Apples aktie för samma tidsperiod får vi inte riktigt lika bra resultat men ändå starkt positiva. Resultatet illustreras i figur 7 samt i tabell IV.

Det går däremot inte alltid lika bra beroende på vad det är för typ av aktier. Om man istället väljer att analysera aktier som under senaste året har gått mycket negativt så blir kapital utvecklingen också negativ för vissa aktier. Vid längre träningstid (fler epoker) kan man däremot vända den trenden och istället ha en positiv utveckling trots negativ aktietrend. Här visas analysen av aktien TEVA under samma tidsperiod som båda andra. Resultatet illustreras i figur 8 samt i tabell V.



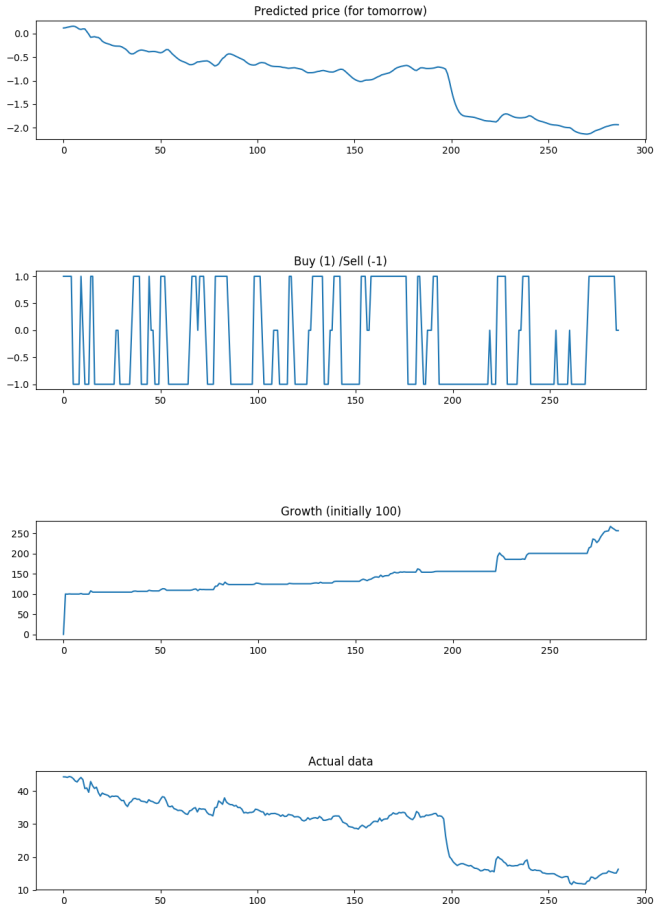
Figur 6. Resultat för CSCOs aktie



Figur 7. Resultat för AAPLs aktie

V. ANALYS

Tyvärr så misslyckades försöket att analysera personer känslor kring företaget. Inte på grund av att problem med att komma fram till polaritets värden på texten. För det användes en färdigbyggd modul med värden på ord vilket kan användas för att beräkna polariteten totalt. Problemen uppstod istället



Figur 8. Resultat för TEVAs aktie

i att hitta ett bra API. Twitters API tillät oss endast att hämta data från sju dagar tillbaka, vilket inte är tillräckligt. Det var väldigt få nyhetssidor som bidrog med ett API som uppfyllde de krav som fanns. Nämligen att kunna söka i deras nyhetsartiklar efter ett specifikt företag och ett specifikt datum. När vi väl hittade det vi sökte efter var antalet förfrågningar mycket begränsat eftersom de vill ha betalt för att man ska få göra fler än exempelvis 100 på en dag eller helt enkelt försökte skydda sig själva från överbelastning. Det sista alternativet för att i alla fall ha någon form av sentiment data var att använda sig utav ett API med färdiganalyserad data men tyvärr är detta inte en service som erbjuds gratis.

Trots att vi inte har input från människor känslor så har vi skapat en relativt bra modell för aktiemarknadsanalys. I de grafer som presenteras under resultatdelen ser man hur den ganska nära följer den verkliga grafen vilket gör att vi vet att den åtminstone gör hyfsade förutsägelser. Det som däremot visar på det verkliga resultatet syns istället på utvecklingen över tid. Vi ser att varje gång aktiekursen börjar svänga upp så har modellen snabbt hängt på och investerat men att utvecklingen nästan aldrig går ner då den direkt sålt alla aktier. Såklart blir det bäst när aktiekursen är påväg uppåt eftersom det är då man kan tjäna pengar. En aktie som växlar mellan ganska höga och låga värden skulle vara det optimala då det ger möjlighet att följa med i alla uppgångar med unvika nedgångar. Skulle det däremot komma en stor plötslig nedgång finns det en stor risk att modellen inte skulle hinna sälja i tid.

För att förhindra att den köper och säljer konstant vid minsta ökning/minskning finns det en tröskel som ger en rimligare projektion av hur det skulle kunna se ut i verkligheten om man använder sig utav ett sånt här program för att avgöra när man ska köpa och sälja. Oftast kostar det en mindre summa pengar att köpa och sälja aktier vilket gör att vi inte vill sälja och köpa allt för ofta. Med den konstant som används i dagsläget har modellen gjort ca ett köp varje vecka vilket är helt inom rimliga gränser för vad man kan genomföra manuellt. Såvida modellen inte har några katastrofala fel inbyggda har vi skapat ett verktyg som kan mångdubbla den avkastning man får av investeringar genom ett mindre antal köp och försäljningar varje år. Detta räddar dig däremot inte mot eventuella börskrascher som inte gått att förutse med den data som funnits tillgänglig. En annan betydande faktor är att det är svårt att hitta bra data för vissa aktier. Detta påverkar såklart hur precis grafen är då maskininlärning är helt beroende av att hitta mönster i den data som används under träningen.

Ett uppenbart fel i det här programmet är att vi inte räknar med att det kostar pengar att köpa och sälja aktier. Ett problem med detta är att de procentuella ökningarna kan låta väldigt lockande men för varje köp och försäljning förlorar man en mindre summa pengar. Detta värde bör i vårt fall vara ganska liten då det oftast inte kostar mer än någon eller några kronor varje gång. Detta är inom rimliga gränser om man köper och säljer under 50 gånger per år.

Vi räknar även med att våra tillgångar alltid köps och säljs på de exakta tillfällen vi gör dessa transaktioner. Det kan ge en falsk projektion av hur mycket man kan tjäna då man inte alltid kan sälja alla ens tillgångar framförallt om man har stora mängder aktier.

Referenser

[1] L. Weng, *Predict Stock Prices Using RNN: Part 1*, 2017. URL: https://lilianweng.github.io/lil-log/2017/07/08/predict-stock-prices-using-RNN-part-1.html (hämtad 2017-12-11).

[2] A. Karpathy, *The Unreasonable Effectiveness of Recurrent Neural Networks*, 2015. URL: http://karpathy.github.io/2015/05/21/rnn-effectiveness/ (hämtad 2017-12-03).

[3] N. K. Ahmed, A. F. Atiya, N. E. Gayar och H. El-Shishiny, "An Empirical Comparison of Machine Learning Models for Time Series Forecasting", *Econometric Reviews*, årg. 29, nr 5-6, s. 594–621, 2010, ISSN: 0747-4938. DOI: 10.1080/07474938.2010.481556. URL: http://www.tandfonline.com/doi/abs/10.1080/07474938.2010.481556.

[4] C. Renner, J Peinke och R Friedrich, "Evidence of Markov properties of high frequency exchange rate data", *Physica A*, årg. 298, s. 499–520, 2001. URL: www.elsevier.com/locate/physa.

[5] A. Håkansson, *Natural Language Processing (lecture)*, 2017.

[6] Colah, *Understanding LSTM Networks*, 2015. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/ (hämtad 2017-12-11).
[7] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever och R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research*, årg. 15, s. 1929–1958, 2014. URL: https://www.cs.toronto.edu/$\sim$hinton/absps/JMLRdropout.pdf.
[8] F. and others Chollet, *Keras*, 2015. URL: https://github.com/fchollet/keras.
[9] W. Meints, *Monitor progress of your Keras based neural network using Tensorboard*, 2017. URL: https://fizzylogic.nl/2017/05/08/monitor-progress-of-your-keras-based-neural-network-using-tensorboard/ (hämtad 2017-12-11).
[10] Mart, *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*, 2015. URL: https://www.tensorflow.org/.

BILAGA A
FILÖVERSIKT

A. *stock_price.py*

1) Importera data
2) Skala data för att den ska vara enklare att relatera till varandra
3) Skapa sekvenser för test och träning.
4) Bygg modellen
5) Träna modellen
6) Spara den tränade modellen
7) Skapa nya värden med hjälp av modellen
8) Kontrollera när vi enligt modellen borde köpa och sälja
9) Genomför sekvensen med köp och sälj operationen
10) Skriv ut resultaten från förutsäelser, köp och sälj operationer, utveckling av det kapital som förvaltas samt utvecklingen för aktien.

B. *data_processing.py*

1) Delar datan i delar för test och träning
2) Skapar block enligt tidigare beskrivning för träningsdatan
3) Blanda listan med träningsdata
4) Skapar block enligt tidigare beskrivning för testdatan
5) Skillnaden här att steget mellan starten på varje sekvens alltid är 1.
6) Delar blocken så att sista värdet i varje sekvens plockas bort och istället används som det korrekta svaret.
7) Returnerar datan [träning, träning_svar, test, test_svar]

BILAGA B
ANVÄNDA BIBLIOTEK

1) Tensorflow
2) matplotlib
3) numpy
4) pandas
5) keras
6) sklearn