CA400 Technical Specification

Ben Kelly - 15337716
Supervisor - Martin Crane

Abstract

SmartPredict is a Web application trading platform which aims to help users make intelligent decisions about buying and selling of currencies (foreign exchange & cryptocurrency markets). This is achieved through machine learning and chart analysis. The application will provide the user with predictions for the market based on the knowledge it has acquired from past data. Recommending whether the user should buy, sell or hold their specific currency according to its current value. A user is able to view the current market sentiment, review the prediction of the machine learning algorithms and interact with charts of past prices data all within the application. The following technical specification will give a comprehensive review of the development of SmartPredict. It aims to aid understanding of the functionality, implementation strategies and design decisions made in the production and testing phases of the project. The project itself can be found online at http://kellyb45.pythonanywhere.com/smartpredict.

Table of Contents

1. Overview 3
1.1 Introduction 3
1.2 Motivation 3
1.3 Glossary 4
2. General Description 5
2.1 Business Context 5
2.2 Research Conducted 5
3. Design.....6
3.1 System Architecture 6
3.2 Schneiderman's 8 Golden Rules of Interface Design 7
3.3 Context Diagram 10
4. Implementation 11
4.1 Overall Django Application Structure 11
4.2 Crypto AI 11
4.3 Forex AI 13
4.4 Sentiment Analysis 15
4.5 Chart Analysis..... 16
4.6 Bootstrap Front-end..... 17
4.7 RSI and MACD..... 17
5. Problems solved 19
6. Testing 20
7. Future Work..... 21
8. References..... 21

1. Overview

1.1 Introduction

SmartPredict is a Web application trading platform which aims to help users make intelligent decisions on buying and selling of currencies (foreign exchange & cryptocurrency markets). This is achieved through machine learning and chart analysis. The application will provide the user with predictions for the market based on the knowledge it has gained from past and present data. Recommending whether the user should buy, sell or hold their specific currency relative to its current value. A user will be able to trade, view the current market sentiment and review the prediction of the machine learning algorithm all within the application.

Foreign currency exchange (Forex) remains the largest financial market in the world, the amount traded on Forex exceeds all the world's equity markets combined. Traders participating in this market make wagers based on the values of currencies, they profit when their predictions are accurate this is where SmartPredict's prediction element thrives.

The cryptocurrency exchange is something extremely new to the trading environment. It has been the fastest growing method of capital trading in recent history and has captured the imagination of millions of people. Attracting new people to the industry who are perhaps less financially experienced than the typical trader and more software/tech influenced. This is where SmartPredict will appeal to both experienced and novice traders.

1.2 Motivation

I've always had a keen interest in finance as a whole, but when the time came to choose a discipline to study in college I decided that computing would be a better fit for me. When the idea of Bitcoin and other Cryptocurrencies came along it naturally caught my eye, being the product of my two main interests. The technology rapidly developing today is causing a complete shift in the way individuals and corporations finances are managed, I believe this whole industry will maintain shifting towards tech as it evolves and that decentralised cryptocurrencies such as Bitcoin will be the future of banking and finance. These emerging technologies are of huge interest to me and another reason why I chose a project based in this field.

The project as a whole has been a huge learning curve for, when I first began the project I naively assumed I had a good knowledge of the trading world and how currency fluctuation works. This was soon to be proven inaccurate, to discover what users would need to make good trading decisions took a lot of learning and reading which will be outlined in section 2.2 Research. The project as also served as a great opportunity to not only learn more about topics I was interested in but it has given me

a much deeper understanding of AI and Machine Learning which I feel will help me in my future career.

1.3 Glossary

Forex - Foreign currency exchange, a market in which the foreign currencies of the world are traded.

LSTM - Long Short-Term Memory, an extremely powerful algorithm that can classify, cluster and make predictions about data, particularly time series and text.

ARIMA - AutoRegressive Integrated Moving Average (ARIMA). A class of model that captures a suite of different standard temporal structures in time series data.

MACD - Moving Average Convergence and Divergence (MACD). A popular form of analysis utilised by traders worldwide which act as a buy/sell trigger. It can be thought of as an indication of the momentum of a currency.

MVC - The Model-View-Controller (MVC) is an architectural pattern that separates an application into three main logical components: the model, the view, and the controller. Each of these components is built to handle specific development aspects of an application.

Open - Refers to the time at which a particular market opens.

Close - Refers to the time at which a particular market closes.

Neural Network - A series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates.

Overfitting - When the model is too closely aligned to the training data causing artificially high accuracies.

Dropout - A process of setting random nodes to zero in a neural network to prevent overfitting.

2. General Description

2.1 Business Context

SmartPredict's target audience is anyone who has an interest in trading. Whether they are complete novices or experienced traders, they should be able to use SmartPredict from the get-go. My goal is for SmartPredict is to be as intuitive as possible while having complex functionality behind the interface. In essence, this app aims to level the playing field for all traders, so that even beginners can make intelligent trades using SmartPredict's prediction algorithm to make informed decisions. Traders will also be able to learn visually through live graphs about the various currency data.

Generally speaking, the Forex market is highly volatile and auction-based. It is open to investment professionals as well as amateur traders, particularly through online and app-based brokerages specialising in this area. One other important aspect of the Forex market which factors into the SmartPredict experience is the fact that it is open continuously, due to time differences around the world. The combination of the worlds largest financial market, as well as the world's fastest growing one with the aid of a prediction algorithm all in the same place, will be what sets SmartPredict apart from the competition.

As of last year, one in every 7 adults in Ireland made money from trading online. This approximates to nearly 1 million people in Ireland (North and South). This number drastically increases when we compare the number of active traders in other regions particularly Asia (3.2 million) and North America (1.5 million). Amounting to a total of 13.9 million traders worldwide. SmartPredict can, therefore, be used by an extremely large group of people.

2.2 Research Conducted

Django App:

The initial focus for the project was to get the Django app up and running. Although I was comfortable with MVC architecture and python I had zero experience with the Django app framework I had to do a lot of research about how the overall system worked and communicated within itself. Youtube tutorials and the official Django walkthrough were invaluable pieces of information to work from.

AI & Finance:

I had some theoretical knowledge about AI and some algorithms but I had never actually written any code that could be considered intelligent. This was perhaps the most valuable piece of functionality of the project so it was vital that it not only worked but worked well. It is extremely difficult to accurately predict something as temperamental as cryptocurrency and forex. Websites like TowrdsDataScience really helped in this regard. Initially, I assumed that one model (tweaked slightly maybe)

would be accurate enough for both markets. After some reading and trial phases, this was proven wrong.LSTM works really well for the more volatile Bitcoin but for the more stable market of Forex, an ARIMA model is much more accurate and is used widely throughout the prediction world.

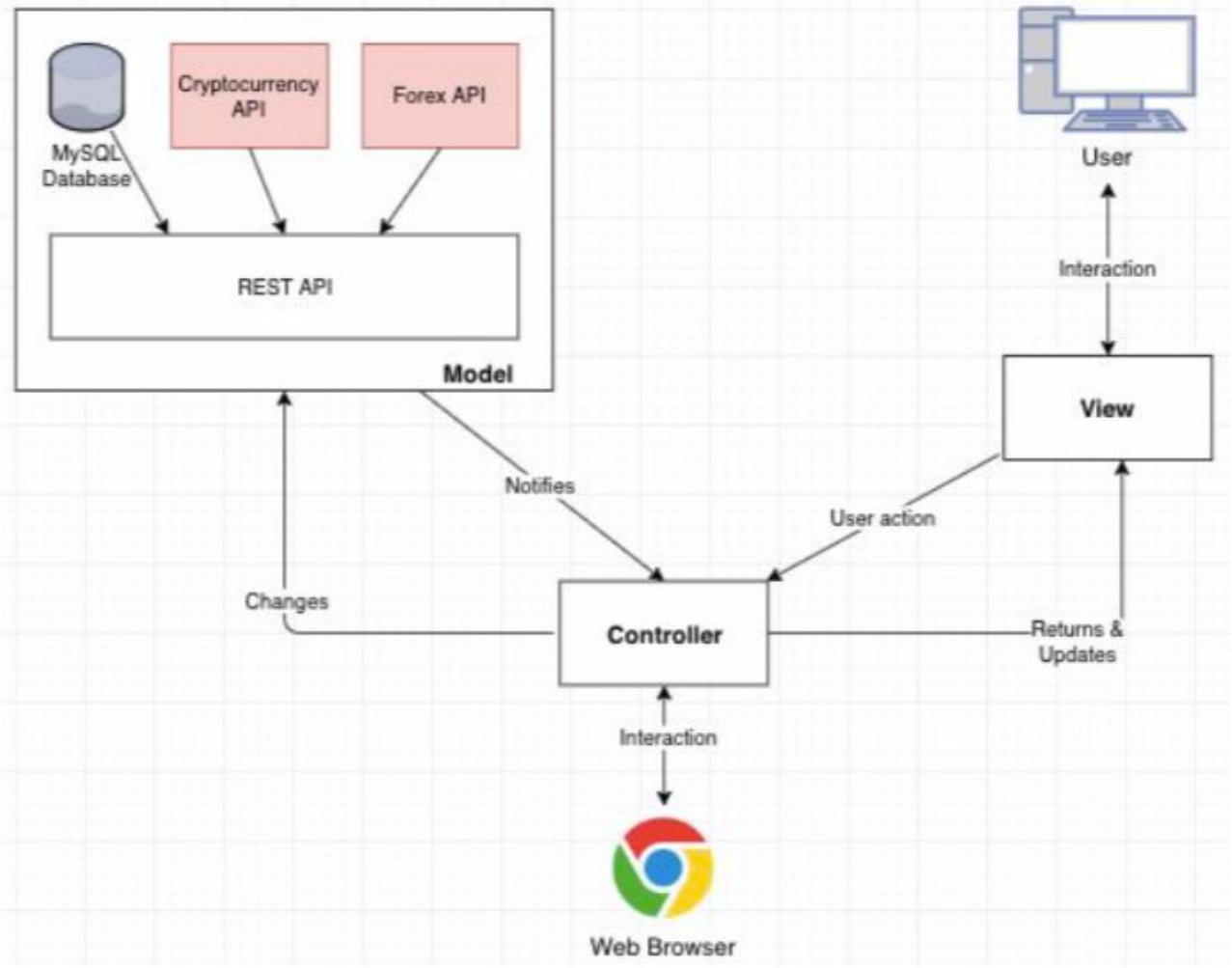
Having the python skills to be able to convert general formulae for RSI & MACD really helped with the chart visualisation on both Dashboards. Some niche functions were needed in these calculations like mean squared error which was imported. The research that went into choosing which algorithm to use was the tricky part, as there are so many different algorithms by which some traders swear by. I tried to keep the charts and data returned to the user as clear and easy to understand as possible while still adding value to the project

Sentiment Analysis:
Sentiment analysis was not something that I initially intended for SmartPredict, however when my project was approved, Tomas Ward recommended that I use it. After some reading on the topic, I realised how essential it was to the user when making trading decisions. It gives a really good insight into how the currency/market is being received by the general public and can have an effect on the price.
See section 9 References for the full list of research papers/websites consulted.

3. Design

3.1 System Architecture

Django uses a variation on the typical MVC(Model View Controller) architecture of Web Apps known as MVT (Model View Template) but other the behaviour can be described as MVC so instead of introducing new terminology for the sake of it we'll stick to MVC.The following system architecture diagram gives a better explanation of how each element interacts with one another than I could do explaining each permutation by text. 3rd party elements are outlined in red.



3.2 Schneiderman's 8 Golden Rules of Interface Design

When developing the User Interface Schneiderman's rules were employed to the best of my ability, examples of which will be listed below.

1. **Strive for consistency** by utilizing familiar icons, colours, menu hierarchy, call-to-actions, and user flows when designing similar situations and sequence of actions. In Smartpredict from the moment the user navigates to the website for the first time, they will be met with the same responsive UI, colour scheme, menu navigation (header & footer) and logo. This was essential as standardizing the way information is conveyed ensures users are able to apply knowledge from one click to another; without the need to learn new representations for the same actions. It also helped decrease the learning curve for new users and ensures once this information is retained they don't need to re-learn another method for doing a similar task.

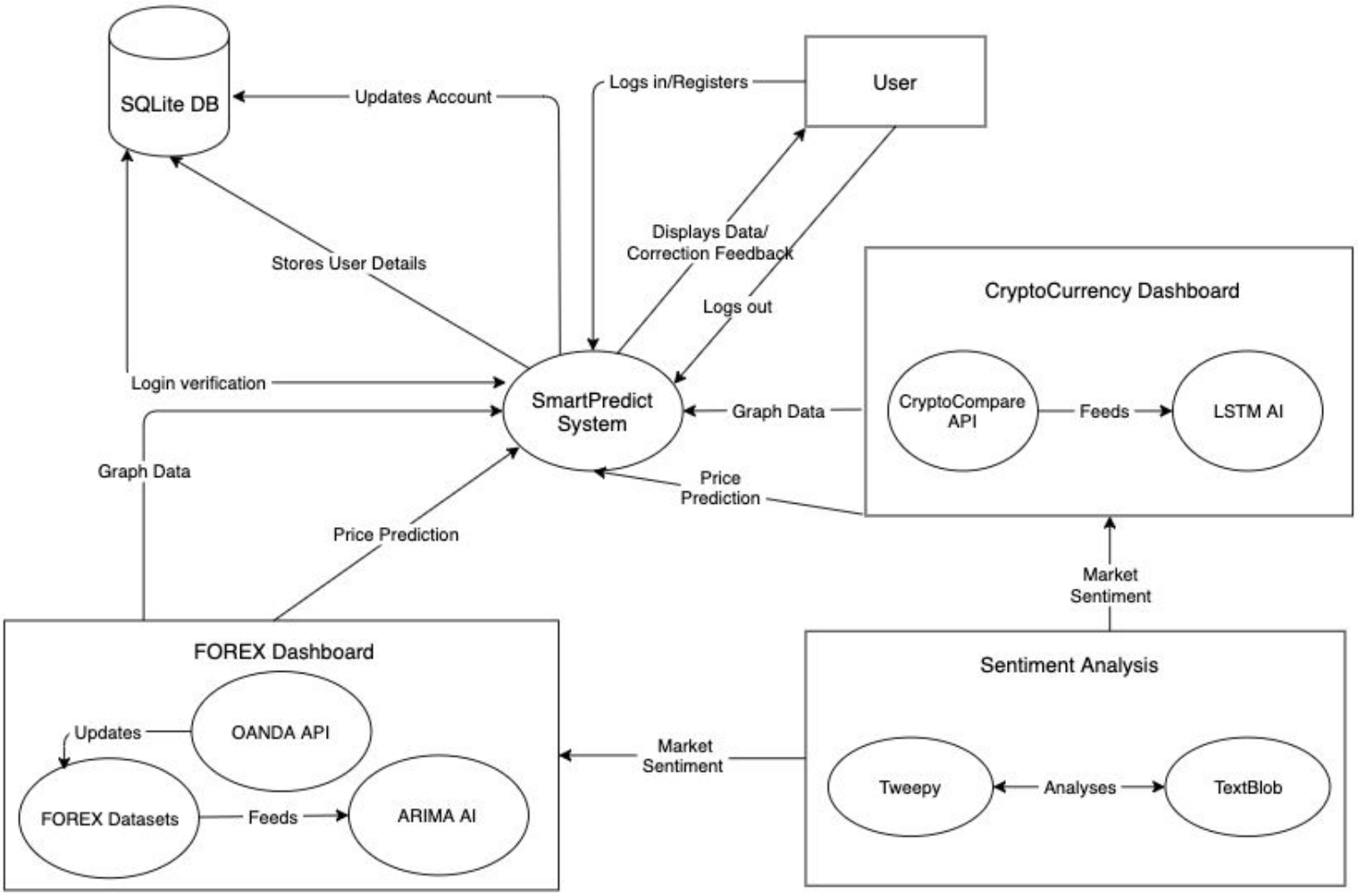
2. **Enable frequent users to use shortcuts**. With increased use comes the demand for quicker methods of completing tasks. In SmartPredict, regular users will stay logged in (provided they don't manually log out) so they don't need to re-login every time they wish to use the app.
3. **Offer informative feedback.** The user should know where they are at and what is going on at all times. For every action, there should be appropriate, human-readable feedback within a reasonable amount of time. Smartpredit's headers navigation emphasises which page the user is on to offer feedback on what the user should be doing. Also when logging in/ registering and some criteria are not met (password length for example) the user will be met with specific instructions on how to fix the issue.
4. **Design dialogue to yield closure**. Don't keep your users guessing. Tell them what their action has led them to. SmartPredict achieves this by giving more information about a graph when the user is hovering over it as shown below.

5. **Offer simple error handling.** No one likes to be told they're wrong, especially your users. Systems should be designed to be as fool-proof as possible, but when unavoidable errors occur, ensure users are provided with simple, intuitive step-by-step instructions to solve the problem as quickly and painlessly as possible. For example, flag the text fields where the users forgot to provide input in an online form.

6. **Permit easy reversal of actions.** Designers should aim to offer users obvious ways to reverse their actions. To undo or go back to the previous page the user only click the back button on their browser. This relieves some anxiety for the user knowing what they do won't be permanent.
7. **Support internal locus of control.** Allow the users to be the initiators of actions. Give users the sense that they are in full control of events occurring in the digital space. Earn their trust as you design the system to behave as they expect. In smartpredict a user can navigate freely within the app, there is no path that needs to be taken in a certain order to achieve a certain goal. The header eliminates the need for multiple in-page links which can hinder the users understanding of how to navigate.
8. **Reduce short-term memory load.** Human attention is limited and we are only capable of maintaining around five items in our short-term memory at one time. Therefore, interfaces should be as simple as possible with proper information hierarchy and choosing recognition over recall. The aim for the UI has always been one of intuition and ease. This was shown in the user testing to be true.

3.3 Context Diagram

The context diagram gives us a great volume of detail in terms of the system itself and the entities it interacts with. For example, the system only interacts with the database when verifying a user's login or storing credentials within it. In contrast to this, the user can interact with the system in a multitude of ways at any instant as seen below.



4. Implementation

4.1 Overall Django Application Structure

The Django app without any of the functionality mentioned below is very simple. In essence, it is just a vessel to facilitate the following.

- The machine learning modules,
- Login and registration,
- Call APIs to collect more data,
- Display this data in a nice way (Bootstrap),
- Be able to display graphs in different languages (JavaScript) and
- Facilitate navigation through paging.

4.2 Crypto AI

The cryptocurrency algorithm uses a Tensorflow backend, this was the simplest and most efficient way to get an LSTM model trained. It is important to note that while developing and testing both algorithms I made a conscious effort to keep 'tinkering' to a minimum, This would cause overfitting of the small dataset I had of 2000 data entries for Bitcoin. The following steps were taken to mitigate the risk of overfitting.

- Normalisation
- Dropout

The algorithm itself is a form of RNN (Recurrent Neural Network) and uses the following libraries

- Numpy
- Keras (Tensorflow API)
- MatPlotLib
- Pandas
- SciKitLearn
- Seaborn

My data comes from the same API which I used in the crypto dashboard, CryptoCompare, but instead, it uses the longer sequence of 2000 price points instead of 100 as used to display the graphical data.

Optimiser: AdamOptimizer: Adaptive Moment Estimation (Adam) keeps separate learning rates for each weight as well as an exponentially decaying average of previous gradients. This combines elements of Momentum and Adagrad together and is fairly memory efficient since it doesn’t keep a history of anything (just the rolling averages). It is reputed to work well for both sparse matrices and noisy data. Adam seems promising for the stock market data. This optimiser was extremely useful when dealing with Bitcoin data which is extremely noisy, and non-recent memory is not as important as it is in forex.

Window length: As mentioned on numerous occasions, Bitcoin is extremely volatile, what happened two weeks ago isn't always relevant today. Thus a small number for window_len was chosen to try to decrease the effects of 'non-relevant' data and try to emphasise what has happened closer to the time i.e within the last 3 days.

Train-Test split: Both AI's use a split of 90:10 when studying for Data Mining in first semester this year I found that this split is a good starting place and experimentation can be used to find a more optimal split, in both cases never happened.

Dropout: As mentioned before this was essential to prevent overfitting so a non-trivial percentage of 25% was used here. This also prevented an NP-complete problem which may have been the case if the model exhaustively examined each path every iteration. It also slows the learning rate of ADAM in our case. To maximise accuracy a combination of two main factors were experimented with. They were also tested using two loss functions, mean absolute error (MAE) and mean squared error (MSE).

- No. of neurons
- No. of epochs

```
Error:0.02540510253273065 lstm_neurons:20 epochs:20 loss:mse
Error:0.022855874390560266 lstm_neurons:20 epochs:50 loss:mse
Error:0.023686172909045726 lstm_neurons:30 epochs:25 loss:mse
Error:0.022373680197217676 lstm_neurons:20 epochs:20 loss:mae
Error:0.021694757455838164 lstm_neurons:20 epochs:100 loss:mae
Error:0.02193876019376793 lstm_neurons:30 epochs:50 loss:mae
Error:0.022392384178562638 lstm_neurons:30 epochs:25 loss:mae
Error:0.021512669211953196 lstm_neurons:20 epochs:30 loss:mae
```

The model is build like this, take note of the arguments mentioned above.

```
def build_lstm_model(input_data, output_size, neurons=20, activ_func='linear',
dropout=0.25, loss='mae', optimizer='adam'):

    model = Sequential()
    model.add(LSTM(neurons, input_shape=(input_data.shape[1], input_data.shape[2])))

    model.add(Dropout(dropout))
    model.add(Dense(units=output_size))
    model.add(Activation(activ_func))

    model.compile(loss=loss, optimizer=optimizer)
    return model
```

The error, no. of neurons, no. of epochs and loss function are written to a file for easy comparison of results. The last 10 elements of the prediction list are also written with the very last value returned to the dashboard for the predicted closing price.

```
with open('pred.txt', 'a') as file:
   file.write("Error:%s" % error)
   file.write(" lstm_neurons:%s" % lstm_neurons)
   file.write(" epochs:%s" % epochs)
   file.write(" loss:%s" % loss + "\n")
   for p in preds[-10:]:
       p = round(p, 2)
       file.write(str(p)+"\n")
my_pred = preds[-1]
my_pred = round(my_pred, 2)
return my_pred
```

4.3 Forex AI

After some reading and experimentation, I discovered that using the LSTM model defined above wouldn't be a good idea for the forex data. This is much less noisy and less volatile than Bitcoin. So an ARIMA model was used. StatsModels had a nice implementation of ARIMA so I decided to use that.

There was much more past data available for forex as it has been around for far longer than Bitcoin and I was also doing four times the amount of work than the crypto AI so I decided instead of calling the API every time I wanted a prediction I would store all of the data into files and when re-running the algorithm at a later date it would automatically check for any updates in the data, append them to the file and get to work. This is done four times in total and four separate text files are stored, the data is written to fx1,2,3,4.txt. With each prediction appended to the end of fx.txt in order to maintain the format.

```
eur_usd =
requests.get('https://www.alphavantage.co/query?function=FX_DAILY&from_symbol=EU
R&to_symbol=USD&outputsize=full&apikey=its_a_secret_shh)
eur_usd = eur_usd.json()
eur_usd_prices = []

l = []
today = datetime.today()
str_today = today.strftime('%Y-%m-%d')
stri = datetime.strptime(str_today, '%Y-%m-%d')
with open('fx1.txt', 'r') as file:
   f = file.readlines()
   day = f[-1]
   day = day[:10]
   day = day.strip("\n")
   day = datetime.strptime(day, '%Y-%m-%d')
   file.close()
i = 0
while stri > day:
   #get difference in days
   today2 = datetime.today() - timedelta(days=i)
```

```
    str_today2 = today2.strftime('%Y-%m-%d')
    if str_today2 in str(eur_usd['Time Series FX (Daily)']):
        obj = eur_usd['Time Series FX (Daily)'][str_today2]['4. close']
        obj = float(obj)
        eur_usd_prices.append(obj)
    li = str_today2 + " , " + str(obj)
    l.append(li)
    i += 1
    stri = datetime.strptime(str_today, '%Y-%m-%d') - timedelta(days=i)
for item in l[::-1]:
    with open('fx1.txt', 'a') as file2:
        file2.write(item)
        file2.write("\n")
```

ARIMA was chosen as it is particularly good at detecting unusual events and estimating the magnitude of their effect, this is particularly useful when attempting to measure the future prices of forex currencies as most f the time they are very steady with little fluctuation but can change drastically due to an unforeseen event.

```
def StartARIMAForecasting(Actual, P, D, Q):
    model = ARIMA(Actual, order=(P, D, Q))
    model_fit = model.fit(disp=0)
    prediction = model_fit.forecast()[0]
    return prediction
```

- P is the number of autoregressive terms. In other words how many days in the past are relevant to predict future days. Just like the crypto AI, the past 3 days are most important to us.
- D is the number of non seasonal differences needed for stationarity. We are predicting on a daily basis here so we set D=1 because we want to subtract the latest value from yesterday's value (predict tomorrow based on today).
- Q is the number of lagged forecast errors in the prediction equation. In our case, we are measuring daily so there is zero lag here.

Thus our values for (P, D, Q) are (3,1,0)

```
for timepoint in range(len(TestData)):
    ActualValue = TestData[timepoint]
    # forecast value
    Prediction = StartARIMAForecasting(Actual, 3, 1, 0)
    Predictions.append(Prediction)
    Actual.append(ActualValue)

# Print MAE to see accurate the model is
pred = Predictions[-1][0]
Error = mean_absolute_error(TestData, Predictions)
print('Test Mean Absolute Error: %.3f' % Error)
```

4.4 Sentiment Analysis

The system uses Tweepy to collect the latest 100 tweets from Twitter. Once authorisation was handled using my Twitter API keys I set to work searching for keywords. Fortuny, cryptocurrencies are very unique words. Below is the Bitcoin sentiment analysis function. Lists are needed in this instance to be able to pass to the chart.js module to display the doughnut chart of the sentiment. TextBlob is then used to determine if the tweet is positive, negative or neutral. I found that the .polarity() function here gave good results but I made a conscious decision after using the method for a while to use the score slightly different to the default method, stating that if a score is between 0.05 and -0.05 to count it as neutral. The lists containing the good, bad and neutral tweets are then passed to the chart.js function to be graphed with the total being maintained so that I can get a better understanding if the tweets are truly positive or negative. If I was just to compare lengths of lists here we would get some inaccuracies. For instance, fifty slightly positive tweets would outweigh forty extremely negative tweets. Which shouldn't be the case, so a running total is maintained to prevent this.

```
api = tweepy.API(auth)
btc_tweets = []
for btc_tweet in tweepy.Cursor(api.search, q='bitcoin', lang='en',
tweet_mode='extended').items(100):
   # Defining Tweets Creators Name
   tweettext = str(btc_tweet.full_text.lower().encode('ascii', errors='ignore'))  # encoding
to get rid of characters that may not be able to be displayed
   btc_tweets.append(tweettext)
btc_sent = 0
btc_good = []
btc_bad = []
btc_neutral = []
for tweet in btc_tweets:
   analysis = TextBlob(tweet)
   btc_score = analysis.sentiment.polarity
   btc_sent += btc_score
   if btc_score > 0.05:
       btc_good.append(btc_score)
   elif btc_score < -0.05:
       btc_bad.append(btc_score)
   else:
       btc_neutral.append(btc_score)
btc_sent = round(btc_sent, 2)
btc_sents = []
btc_sents.append(len(btc_good))
btc_sents.append(len(btc_bad))
btc_sents.append(len(btc_neutral))
```

4.5 Chart Analysis

The charts in SmartPredict are all written in JavaScript using chart.js. This was one of the reasons why Django was chosen as the development environment, as it allows multiple languages to communicate (almost) seamlessly. The JS itself is run within a script in the HTML template for the page. Below we can see the values that are received from the view contained in curly braces. This is how Django passes variables from the View to the Template in a language called Jinja, again this was something completely new to me so this had to be fully understood before I could implement any graphs. Once the data is referenced and in the correct format, it is just a matter of graphing it correctly.

```
var ctx =
document.getElementById('BTCChart').getContext('2d');
 var times = [];
 {% for t in times %}
 times.push('{{t}}');
 {%endfor%}
 var chart = new Chart(ctx, {
    // The type of chart we want to create
    type: 'line',

    // The data for our dataset
    data: {
        labels: times,
        datasets: [{
            label: 'Bitcoin Price €',
            backgroundColor: 'rgb(255, 99, 132)',
            borderColor: 'rgb(255, 99, 132)',
            borderWidth: 1,
            fill: false,
            data: {{btc_prices}}
        }]
    },

    // Configuration options go here
    options: {
      scales: {
          yAxes: [{
              scaleLabel: {
                  display: true,
                  labelString: 'Price',
                  fontSize: 20
              }
          }],
          xAxes: [{
              scaleLabel: {
                  display: true,
                  labelString: 'Date',
```

```
                    fontSize: 20
                },
            }]
        }
    }
});
```

4.6 Bootstrap Front-end

To make it easier for users to understand what each piece of functionality means I thought it would be a nice touch if I could use the Bootstrap front-end to make certain texts and figures change colour on the indication of their meaning. This can be seen in the table for RSI of a currency or the piece of text accompanying the sentiment analysis of a market. The number or text will turn:

- Green if the number is deemed to be positive/strong.
- Yellow if the number is deemed to be neutral.
- Red if the number is to be deemed negative not to be confused with a number with a value that is less than zero.

These colours are simply to indicate performance, not the specific number shown.

```
<td>{% if eth_rsi > 60 %}
<p class="text-success">{{eth_rsi}}</p>
{% elif eth_rsi < 50 %}
<p class="text-danger">{{eth_rsi}}</p>
{% else %}
<p class="text-warning">{{eth_rsi}}</p>
{%endif%}</td>
<td>{% if eth_sent > 3 %}
<p class="text-success">{{eth_sent}}</p>
{% elif eth_sent < -0.5 %}
<p class="text-danger">{{eth_sent}}</p>
{% else %}
<p class="text-warning">{{eth_sent}}</p>
{%endif%}</td>
```

4.7 RSI and MACD

RSI and MACD are two hugely popular methods of calculating momentum and it was vitral for the project to calculate them.

RSI is calculated over a 14 day period and indicates momentum of a currency. This number is always between 0-100 but usually between 30-70, the closer to 70 the better and conversely the closer to 30 the worse.

```
def rsiFunc(stocks, n=14):
    deltas = np.diff(stocks)
    seed = deltas[:n + 1]
    up = seed[seed >= 0].sum() / n
    down = -seed[seed < 0].sum() / n
    rs = up / down
    rsi = np.zeros_like(stocks)
    rsi[:n] = 100. - 100. / (1. + rs)

    for i in range(n, len(stocks)):
        delta = deltas[i - 1]  # cause the diff is 1 shorter
        if delta > 0:
            upval = delta
            downval = 0.
        else:
            upval = 0.
            downval = -delta

        up = (up * (n - 1) + upval) / n
        down = (down * (n - 1) + downval) / n
        rs = up / down

        rsi[i] = 100. - 100. / (1. + rs)
    return rsi[-1]
```

The MACD line is calculated by subtracting the slow (26 day period) from the fast (12 day period) with the signal line being generated over 9 days.

```
def computeMACD(x, slow=26, fast=12):
    emaslow = ExpMovingAverage(x, slow)
    emafast = ExpMovingAverage(x, fast)
    btc_macd_dif = []
    j = 0
    while j < len(emafast):
        dif = emafast[j] - emaslow[j]
        btc_macd_dif.append(dif)
        j += 1
    btc_macd_cotrol = ExpMovingAverage(btc_macd_dif, 9)
    context = {'btc_macd_dif': btc_macd_dif, 'btc_macd_cotrol': btc_macd_cotrol}
    return context
```

5. Problems solved

During the course of the project, a multitude of problems were encountered. Here is a list of the major ones. All of the problems listed below and more insight into the development process is documented in the blog of the project on Gitlab.

1. **AWS charging for their services.** After connecting and setting up an Amazon Web Services Database instance to the app I ran into issues with Amazon charging me for using their RDS system. They charged me $10 for exceeding their free usage limit, I figured that $10 every month with little usage of the database would be unsustainable for the duration of the project, particularly when it came to the more database intensive periods. I then faced the problem of creating a new Database instance and re-connecting it to the app. This was solved by connecting a local SQLite database to the project. Which was slightly disappointing as I had hoped to gain insights such as the demographics of the app via the use of a cloud-based system such as AWS.
2. Breaking up the **JSON** object into smaller pieces of usable information and formatting it so it is readable, some reading up on JSON had to be done for this to work. The elements were then returned as green if positive, red if deemed negative and yellow if deemed neutral.
3. As mentioned in the research section, I had virtually no experience with writing or using **Machine Learning algorithms**. A lot of reading and learning was necessary to get a grasp of how to develop these types of programmes. Tensorflow and StatsModels were obviously a huge help with fitting and training the data as well as sites like stackoverflow and towardsdatascience. The solution can be seen in my implementation of both algorithms.
4. **Static file collection on pythonanywhere** version of the app. As the two apps (local & online) have two different link roots, they handle static files differently. This was preventing paging and displaying of images. This was fixed after some reading up on how pythonanywhere deals with static files along with Django as it is slightly different than pure Django (localhost).
5. **Trading.** Unfortunately one element of the functional specification which I was not able to implement was the trading of currencies. I had initially intended on using API keys so that the users could just copy and paste them into the app but this proved unsecure and actually got the live implementation of the application taken down on Pythonanywhere. My alternative to this was using OAuth authentication to allow the user to trade but this proved extremely difficult to implement and I didn't manage to accomplish it in the timespan for this project.
6. **Passing values to chart.js so that they can be read.** The problem was when I pass the python list to the javascript implementation of Chart.js is gives out when the list is anything but integers. Initially, I had datestamps for the given days' prices but the user wouldn't be able to read this, the workaround

involved converting the timestamp to readable dates using the strftime() method, converting this to a string, passing it to the javascript chart function and then iterating through this list on the front end. You will notice a try and except clause here,. This was because unfortunately the dataset was not 100% complete, some days are missing, so instead of failing completely when it encounters a day not in the jSON object it skips it and moves on.

Python views.py

```
days = []
i = 0
for element in eur_usd['Time Series FX (Daily)']:
   today = datetime.today() - timedelta(days=i)
   today = today.strftime('%Y-%m-%d')
   try:
       if today in str(eur_usd['Time Series FX
(Daily)']):
           days.append(today)
```

Jinja and JavaScript dashboard.html

```
{% for d in days %}
times.push('{{d}}');
{%endfor%}
times = times.reverse();
```

6. Testing

The following testing types were carried out as part of the results analysis for the project

- User Testing
- Unit & Coverage Testing
- AI Testing
- Environment & Usability Testing
- Lint testing

Please see the testing document for further information about the testing of SmartPredict.

7. Future Work

Overall I am extremely happy with the project as a whole. It embodies a huge amount of time and effort put in by myself over a short time period of two semesters I feel like I have accomplished a lot. It has been extremely difficult to manage this project while studying for six other modules running concurrently to it but I have managed to survive (just). For future developments of the project I would propose the following.

- Get the trading functionality fully finished through an OAuth connection.
- Predict more currencies particularly on the crypto side perhaps using different machine learning algorithms.
- Use smoothing to get more accurate sentiment analysis perhaps Exponential weighted Moving Average in predict.py could be used with a higher weighting on bad news and forgetting events that happened more than a few days ago.

8. References

Django:
https://docs.djangoproject.com/en/2.1
https://www.youtube.com/user/Max204204204
https://simpleisbetterthancomplex.com/tutorial/2018/02/03/how-to-use-restful-apis-with-django.html
https://wsvincent.com/
https://www.youtube.com/watch?v=-oQvMHpKkms

AI & Finance:
https://www.youtube.com/user/sentdex
https://www.investopedia.com/virtual-currency-4427740
https://www.dublincity.ie/councilmeetings/documents/s13720/Report%2030-2017%20Impact%20of%20Trading%20Online%20-%20Stephen%20Brennan.pdf
https://stackoverflow.com/
https://www.kaggle.com/sumi25/understand-arima-and-tune-p-d-q
https://min-api.cryptocompare.com/documentation?key=Toplists&cat=TopTotalVolumeEndpointFull&api_key=5fd159fba0ac5f6c57a7408f616c83e708847738a64f5a0d80ce1d21d173aafc
https://towardsdatascience.com/
http://colah.github.io/posts/2015-08-Understanding-LSTMs/
http://cs229.stanford.edu/proj2017/final-reports/5212256.pdf

General:
https://realpython.com/testing-in-django-part-1-best-practices-and-examples/
https://getbootstrap.com/docs/4.3/components/media-object/
https://www.interaction-design.org/literature/article/shneiderman-s-eight-golden-rules-will-help-you-design-better-interfaces