Solent University

Faculty of Business, Law and Digital Technologies

COM 624 AE1 Machine Learning Assessment Report

Author : **Q15747930(Danial Agha)**

Course Title : **Digital and Technology Solutions**

Module Leader : **Bacha Rehman**

Date : **12/11/2023**

Table of Contents

List of Figures

Introduction

Within this report I display and document the learning that I have gained throughout the module and whilst undertaking this assessment. I will discuss within this report how I have completed all 7 tasks and explain what libraries and machine learning algorithms I have used to complete that specific task.

I will describe and go into detail my learning and knowledge regarding the machine learning algorithms that I have utilised to assist me in completing the tasks for this Financial Stock Analysis Software Artefact Solution.

For this software solution that I created, I implemented the use of a GitHub repository to store my work online and see the incremental commit changes I have made whilst creating this system.

Financial Stock Analysis Software Artefact Solution GitHub Repository:
https://github.com/DAghaSolent/COM624Assessment/

I also recorded a demonstration of my software solution in action:
https://www.youtube.com/watch?v=17pJkSaNEOY&ab_channel=DanialAgha

System Development

My choice of tech stack when developing this this system was python, this was due to python's extensive support for various libraries and modules, specifically tailored for creating software solutions in the realm of Machine Learning and Data Science. Figure 1 below displays all the libraries and modules utilised in creating my software solution.

```
import yfinance as yf
from datetime import datetime, timedelta
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
import seaborn as sns
from prophet import Prophet
from prophet.plot import plot_plotly
from sklearn.preprocessing import MinMaxScaler
import numpy as np
from tensorflow.keras.layers import LSTM
from tensorflow.keras.layers import Dense
from tensorflow.keras.models import Sequential
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import mean_squared_error
import math
import warnings
import streamlit as st
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from matplotlib.dates import date2num
import matplotlib.dates as mdates
```

Figure 1 Imported Libraries and Modules for Software Solution

To write the code to build this system I utilised Pycharm as my choice of IDE environment due to my familiarity of using it in the past compared to another python supported IDE's. As mentioned earlier I used GitHub to store my online codebase for my software solution, I utilised Git as a suitable use for a Version Control System to be able to track the incremental changes and additions that I made to the software solution.

System Overview

There are two key files for this system, the first key file is the main.py file, this file is crucial to the whole system as it houses all the code and Machine Learning algorithms that were implemented by me to complete the task requirements laid out in the brief. When designing and writing out this main application, I decided to write my code for my software solution modular within my main.py file, any task requirement code or machine learning algorithm has been made into a function, two key reasons for this were, the first reason was it was easier and faster for me to test as I would only have to test the function itself and not the whole main.py file, the other reason was seamless interaction with my GUI, the GUI file that I created has the ability to call upon one of these modular functions on the main.py with ease depending on the users choice. Figures 2 and 3 are examples of my modular code structure for some of my functions.

```
def fb_prophet():
    # Resetting and clearing the data to be processed for the Facebook
Prophet Method
    selected_stocks.reset_index(inplace=True)
    selected_stocks_Date = selected_stocks['Date']

    for stock in selected_stocks.iloc[:, 1:]:
        prophet = Prophet(
            daily_seasonality=True,
            yearly_seasonality=True,
            weekly_seasonality=True,
            changepoint_prior_scale=0.05,
            seasonality_prior_scale=10.0
        )

        # Creating a new DataFrame with the required columns for Facebook
Prophet Prediction
        new_prophetDF = pd.DataFrame({'ds': selected_stocks_Date, 'y':
selected_stocks[stock]})
        prophet.fit(new_prophetDF)

        # Creating a DataFrame to be used for the prediction
        future = prophet.make_future_dataframe(periods=365)

        # Passing the future DataFrame to generate a forecast prediction
for my selected stocks.
        forecast = prophet.predict(future)

        # Plot the predictions that were made by Facebook Prophet Market
prediction
        fig = plot_plotly(prophet, forecast)
        fig.update_layout(xaxis_title="Dates", yaxis_title="Stock Prices",
title_text=f"Facebook Prophet Prediction for {stock}")
        st.plotly_chart(fig)
        with st.expander(f"**Click here to see FB Prophet's Predicted
Adjusted Close Prices for: {stock}**"):
            st.write(forecast[['ds', 'yhat']].rename(columns={'ds': 'Date',
'yhat': 'Adjusted Close Prices'}))
```

Figure 2 Showcase of fb_prophet function built in a modular way.

```
def time_series_plots_for_my_selected_stocks():
    # Creating and displaying a chart with a historical view of Adjusted
Close prices for all my selected stocks.
    plt.figure(figsize=(12, 10))

    for stock in selected_stocks:
        plt.plot(selected_stocks.index, selected_stocks[stock],
label=stock)

    plt.title("Time Series Plot of Adjusted Close Prices for my selected
stocks")
    plt.xlabel("Date")
    plt.ylabel("Adjusted Close Prices")
    plt.legend()
    st.subheader("Time Series Analysis of my selected stocks")
    st.write("Below is a Time Series Visual Analysis where I'm displaying
and comparing Adjusted Close Prices for my "
             "selected stocks in the span of one year. Within this Time
Series Analysis we can see that BKNG's Adjusted"
             " Close Prices is far higher in value than the other 3
selected stocks which are 'AMD', 'NVDA' and 'ORLY'")
    st.pyplot()
```

Figure 3 Showcase of my time_series_plots function built in a modular way.

The other main file as mentioned earlier is my GUI file called “streamlitGUI.py”. This file is a Graphical User Interface which handles all the user interactions with the user and my application. With this implementation a user can navigate around my application and see data analytics in real time provided by the machine learning loaded on my web application.

My application utilises and open-source Python Library called Streamlit which I used to create my GUI for my application, it is a useful tool in building interactive applications for Machine Learning and Data Science.

Task Requirements

Task 1 Retrieving the Data:

The first task as set out in the assessment brief, was to gather the suitable data to perform financial stock analysis operations on this data for the later task requirements within this project. For this project I was given a list of the Nasdaq 100 companies list, from there I had to utilise a script or library to download at least 1 year stock data for the Nasdaq 100 companies. To achieve this task, I built my own custom script that downloads stock data within a 1 year period with the use of a python library called yfinance, which allowed me to conveniently download the required Nasdaq 100 stock data from Yahoo Finance. Figure 4 showcases the code I wrote to retrieve the Nasdaq 100 companies stock data.

```
# Nasdaq 100 companies stored in a list below called tickers
tickers = ['AAPL', 'MSFT', 'AMZN', 'NVDA', 'META', 'AVGO', 'GOOGL', 'GOOG',
'TSLA', 'ADBE', 'COST', 'PEP', 'NFLX', 'AMD'
           , 'CSCO', 'INTC', 'TMUS', 'CMCSA', 'INTU', 'QCOM', 'AMGN',
'TXN', 'HON', 'AMAT', 'SBUX', 'ISRG', 'BKNG',
           'MDLZ', 'LRCX', 'ADP', 'GILD', 'ADI', 'VRTX', 'REGN', 'MU',
'SNPS', 'PANW', 'PDD', 'MELI', 'KLAC', 'CDNS',
           'CSX', 'MAR', 'PYPL', 'CHTR', 'ASML', 'ORLY', 'MNST', 'CTAS',
'ABNB', 'LULU', 'NXPI', 'WDAY', 'CPRT', 'MRVL',
           'PCAR', 'CRWD', 'KDP', 'MCHP', 'ROST', 'ODFL', 'DXCM', 'ADSK',
'KHC', 'PAYX', 'FTNT', 'AEP', 'SGEN', 'CEG',
           'IDXX', 'EXC', 'AZN', 'EA', 'CTSH', 'FAST', 'VRSK', 'CSGP',
'BKR', 'DDOG', 'BIIB', 'XEL', 'GFS',
           'TTD', 'ON', 'MRNA', 'ZS', 'TEAM', 'FANG', 'WBD', 'ANSS',
'DLTR', 'EBAY', 'SIRI', 'WBA', 'ALGN', 'ZM', 'ILMN'
           , 'ENPH', 'JD', 'LCID']

# Creating variables to be used to set dates to download data within a 1-
year timeframe.
end_date = datetime(2023, 12, 26) # Hardcoding the date as I am getting
null errors from a specific stock after 26th Dec
start_date = end_date - timedelta(365)

# Empty dataframe which will be used to store the Adjusted close values for
each Nasdaq 100 company that is stored in
# the tickers list.
adjClose_data = pd.DataFrame()

# Loop through each ticker within the tickers list and download 1 year
adjusted close prices for that individual ticker,
# once obtained put that information in the new data frame that I created
above
for ticker in tickers:
    data = yf.download(ticker, start=start_date, end=end_date)
    adjClose data[ticker] = data['Adj Close']
```

Figure 4 Code that I wrote to download Nasdaq 100 companies Adjusted Close Prices within a 1-year time period.

Figure 4 also has comments that describe what that section of code within the code snippet does.

To keep the data consistent for my solution and due to stock de-listing errors as well I decided to stop the data retrieval on the 26th of December 2023, however I am still retrieving 1 year stock data (26/12/22 – 26/12/23) that I am using for future stock analysis operations with my software solution. Even though I am not providing real up to date data, this is a perfect opportunity to showcase and compare my solution in analysing stock data with real world stock data.

Task 2 A-C Dimension Reduction:

My data has successfully been retrieved with the code that I have written in Figure 4. The current shape of the dataset that I have retrieved holds 100 rows which are my Nasdaq 100 companies and 250 columns which represent the stock dates. To find out the shape of the dataset I utilise the code snippet in Figure 5.

```
st.markdown("Here is the output from the terminal running the code above:
**Before PCA reduction the shape of the "
            f"data frame is{transposed_adjClose_data.shape}**")
```

Figure 5 Code that I wrote to find out the shape of my dataset.

For Task 2A to Task 2C within the assessment brief, is to use a dimensionality reduction algorithm to reduce the columns from the current shape of the downloaded retrieved dataset, in my case I have 250 columns which needed to be reduced to 10 columns. To reduce the columns within my dataset I utilised PCA (Principal Component Analysis).

What is PCA (Principal Component Analysis)?

The columns within a dataset can be represented as dimensions, using a machine learning algorithm on a dataset with a vast number of dimensions can impact the output performance from the machine learning model (Brownlee 2020).

We can improve the performance by utilising PCA to transform and reduce the dimensions within a dataset but still retain most relevant information within the dataset, because the data has been reduced with PCA it makes analysing data much easier and faster for our machine learning algorithms (Jaadi 2023).

PCA simplifies and reduces the dataset by identifying patterns or trends within the data, it creates new artificial variables based of these patterns or trends, known as principal components (Pal 2018).

PCA (Principal Component Analysis) Code Implementation

In figure 6 I utilise PCA reduction algorithm to reduce the columns from 250 to 10 columns and complete the task requirements for 2A to 2C.

```
scaler = StandardScaler()
scaled_data = scaler.fit_transform(transposed_adjClose_data)

# Reducing the data
pca = PCA(n_components=10)
pca_reduced_data = pca.fit_transform(scaled_data)
explained_variance = pca.explained_variance_ratio_

st.markdown("Here is the output from the terminal running the code above:
**After PCA reduction the shape of the "
            f"data frame is{pca_reduced_data.shape}**")
```

Figure 6 PCA Reduction Code

Within Figure 6 I first had to standardise my dataset by scaling the original data to ensure standardization throughout the dataset, this is a crucial step for PCA reduction algorithm. I then create an instance of the PCA class and pass it the number of components that I am intending to reduce the data by, which in this instance is 10 components. I then create a new variable called "pca_reduced_data" and call the "fit_transform method from the PCA class to reduce the data to 10 components, thus reducing the data. Finally, just like in Figure 5 I print out the shape of the "pca_reduced_data" to verify that the PCA reduction was successful.

Task 2 D-H Clustering:

For Task 2D to Task 2H, for these task requirements set out in the brief, I must use any clustering algorithm to group the stocks together, I must group the stocks into 4 cluster groups, from there I will have 4 cluster groups of stock I must select one stock from each cluster group. From the 4 stocks that I have selected, I will then perform analysis operations as required in the assessment brief on my selected stocks. To create these cluster groups I utilised a clustering algorithm, the cluster algorithm that I chose was K-Means cluster.

What is K-Means Cluster?

Clustering is a type of unsupervised learning technique, plays a crucial role in identifying patterns or specific traits within unlabelled data and groups them into clusters depending on similar characteristics within each piece of data (Ali 2022). K-Means which is the clustering algorithm of choice to assist me in completing Task D to Task H, is a simple and popular centroid-based clustering algorithm, it calculates the distance between each data node and a centroid to be able to assign it to the cluster group (Sharma 2023). When implementing K-Means we must pass it the number of cluster groups we want to group the data by, in Figure 7 we look to group the data by 4 clusters reason why “n_clusters=4”, these clusters will act as our centroids meaning there will be 4 randomly placed centroids, the K-Means algorithm will then iteratively calculate the distance between each data node and the 4 centroids, assigning each data node to the closest centroid until all data nodes have been assigned to a cluster thus forming the 4 cluster groups outlined in the assessment brief.

K-Means Cluster Code Implementation

The first part of Task 2 was to reduce the columns by using PCA reduction which is shown in figure 6, The second part of Task 2 was to group the stocks into 4 clusters and then select a stock out of each cluster group to have 4 selected stocks.

```
# Data preprocessing to only use the 10 PCA reduced columns and ignore the
ticker names
pca_reduced_data_numeric_values = pca_reduced_data_dataFrame.iloc[:, 0:10]

# Kmeans Clustering the stocks into 4 clusters
kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
cluster_labels = kmeans.fit_predict(pca_reduced_data_numeric_values)

# Creating a new dataframe to visualise which stock tickers represent in
which cluster group number.
kmeans_clustering_results_df = pd.DataFrame({'Ticker': tickers, 'Assigned
Cluster': cluster_labels})

# Exporting the tickers and their Assigned Cluster label to better
visualise the clusters and which ticker is assigned
# to which cluster.
cluster_csvfilepath = r'C:\Users\Danny\Documents\Uni Solent Work\Year
3\COM624 Machine Learning\COM624 Assessment\kmeans_clustering_results.csv'
kmeans_clustering_results_df.to_csv(cluster_csvfilepath, index=False)
```

Figure 7 K-Means clustering code implementation

In Figure 7 you can see that I create an instance of the K-Means class where I pass within the parameter the number of clusters that I require to complete this task requirement. I export the clustering results as a csv and create a new data frame that I will utilise in Figure 8 just to give me better visibility of the clusters to verify that the task requirements have been completed.

```
# Cluster Lists for better visualisation in terminal and front end GUI
solution.
    cluster0 = []
    cluster1 = []
    cluster2 = []
    cluster3 = []

    # Appending to specific list depending on the Assigned Cluster Number
they have been assigned by KMeans Clustering.
    for index, row in kmeans_clustering_results_df.iterrows():
        if row['Assigned Cluster'] == 0:
            cluster0.append(row['Ticker'])
        elif row['Assigned Cluster'] == 1:
            cluster1.append(row['Ticker'])
        elif row['Assigned Cluster'] == 2:
            cluster2.append(row['Ticker'])
        elif row['Assigned Cluster'] == 3:
            cluster3.append(row['Ticker'])

    st.subheader("Clustering with KMeans")
    st.write("To group the stocks together into 4 separate groups I
utilised the clustering algorithm KMeans to group"
             " the stocks together into 4 separate cluster groups. The code
snippet below is how I utilised KMeans "
             "clustering algorithm to group the stocks into 4 separate
cluster groups.")
    st.code("""
    kmeans = KMeans(n_clusters=4, init='k-means++', random_state=42)
    cluster_labels = kmeans.fit_predict(pca_reduced_data_numeric_values)
    """)

    st.write("Below are the cluster groups and what stocks belong to which
cluster group")
    tab0, tab1, tab2, tab3 = st.tabs(["Cluster Group 0", "Cluster Group 1",
"Cluster Group 2", "Cluster Group 3"])
    # Displaying Cluster Lists
    tab0.write(cluster0)
    tab1.write(cluster1)
    tab2.write(cluster2)
    tab3.write(cluster3)
```

Figure 8 Code written for better visibility for confirming completion of task and GUI implementation.

In figure 8 for better visibility, I decided to make a list for each of the cluster groups and display them to my GUI implementation just for better visibility and verifying that I completed the task. After Figure 7 and 8 code implementation I was able to see the 4 cluster groups, from there I make my selection for the 4 stocks in each group that I will use for stock analysis.

In Figure 9 I state what my selected stocks are and make a data frame to store my selected stocks data that I will use for future stock analysis. I then utilise Yahoo finance to download my selected stock data and add it to the selected_stocks data frame.

```
# Empty dataframe to store the Adjusted Close values for my selected
stocks.
# My selected stocks are [NVDA, AMD, BKNG, ORLY]
selected_stocks = pd.DataFrame()

for ticker in tickers:
    if ticker in ('NVDA', 'AMD', 'BKNG', 'ORLY'):
        selected_stock_data = yf.download(ticker, start=start_date,
end=end_date)
        selected_stocks[ticker] = selected_stock_data['Adj Close']
```

Figure 9 Retrieving my selected stocks data.

Task 3 Present Top-10 Highly Correlated (Positive and Negative) for my chosen stocks:

As stated in the assessment brief for Task 3, I was tasked at presenting the top 10 highly correlated (positive and negative for my chosen stocks). To achieve this task there was no need for any machine learning algorithms, simply just used the ".corr()" function on my selected stocks to get the correlation for my selected stocks and then did the same for the whole data set stocks and finally make a comparison to get the top 10 positive/negative correlation within my stocks. Figure 10 showcases the code to get these correlation values.

```
# Obtain the correlation info for my selected stocks [NVDA, AMD, BKNG,
ORLY].
selected_stocks_correlated = selected_stocks.corr()

# Obtain the correlation info for the whole dataset stocks to compare and
correlate against my selected stocks
adjClose_data_correlated = adjClose_data.corr()
```

Figure 10 Code to retrieve correlation for my stocks and whole data stock correlation.

```
def top10_positive_negative_correlation():
    st.write("In this page I have displayed the top 10 positive and
negative correlations for each of my selected stock"
             " against the whole nasdaq 100 stocks in a heatmap view. At
the bottom of each heatmap there is an "
             "expander on the page click it to view the raw data of the
positive/negative correlation for each of my "
             "selected stocks.")

    st.write("**My Selected Stocks are [NVDA, AMD, BKNG, ORLY]**")

    # Looping through each stock from my selected stocks and displaying the
stock and their top 10 positive/negative
    # correlations from the entire dataset.
    for stock in selected_stocks:
        print(f"Top 10 Positive Correlations with {stock}:")
        st.subheader(f"Top 10 Positive Correlations with {stock}:")
        top10_positive_correlations_with_stock =
adjClose_data_correlated[stock].sort_values(ascending=False).head(11)[1:]
        print(top10_positive_correlations_with_stock)

        # Converting the top10_positive_correlations_with_stock to a
DataFrame, so that I can plot the positive correlation
        # between my selected stocks that are positively correlated against
stocks from the whole dataset.
        top10_positive_correlations_with_stock_df =
pd.DataFrame(top10_positive_correlations_with_stock, columns=[stock])

        # Creating and displaying the heatmap off the positive correlations
for my selected stocks against the stocks from
        # the whole dataset.
        plt.figure(figsize=(10, 8))
        sns.heatmap(top10_positive_correlations_with_stock_df, annot=True,
cmap='coolwarm')
        plt.title(f"Top 10 Positive Correlations with {stock}")
        st.pyplot()
        with st.expander(f"**Click here to see raw data of the Top 10
Positive Correlations with {stock}**"):
            st.write(top10_positive_correlations_with_stock)
```

Figure 11 Code that retrieves top 10 positive correlation stocks against my selected stocks.

```
print(f"Top 10 Negative Correlations with {stock}:")
st.subheader(f"Top 10 Negative Correlations with {stock}:")
top10_negative_correlations_with_stock =
adjClose_data_correlated[stock].sort_values().head(10)
print(top10_negative_correlations_with_stock)

# Converting the top10_negative_correlations_with_stock to a DataFrame, so
that I can plot the negative correlation
# between my selected stocks that are negatively correlated against stocks
from the whole dataset.
top10_negative_correlations_with_stock_df =
pd.DataFrame(top10_negative_correlations_with_stock, columns=[stock])

# Creating and displaying the heatmap off the negative correlations for my
selected stocks against the stocks from
# the whole dataset.
plt.figure(figsize=(10, 8))
sns.heatmap(top10_negative_correlations_with_stock_df, annot=True,
cmap='coolwarm')
st.pyplot()
with st.expander(f"**Click here to see raw data of the Top 10 Negative
Correlations with {stock}**"):
    st.write(top10_negative_correlations_with_stock)
```

Figure 12 Code that retrieves top 10 negative correlation stocks against my selected stocks.

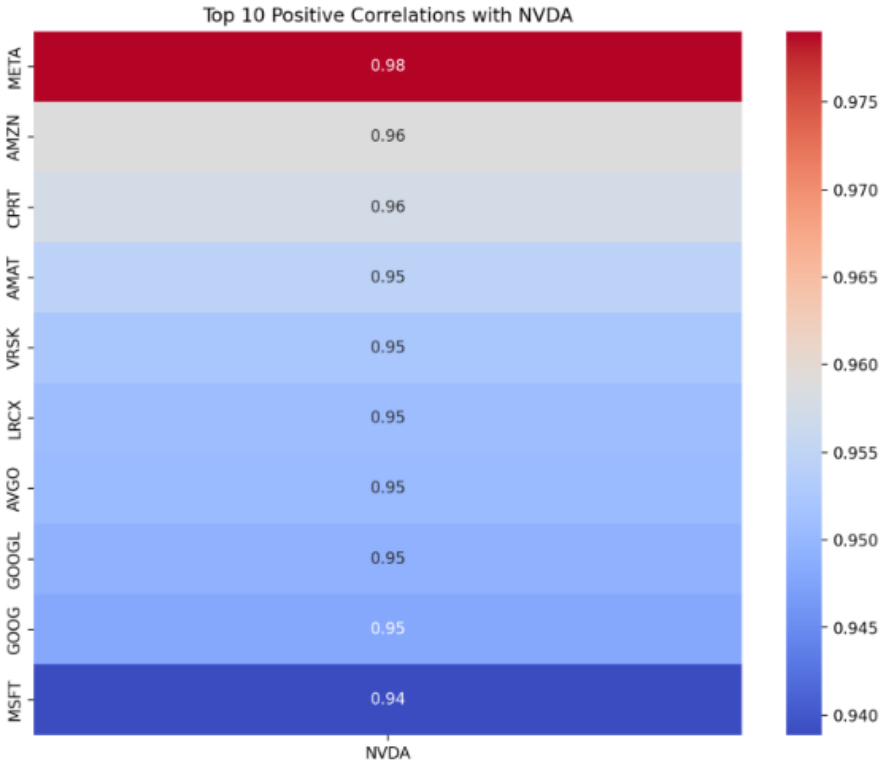
The function in Figures 11 and 12 not only just retrieves the top 10 positive/negative correlations of my stock, with the use of the seaborn library I am able to create a visual heatmap for the correlation index as an EDA (Exploratory Data Analysis) for better visualisation and analysing the top 10 positive and negative correlations for my selected stocks. In Figures 13 and 14 I showcase the positive and negative correlation heatmap and the raw data that is presented to the user on my application.

Task 3: Correlation Analysis

In this page I have displayed the top 10 positive and negative correlations for each of my selected stock against the whole nasdaq 100 stocks in a heatmap view. At the bottom of each heatmap there is an expander on the page click it to view the raw data of the positive/negative correlation for each of my selected stocks.

My Selected Stocks are [NVDA, AMD, BKNG, ORLY]

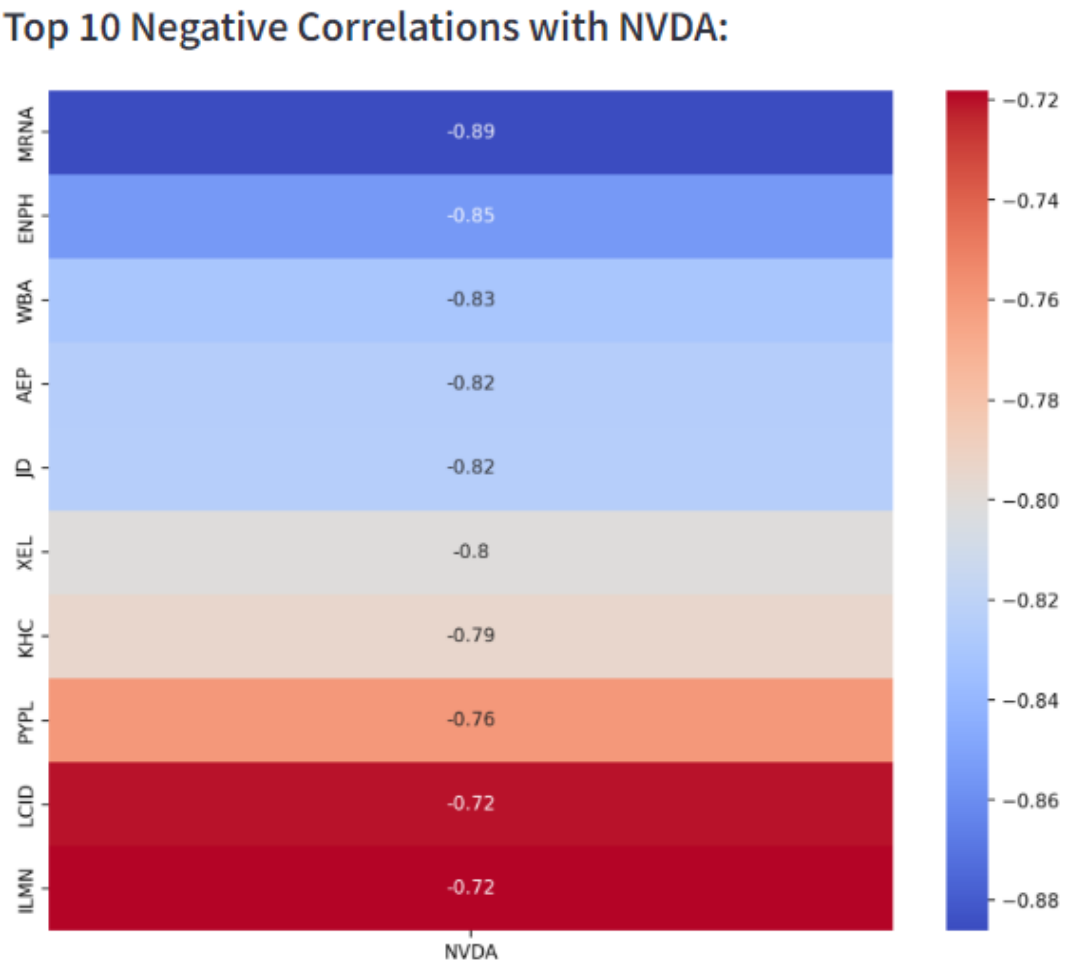
Top 10 Positive Correlations with NVDA:



Click here to see raw data of the Top 10 Positive Correlations with NVDA

	NVDA
META	0.979
AMZN	0.959
CPRT	0.9575
AMAT	0.9545
VRSK	0.9522
LRCX	0.9507
AVGO	0.9503
GOOGL	0.9493
GOOG	0.948
MSFT	0.9389

Figure 13 Visualisation and raw data of top 10 positive correlations for one of my selected stocks.



Click here to see raw data of the Top 10 Negative Correlations with NVDA

	NVDA
MRNA	-0.8861
ENPH	-0.8546
WBA	-0.8307
AEP	-0.8247
JD	-0.824
XEL	-0.8008
KHC	-0.7945
PYPL	-0.7598
LCID	-0.7201
ILMN	-0.7181

Figure 14 Visualisation and raw data of top 10 negative correlations for one of my selected stocks.

Task 4 Performing EDA (Exploratory Data Analysis) for each selected stock

The task requirements as set out in the assessment brief, were to create EDA (Exploratory Data Analysis) for each of my selected stock, for task 3 I already provided EDA for the top 10 positive and negative correlations for my stock in Figures 13 and 14, however I decided to provide two extra EDA's.

Figure 15 is a temporal time series plots for my selected stock. As you can see we can visualise this EDA and make judgement that the adjusted close price for the stock 'BKNG' is far superior than my other selected stocks ['AMD', 'NVDA', 'ORLY].

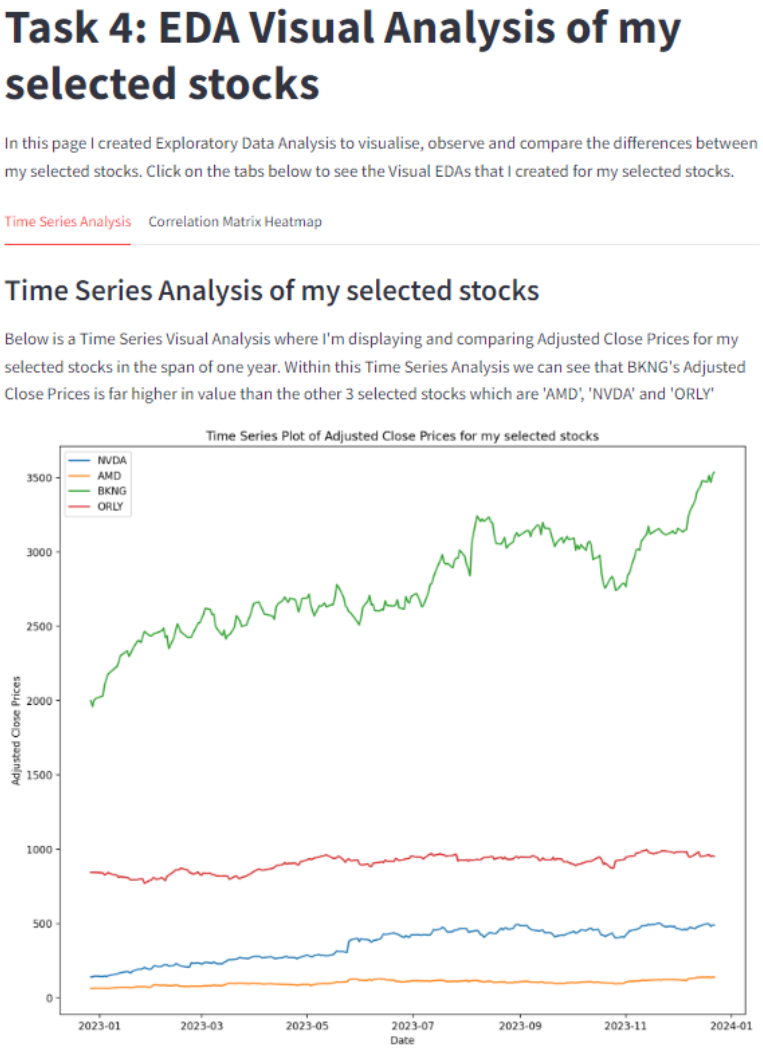


Figure 15 Time Series Analysis of my selected stocks

Figure 16 is the code that I wrote to create the time series plots of my selected stocks to complete this task requirement set out in the assessment brief.

```
def time_series_plots_for_my_selected_stocks():
    # Creating and displaying a chart with a historical view of Adjusted
Close prices for all my selected stocks.
    plt.figure(figsize=(12, 10))

    for stock in selected_stocks:
        plt.plot(selected_stocks.index, selected_stocks[stock],
label=stock)

    plt.title("Time Series Plot of Adjusted Close Prices for my selected
stocks")
    plt.xlabel("Date")
    plt.ylabel("Adjusted Close Prices")
    plt.legend()
    st.subheader("Time Series Analysis of my selected stocks")
    st.write("Below is a Time Series Visual Analysis where I'm displaying
and comparing Adjusted Close Prices for my "
             "selected stocks in the span of one year. Within this Time
Series Analysis we can see that BKNG's Adjusted"
             " Close Prices is far higher in value than the other 3
selected stocks which are 'AMD', 'NVDA' and 'ORLY'")
    st.pyplot()
```

Figure 16 code that I wrote to create time series analysis plots.

The other EDA that I created is shown in Figure 17 which is a correlation heatmap showcasing the positive correlation within my selected stocks. Figure 18 showcases the code that I wrote to implement this correlation EDA.

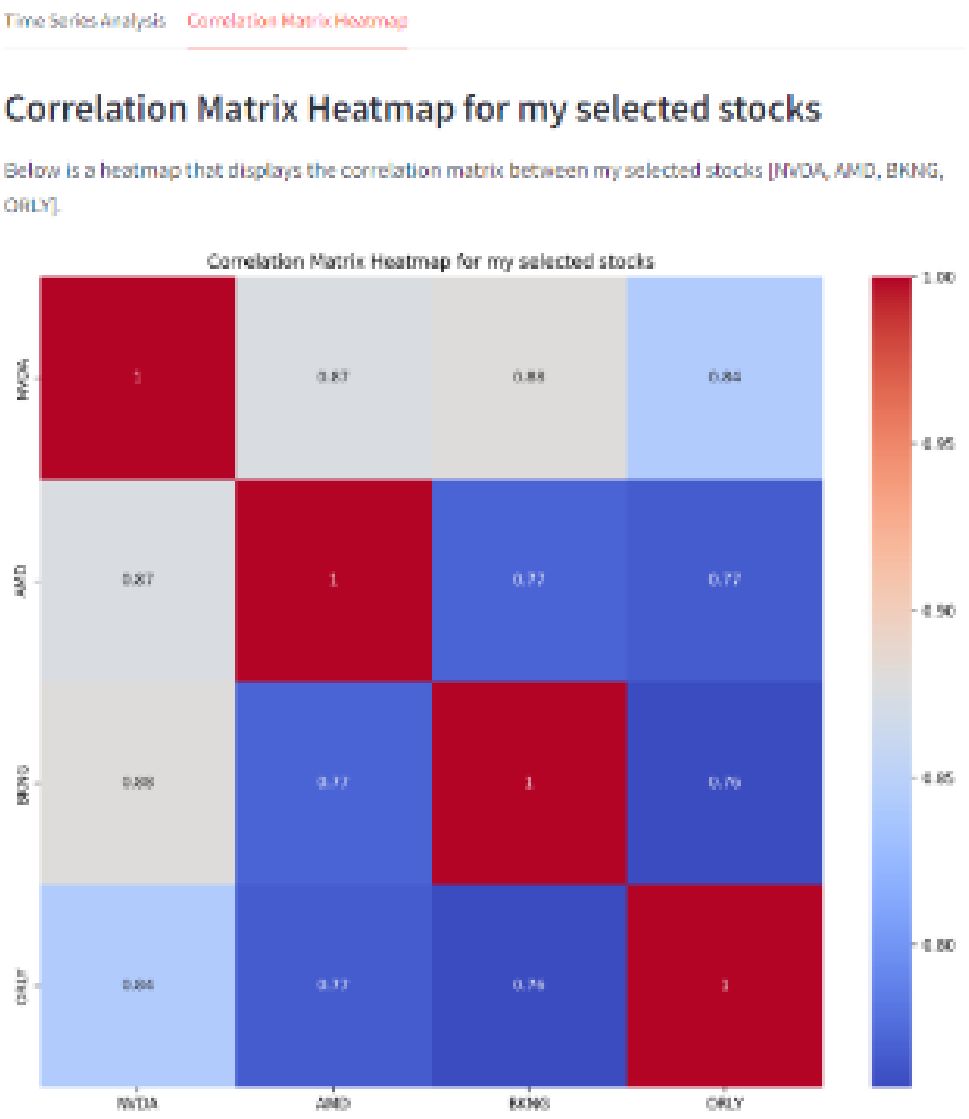


Figure 17 Correlation matrix heatmap for my selected stocks.

```
def correlation_matrix_between_my_selected_stocks():
    print("Correlation Info between my selected stocks")
    # Heatmap that I created to visualise the correlation matrix between my
selected stocks [NVDA, AMD, BKNG, ORLY].
    plt.figure(figsize=(12, 10))
    print(selected_stocks_correlated)
    sns.heatmap(selected_stocks_correlated, annot=True, cmap='coolwarm')
    plt.title("Correlation Matrix Heatmap for my selected stocks")
    st.subheader("Correlation Matrix Heatmap for my selected stocks")
    st.write("Below is a heatmap that displays the correlation matrix
between my selected stocks [NVDA, AMD, BKNG, ORLY].")
    st.pyplot()
```

Figure 18 Code that I wrote to implement the correlation matrix heatmap for my selected stocks.

Task 5 Machine Learning Models for Prediction and Forecasting:

For Task 5 as detailed in the brief, I was tasked at choosing 4 machine learning models and utilising them to train over my selected stock data to predict and forecast the adjusted close prices for selected stocks. I will then critically analyse the performance of these ML models, alongside that determine the best model out of the 4 that can be used for Task 6 to provide market analysis to the user.

Linear Regression

Linear regression stands out as one of the easiest and most accessible Machine Learning algorithms utilised for predictive tasks. Linear Regression is a statistical method that is used for predictive and forecast analysis. Linear Regression makes predictions by modelling the relationship between a dependent variable and one or more independent variables in a linear fashion. The Linear Regression algorithm outputs a straight sloped linear line representing the relationship between the variables (Anon, n.d).

Advantages of Linear Regression is it's a straightforward easy to understand and easy to implement when applying it to my software solution. I had no difficulty in implementing the model to predict my adjusted close prices for my selected stocks.

A major limitation with Linear Regression is that assumes a linear relationship between the dependent and independent variables. If the relationship is not linear and there is quite allot of variance with the relationship between the variables it will not perform accurately when predicting (Gupta 2023). In Figure 19 you can see due to the variance in variables for the stock "AMD", linear regression is predicting poorly.

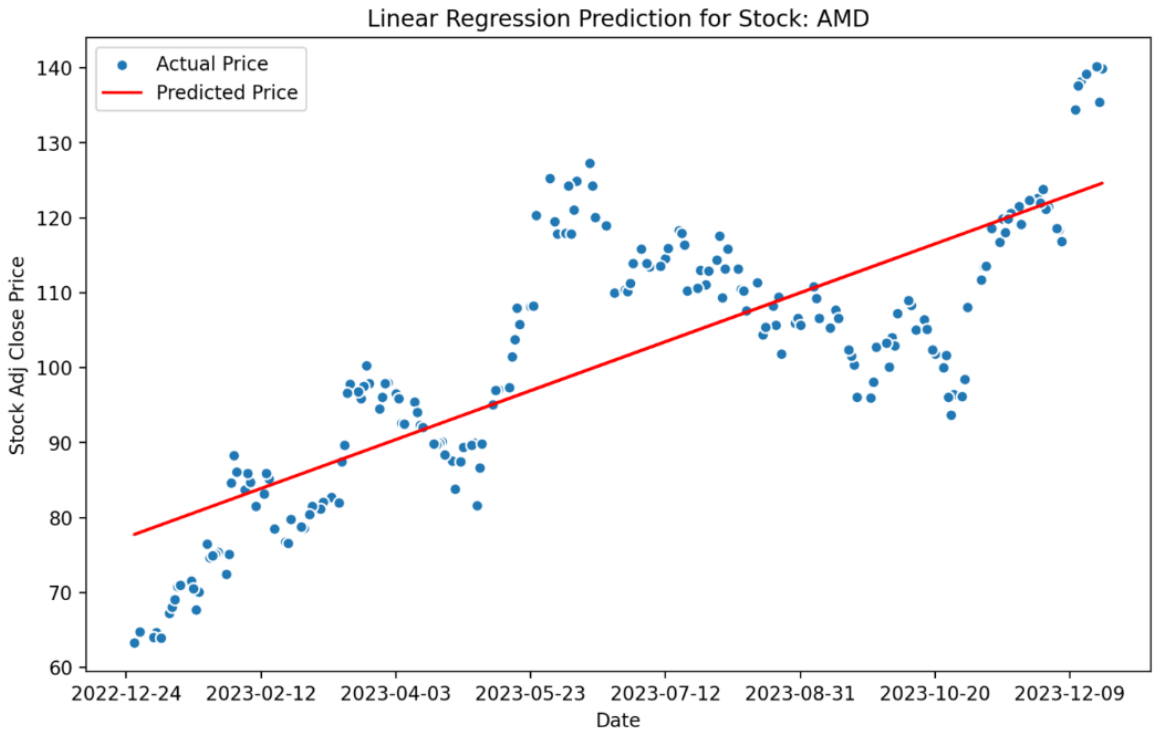


Figure 19 Linear Regression limitation when predicting stock: AMD due to the variance with data variables.

Linear Regression Code Implementation

```
def linear_regression():
    for stock in selected_stocks:
        # Accessing the Adj Close prices for each of my selected stocks
        stock_price = selected_stocks[stock]

        # Split data into train and test set: 80% / 20%
        train, test = train_test_split(stock_price.to_frame(),
test_size=0.20)

        # Reshape x and y train data for linear regression
        X_train = date2num(train.index).astype(float).reshape(-1, 1)
        y_train = train[stock]

        # Create Linear Regression Model and then fit the data to the model
        model = LinearRegression()
        model.fit(X_train, y_train)

        # Visualize the Linear Regression predictions against the actual
price data
        linear_regression_fig = plt.figure(figsize=(10, 6))
        plt.title(f"Linear Regression Prediction for Stock: {stock}")
        plt.scatter(X_train, y_train, edgecolors='w', label='Actual Price')
        plt.plot(X_train, model.predict(X_train), color='r',
label='Predicted Price')
        plt.gca().xaxis.set_major_formatter(mdates.DateFormatter('%Y-%m-
%d'))
        plt.xlabel('Date')
        plt.ylabel('Stock Adj Close Price')
        plt.legend()
        st.pyplot(linear_regression_fig)
```

Figure 20 Code Implementation for Linear Regression

Figure 20 is the code that I wrote to predict and forecast adjusted close prices for my selected stocks using Linear Regression.

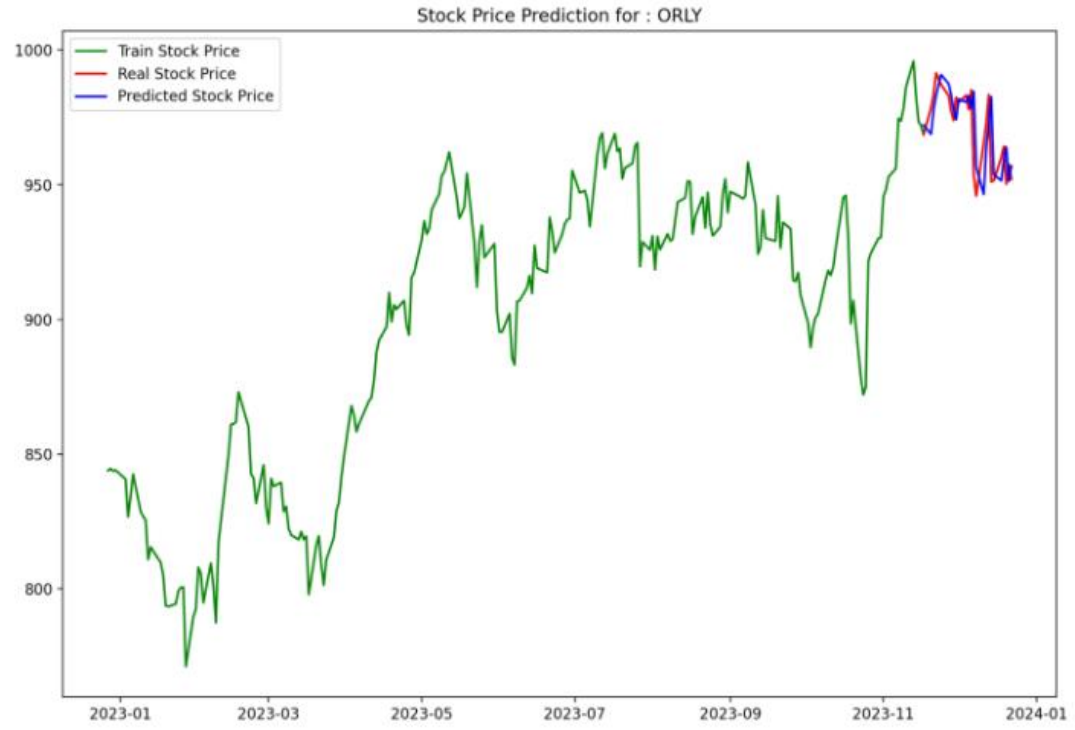
ARIMA (Autoregressive Integrated Moving Average)

ARIMA (Autoregressive Integrated Moving Average) is a statistical model that makes predictions based on historic data. Widely utilised in the realms of finance and economics, ARIMA utilises historical data to make predictions. ARIMA shines in forecasting stock prices by analysing previous stock data (Hayes 2023).

One disadvantage that I was suffering when compared to the other implemented models was the difficulty in setting up the ARIMA model to forecast and predict my selected stocks. It was also sensitive to the model parameters and the trained data, there was quite allot of wasted time and effort in trialling and erroring to find the optimal parameters for this model.

Another disadvantage I discovered is ARIMA'S poor performance in forecasting the adjusted close prices for my selected stocks. As highlighted by Hayes in his article, ARIMA is good with short-term forecasting but is not built for long-term forecasting (Hayes 2023). The 7-day forecasting period I attempted for my selected stocks might be too long and a challenge for the model reason for its less accurate predictions for my selected stocks.

In Figure 21 you can see the performance of the predicted stock price against the real stock price with 90% historical train data. Alongside that you can see the poor performance that is brought up with ARIMA when attempting to perform a 7-day forecast with my selected stocks.



Click here to view the performance of the ARIMA Model prediction on Stock: ORLY

Below attempting to utilise the ARIMA model to give a 7 day future forecasted prediction of Adj Close Price for Stock: ORLY

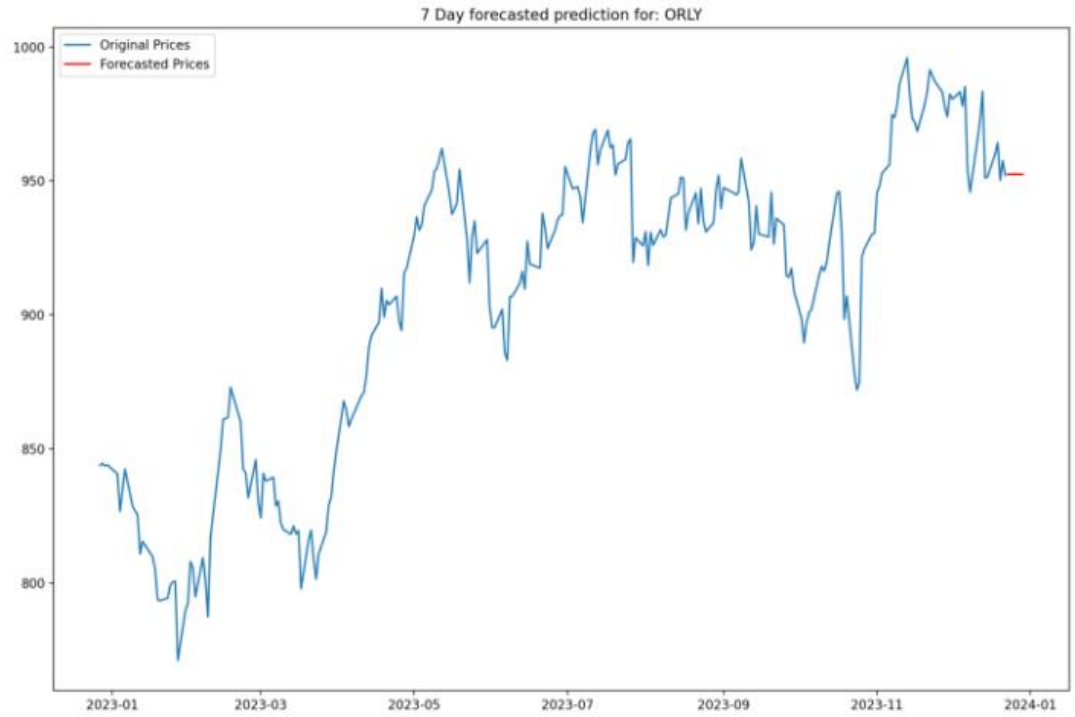


Figure 21 Performance of ARIMA algorithm on my selected Stock: ORLY

ARIMA Code Implementation

Figure 22 showcases my ARIMA code implementation, here I am splitting and prepping the data for the model, creating the model, and then fitting the model, then providing multiple forecasts with the ARIMA Model.

```
def arima():
    for stock in selected_stocks:
        stock_prices = selected_stocks[stock]

        # Split the data into training data and testing data
        train_size = int(len(stock_prices) * 0.9)
        train_data = stock_prices[:train_size]
        test_data = stock_prices[train_size:]

        # Building the train and test data for the model
        history = [x for x in train_data]

        # Storing the stock prices predictions
        predictions = list()

        # Creating the Arima Model and fitting the Arima model ready for
training.
        arima_model = ARIMA(history, order=(1, 1, 0))
        fitted_arima_model = arima_model.fit()
        forcasted_values = fitted_arima_model.forecast()[0]
        predictions.append(forcasted_values)
        history.append(test_data[0])

        # Rolling multiple forecasts
        for i in range(1, len(test_data)):
            # Prediction
            arima_model = ARIMA(history, order=(1, 1, 0))
            fitted_arima_model = arima_model.fit()
            forcasted_values = fitted_arima_model.forecast()[0]
            predictions.append(forcasted_values)
            observations = test_data[i]
            history.append(observations)
```

Figure 22 Prepping the data for the newly created ARIMA model, then creating the ARIMA model and then rolling multiple forecasts.

In Figure 23 is code that I wrote that plots the ARIMA results predicting against real stock data with historic data, I also output the performance of the ARIMA model and finally plot the forecasted predicted adjusted close prices for each of my selected stock for the next 7 days.

```
# Plotting the results
plt.figure(figsize=(12, 8))
plt.plot(stock_prices, color='green', label='Train Stock Price')
plt.plot(test_data.index, test_data, color='red', label='Real Stock Price')
plt.plot(test_data.index, predictions, color='blue', label='Predicted Stock
Price')
plt.title(f'Stock Price Prediction for : {stock}')
plt.legend()
st.pyplot()

with st.expander(f"**Click here to view the performance of the ARIMA Model
prediction on Stock: {stock}**"):
    # Reporting Performance for the ARIMA model
    mse = mean_squared_error(test_data, predictions)
    st.write('MSE: ' + str(mse))
    mae = mean_absolute_error(test_data, predictions)
    st.write('MAE: ' + str(mae))
    rmse = math.sqrt(mean_squared_error(test_data, predictions))
    st.write('RMSE: ' + str(rmse))

# Utilising my current Arima model to predict stock prices for the next 7
days
future_arima_model = ARIMA(stock_prices, order=(1, 1, 0))
fitted_future_arima_model = future_arima_model.fit()
next7_forecasted_values = fitted_future_arima_model.forecast(steps=7)

st.write("Below attempting to utilise the ARIMA model to give a 7 day
future forecasted prediction of Adj Close"
         f" Price for Stock: {stock}")
# Plotting the forecasted predicted prices for each stock in the next 7
days
plt.figure(figsize=(12, 8))
plt.plot(stock_prices.index, stock_prices, label='Original Prices')
forecast_dates = pd.date_range(start=stock_prices.index[-1], periods=8,
freq='D')[1:]
plt.plot(forecast_dates, next7_forecasted_values, color='red',
label="Forecasted Prices")
plt.title(f"7 Day forecasted prediction for: {stock}")
plt.legend()
plt.tight_layout()
st.pyplot()
```

Figure 23 Plotting prediction vs real stock prices, reporting performance, and plotting forecasted prices for each stock in the next 7 days.

LSTM (Long Short-Term Memory)

LSTM is a type of Recurrent Neural network which has the capacity to learn to improve its efficiency with a specific problem (Brownlee 2021). An RNN has the ability to remember previous information and utilise that information to assist with processing a task, the shortcoming off RNN though is they are not affective in remembering or processing long-term dependencies due to it's vanishing gradient. How LSTM differs from an RNN is that it utilises model Layers and Memory cells that allow it to retain important information over a sequence to train the LSTM model (Saxena 2024). With the use of memory cells with LSTM has the ability to capture, retain and persist data making it a good choice for time-series prediction.

One positive is that LSTM has the capability to learn trends and patterns within time series data effectively. Srivatsavaya highlights a disadvantage that I suffered when implementing LSTM for my solution, and that is LSTM on large datasets can be time consuming to train and be recourse hungry whilst training (Srivatsavaya 2023), when training my model, I had to stick to a low epoch iteration so that it wouldn't be time consuming or use up too many of my resources when training my model.

In Figure 24 I showcase the three different graphs, one showcasing the validation loss whilst training the model, the second graph is plotting predictions with real historical data for my selected stock, finally the last graph is plotting it's prediction vs training data that was fed to the model.

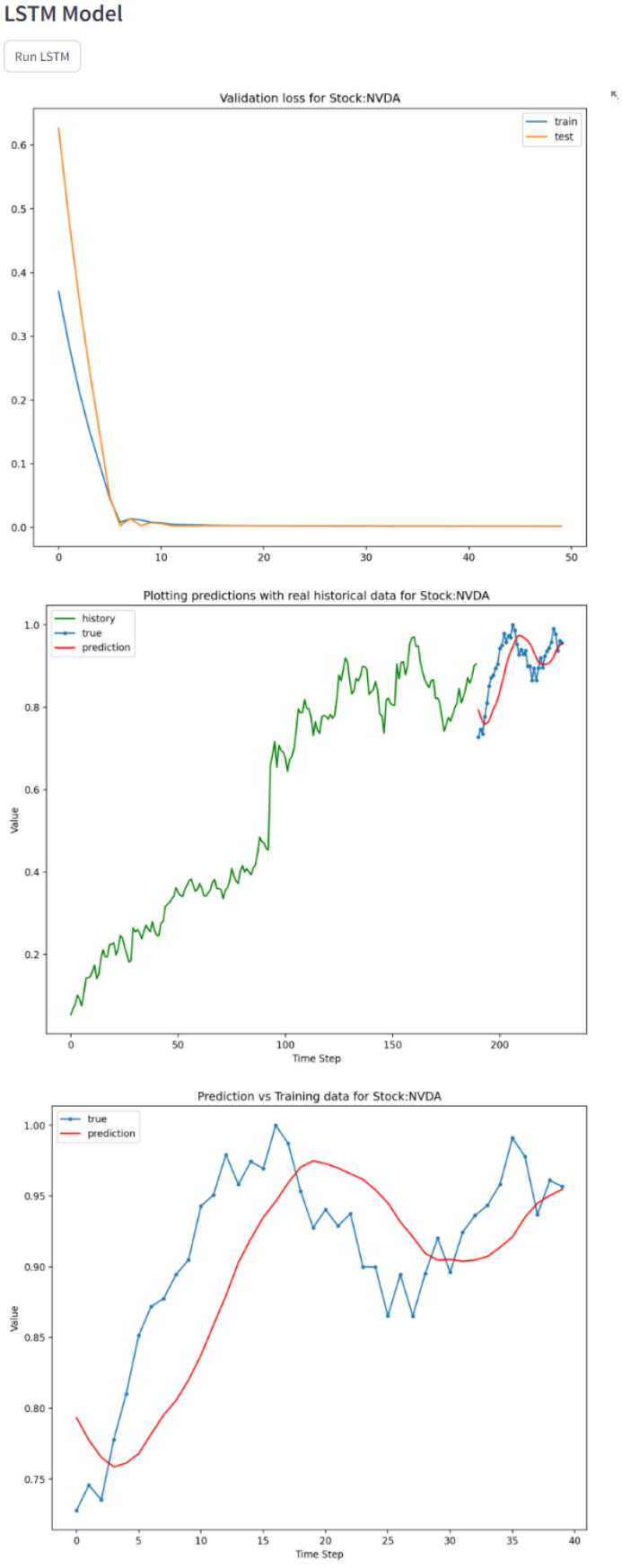


Figure 24 Three graphs that are presented to the user for my selected stocks.

LSTM Code Implementation

```
def lstm():
    for stock in selected_stocks:
        # Normalizing the data using MinMax Scaling
        scaler = MinMaxScaler()
        scaled_data =
scaler.fit_transform(selected_stocks[stock].values.reshape(-1, 1))

        # Splitting the test and train data of the scaled data.
        train_size = int(len(scaled_data) * 0.8)
        test_size = int(len(scaled_data)) - train_size
        train_data, test_data = scaled_data[0: train_size],
scaled_data[train_size:len(scaled_data), :1]
        print(len(train_data), len(test_data))

        # Function created to create the dataset for LSTM prediction
        def create_dataset(data, time_steps):
            X, y = [], []
            for i in range(len(data) - time_steps):
                X.append(data[i:(i + time_steps)])
                y.append(data[i + time_steps])
            return np.array(X), np.array(y)

        time_steps = 10

        # Using the create dataset function to create the dataset that will
be used for LSTM Prediction
        X_train, y_train = create_dataset(train_data, time_steps)
        X_test, y_test = create_dataset(test_data, time_steps)
        print(X_train[0], y_train[0])

        # Creating the LSTM model
        model = Sequential()
        model.add(LSTM(units=50, activation='relu',
input_shape=(time_steps, 1)))
        model.add(Dense(units=1))
        model.compile(optimizer='adam', loss='mse')

        # Training the model
        history = model.fit(X_train, y_train, epochs=50, batch_size=32,
validation_split=0.1, verbose=1, shuffle=False)
```

Figure 25 LSTM code implementation: Prepping dataset for LSTM and creating the LSTM model.

```
# Visualising the training and testing process validation during training
the LSTM model that I have created above.
fig_validation_loss = plt.figure(figsize=(10, 8))
plt.plot(history.history['loss'], label='train')
plt.plot(history.history['val_loss'], label='test')
plt.title(f"Validation loss for Stock:{stock}")
plt.legend()
st.pyplot(fig_validation_loss)

y_prediction = model.predict(X_test)

# Evaluation of the predicted results made by the LSTM model, the
visualisation shows the historic data and then
# presents the future prediction made by the LSTM model for each stock.
Finally plot the predicted results
fig_prediction = plt.figure(figsize=(10, 8))
plt.plot(np.arange(0, len(y_train)), y_train, 'g', label="history")
plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)), y_test,
marker='.', label="true")
plt.plot(np.arange(len(y_train), len(y_train) + len(y_test)), y_prediction,
'r', label="prediction")
plt.ylabel('Value')
plt.xlabel('Time Step')
plt.title(f"Plotting predictions with real historical data for
Stock:{stock}")
plt.legend()
st.pyplot(fig_prediction)

fig_single_prediction = plt.figure(figsize=(10, 8))
plt.plot(y_test, marker='.', label="true")
plt.plot(y_prediction, 'r', label="prediction")
plt.ylabel('Value')
plt.xlabel('Time Step')
plt.title(f"Prediction vs Training data for Stock:{stock}")
plt.legend()
plt.show()
st.pyplot(fig_single_prediction)
```

Figure 26 Visualising the validation loss, predictions with historical data and prediction vs training data, all visual plots for my selected stocks.

Figures 25 and 26 are my code implementation of utilising LSTM to predict and forecast adjusted close prices for my selected stocks.

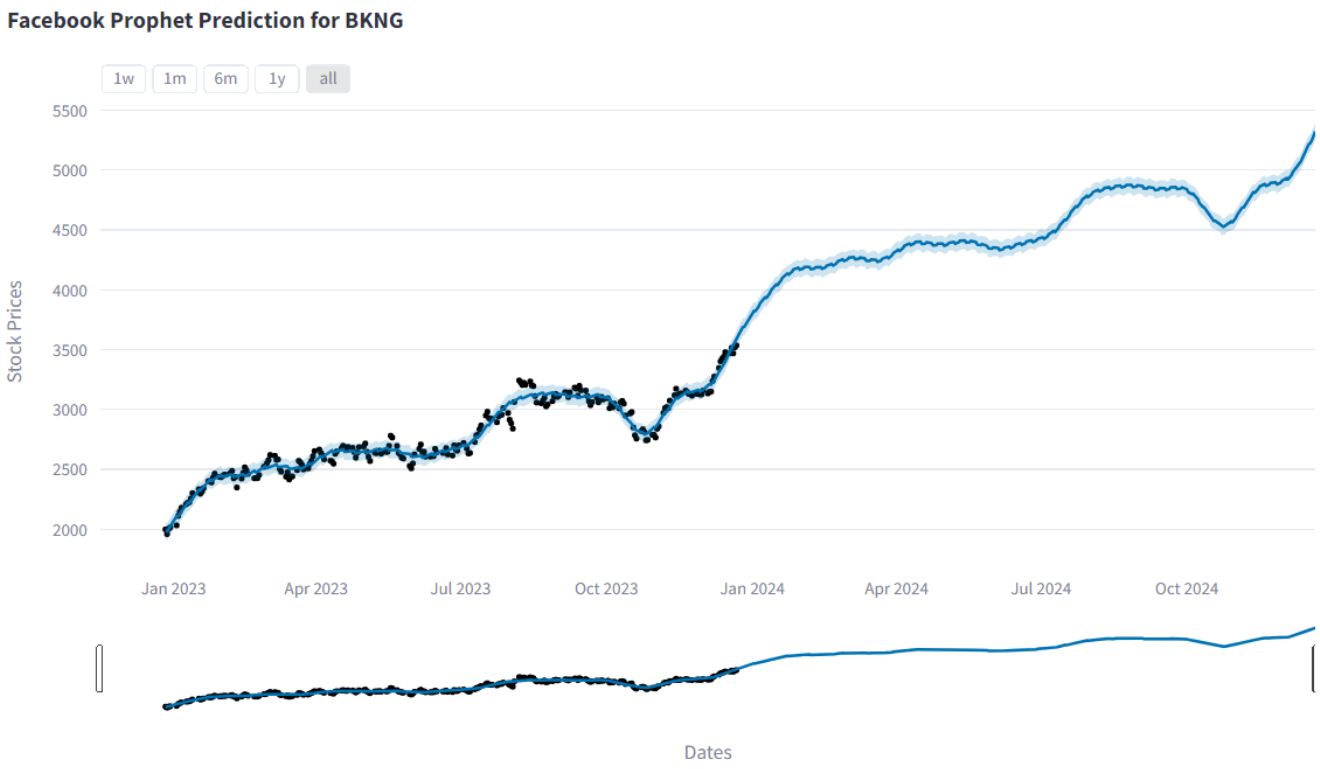
Facebook Prophet

FB Prophet or Prophet as it can be known as is a forecasting algorithm developed by Facebook (Khare 2023). It is designed for forecasting time series data, where Facebook Prophet excels is dealing with large sets of historical data, showcasing its effectiveness in predicting trends or patterns over time. Whether dealing with seasonality or historical data trends, Prophet has been proven to excel in predicting sales and stock prices (Anon, n.d).

One advantage that I came across with FB Prophet was it was the simplest model compared to other models discussed to implement for forecasting and predicting my selected stocks, an article by Hayes also agrees with the simplicity of using FB Prophet (Hayes 2021).

A negative highlighted in this article by Vasselin and Bertrand is that it is prone to overfitting, if you provide FB Prophet with too much data than recommended than the performance and accuracy of the forecast prediction decreases (Vasselin and Bertrand 2022).

In Figure 27 I showcase myself utilising FB Prophet on my application in forecasting and predicting adjusted close prices for one of my selected stocks, on my application I have presented a graph where we can compare the predicted plots made by the FB Prophet model represented with the blue line and the black dots representing actual data. As you can see in Figure 27 the performance of the prediction made by FB Prophet is very close and accurate to actual data showing a strong prediction performance by FB Prophet.



Click here to see FB Prophet's Predicted Adjusted Close Prices for: BKNG

	Date	Adjusted Close Prices
0	2022-12-27 00:00:00	1,980.2012
1	2022-12-28 00:00:00	1,983.3844
2	2022-12-29 00:00:00	2,003.3274
3	2022-12-30 00:00:00	2,027.4201
4	2023-01-03 00:00:00	2,109.3247
5	2023-01-04 00:00:00	2,108.4168
6	2023-01-05 00:00:00	2,124.7926
7	2023-01-06 00:00:00	2,145.8663
8	2023-01-09 00:00:00	2,204.5943
9	2023-01-10 00:00:00	2,220.821

Figure 27 FB Prophet Prediction of Stock BKNG showcased on my GUI Application.

Facebook Prophet Code Implementation

In Figure 28 showcases the code that I wrote to implement FB Prophet to predict and forecast my stocks. As you can see in the code implementation there is allot less lines required and no requirements to split up or prepare data for the model which showcases the simplicity to utilise FB Prophet highlighted by the article by Hayes.

```
def fb_prophet():
    # Resetting and clearing the data to be processed for the Facebook
Prophet Method
    selected_stocks.reset_index(inplace=True)
    selected_stocks_Date = selected_stocks['Date']

    for stock in selected_stocks.iloc[:, 1:]:
        prophet = Prophet(
            daily_seasonality=True,
            yearly_seasonality=True,
            weekly_seasonality=True,
            changepoint_prior_scale=0.05,
            seasonality_prior_scale=10.0
        )

        # Creating a new DataFrame with the required columns for Facebook
Prophet Prediction
        new_prophetDF = pd.DataFrame({'ds': selected_stocks_Date, 'y':
selected_stocks[stock]})
        prophet.fit(new_prophetDF)

        # Creating a DataFrame to be used for the prediction
        future = prophet.make_future_dataframe(periods=365)

        # Passing the future DataFrame to generate a forecast prediction
for my selected stocks.
        forecast = prophet.predict(future)

        # Plot the predictions that were made by Facebook Prophet Market
prediction
        fig = plot_plotly(prophet, forecast)
        fig.update_layout(xaxis_title="Dates", yaxis_title="Stock Prices",
title_text=f"Facebook Prophet Prediction for {stock}")
        st.plotly_chart(fig)
        with st.expander(f"**Click here to see FB Prophet's Predicted
Adjusted Close Prices for: {stock}**"):
            st.write(forecast[['ds', 'yhat']].rename(columns={'ds': 'Date',
'yhat': 'Adjusted Close Prices'}))
```

Figure 28 Code Implementation for FB Prophet

Task 6 General Trading Signals and Analysis

For Task 6 as stated in the assessment brief, is to provide market analysis for a specific time variance (7, 14, 30 days) using the forecast prediction that I perceive is the best one after comparing and analysing the 4 models that I utilised in Task 5 to make predictions. After my analysis of the performance of the 4 models I am adamant at utilising FB Prophet as I strongly believe it showed the best performance when predicting and forecasting my selected stocks against real data, alongside that it showed a strong performance when forecasting future stock price predictions. With that in mind Figures 29,30 and 31 showcase the code implementation utilising FB Prophet to showcase market analysis depending on the user's choice of stock and the time variance that they have selected.

```
def user_selected_stock_forecast_analysis():
    stock_user_selection = st.text_input("Enter the stock into the text
field that you would like to analyse (e.g AAPL)"
                                         " 👇").upper()
    user_selected_stock_DF = pd.DataFrame()
    user_selected_stock_data = yf.download(stock_user_selection,
start=start_date, end=end_date)
    user_selected_stock_DF[stock_user_selection] =
user_selected_stock_data['Adj Close']
    user_input = st.selectbox("Select the analysis duration:", [7, 14, 30])
    if st.button("Analyse Stock"):
        if stock_user_selection in tickers:
            if user_input == 7:
                st.write(f"Analysing and forecasting stock prices for
{stock_user_selection} for the next 7 days")

user_selected_stock_forecast_analysis_with_fbProphet(user_selected_stock_DF
, future_days=7)
            elif user_input == 14:
                st.write(f"Analysing and forecasting stock prices for
{stock_user_selection} for the next 14 days")

user_selected_stock_forecast_analysis_with_fbProphet(user_selected_stock_DF
, future_days=14)
            elif user_input == 30:
                st.write(f"Analysing and forecasting stock prices for
{stock_user_selection} for the next 30 days")

user_selected_stock_forecast_analysis_with_fbProphet(user_selected_stock_DF
, future_days=30)
        else:
            st.write("Unable to find Stock information for that inputted
Stock Code")
```

Figure 29 Code Implementation that takes the users selected stock and time variance to pass it to my user selected FB Prophet function for market analysis.

```
def
user_selected_stock_forecast_analysis_with_fbProphet(user_selected_stock,
future_days=365):
    # This function has been created to allow the user to select a stock
within the tickers list to be able to get an
    # analysis forecasting prediction provided by fb_prophet for their
selected stock depending on the choice of time.

    user_selected_stock_date = user_selected_stock.index
    user_selected_stock_prices =
user_selected_stock[user_selected_stock.columns[0]]

    # Fetch today's latest stock data and then adding it to the
user_selected_stock DataFrame
    latest_data = yf.download(user_selected_stock.columns[0],
start=end_date, end=end_date)
    user_selected_stock = pd.concat([user_selected_stock, latest_data['Adj
Close']])

    prophet = Prophet(
        daily_seasonality=True,
        yearly_seasonality=True,
        weekly_seasonality=True,
        changepoint_prior_scale=0.05,
        seasonality_prior_scale=10.0
    )

    # Creating a new DataFrame with the required columns for Facebook
Prophet Prediction
    newProphetDF = pd.DataFrame({'ds': user_selected_stock_date, 'y':
user_selected_stock_prices})
    prophet.fit(newProphetDF)

    # Creating a new DataFrame to be will be used for the prediction, but
for this prediction we will pass the period
    # as a variable so that we can utilise this function for forecast stock
prices for the users inputted stock
    # against different time periods which are 7, 14 and 30 days.
    future = prophet.make_future_dataframe(periods=future_days)

    # Passing the future DataFrame with the future days variable that will
be passed and changed depending on the
    # forecasting analysis on the users selected stock
    forecast = prophet.predict(future)
```

Figure 30 Facebook Prophet user selected stock and time variance code implementation part 1 of 2.

```
# Plot the predictions that were made by Facebook Prophet Market prediction
fig = plot_plotly(prophet, forecast)
fig.update_layout(xaxis_title="Dates", yaxis_title="Stock Prices",
title_text=f"Facebook Prophet Prediction for
{user_selected_stock.columns[0]}")
st.subheader(f"Selected Stock: {user_selected_stock.columns[0]} Facebook
Prophet Prediction Chart")
st.plotly_chart(fig)

# Accessing the last/latest 'Adj Close' price for the user-selected stock
st.subheader(f"Selected Stock: {user_selected_stock.columns[0]} Latest
Stock Information")
last_stock_date_price = user_selected_stock.iloc[-1]
last_stock_date = user_selected_stock.index[-1]
st.write(f"**Latest Updated Stock Date: {last_stock_date}**")
st.write(f"**Latest Stock Price: {last_stock_date_price.iloc[0]}**")
print(f"Latest Stock Price: {last_stock_date_price.iloc[0]}")

# Print the forecasted prices for the users selected stock depending on the
future days that have been entered by
# the user with the user_selected_stock_forecast_analysis function.
forecast.rename(columns={'ds': 'Date', 'yhat': 'Adj Close Price'},
inplace=True)
st.subheader(f"Forecasted Prices for
Stock({user_selected_stock.columns[0]}) for the Next {future_days} Days:")
st.write(forecast[['Date', 'Adj Close Price']].tail(future_days))

# I am retrieving data for the different time variance depending on the
future days variable which will be 7, 14, 30
# days. I access the data within the forecast dataframe and then offset the
data depending on the time variance from
# the future_days variable, I then save this as a variable to be used for
comparison for stock analysis.
future_days_stock_price = forecast.loc[forecast['Date'] == (last_stock_date
+ pd.DateOffset(days=future_days)), 'Adj Close Price'].iloc[0]

# Display stock analysis to the user regarding if they should invest/buy
into the stock or sell/not invest into the
# stock depending on their time analysis.
st.subheader("User Selected Stock Analysis")
if last_stock_date_price.iloc[0] > future_days_stock_price:
    st.write("**I'd advise that you sell this stock or don't invest in this
stock at all.**")
    st.write(f"**Latest stock price for ({user_selected_stock.columns[0]})
is: {last_stock_date_price.iloc[0]}.**")
    st.write(f"**Which is higher than the predicted stock price valued at:
{future_days_stock_price} in {future_days} "
             "days time.**")
elif last_stock_date_price.iloc[0] < future_days_stock_price:
    st.write("**I'd advise you to invest in this stock.**")
    st.write(f"**Latest stock price for ({user_selected_stock.columns[0]})
is: {last_stock_date_price.iloc[0]}.**")
    st.write(f"**Which is lower than the predicted stock price valued at:
{future_days_stock_price} in {future_days}**"
             f" **days time**")
```

Figure 31 Facebook Prophet user selected stock and time variance code implementation part 2 of 2.

Task 7 GUI (Graphical User Interface)

As stated in the task requirements brief for Task 7, is to provide a Graphical User Interface that allows the user to perform EDA, Correlation, Prediction Forecasting and Generate buy/sell signals.

There were two choices of GUI implementation that I can utilise, I could use Tkinter and make a desktop application or utilise Streamlit to make a web application, I decided to opt for Streamlit as it excels in creating and sharing web applications in the realm of machine learning projects with minimal effort. When researching the best option for my GUI application, I found when looking at documentation for streamlet highlighted the simplicity it has when publicly deploying the application.

Conclusion

In conclusion I believe I have made a great application that is feature complete and that has completed all the tasks as per stated in the assessment brief. In this report and my Software Artefact Solution I have highlighted my knowledge and understanding utilising specific libraries to improve data visualisation, understood the use of dimensionally reduction algorithms to reduce the size of data without the need of losing crucial and key data and finally understood the utilisation of Machine Learning models to predict and forecast stock data and discuss pros and cons of each machine learning algorithm.

If I had more time or outside of this project deadline, I would look to improve or resolve the caching error that sometimes occurs as stated in my demonstration video. Another addition that I seek to do is deploy this as a public web application into the real world as I am unable to do this now due to time restrictions.

References

BROWNLEE, J., 2020. Principal Component Analysis for Dimensionality Reduction in Python [Viewed on 9th January 2024]. Available From: https://machinelearningmastery.com/principal-components-analysis-for-dimensionality-reduction-in-python/
JAADI, Z ., 2023. A Step-by-Step Explanation of Principal Component Analysis(PCA) [Viewed on 9th January 2024]. Available From: https://builtin.com/data-science/step-step-explanation-principal-component-analysis
PAL, A., 2018. Understanding Dimension Reduction with Principal Component Analysis(PCA) [Viewed on 9th January 2024]. Available From: https://blog.paperspace.com/dimension-reduction-with-principal-component-analysis/
ALI, M., 2022. Clustering in Machine Learning: 5 Essential Clustering Algorithms [Viewed on 10th January 2024]. Available From: https://www.datacamp.com/blog/clustering-in-machine-learning-5-essential-clustering-algorithms
SHARMA, N., 2023. K-Means Clustering Explained [Viewed on 10th January 2024]. Available From: https://neptune.ai/blog/k-means-clustering
ANON., date unknown. Linear Regression in Machine Learning [Viewed on 10th January 2024]. Available From: https://www.javatpoint.com/linear-regression-in-machine-learning
GUPTA, M., 2023. Linear Regression in Machine Learning [Viewed on 10th January 2024]. Available From: https://www.geeksforgeeks.org/ml-linear-regression/
HAYES, A., 2023. Autoregressive Integrated Moving Average (ARIMA) Prediction Model [Viewed on 11th January 2024]. Available From: https://www.investopedia.com/terms/a/autoregressive-integrated-moving-average-arima.asp
BROWNLEE, J., 2021. A Gentle Introduction to Long Short-Term Memory Networks by the Experts [Viewed on 12th January 2024]. Available From: https://machinelearningmastery.com/gentle-introduction-long-short-term-memory-networks-experts/
SAXENA, S., 2024. What is LSTM? Introduction to Long Short-Term Memory [Viewed on 12th January 2024]. Available From: https://www.analyticsvidhya.com/blog/2021/03/introduction-to-long-short-term-memory-lstm/
SRIVATSAVAYA, P., 2023. LSTM – Implementation, Advantages and Disadvantages [Viewed on 12th January 2024]. Available From: https://medium.com/@prudhviraju.srivatsavaya/lstm-implementation-advantages-and-diadvantages-914a96fa0acb
KHARE, P., 2023. Understanding FB Prophet: A Time Series Forecasting Algorithm [Viewed on 12th January 2024]. Available From: https://medium.com/illumination/understanding-fb-prophet-a-time-series-forecasting-algorithm-c998bc52ca10#:~:text=FBProphet%20is%20a%20powerful%20time,uncertainty%20estimation%20in%20the%20predictions.

ANON., date unknown. PROPHET Forecasting at scale [Viewed on 12th January 2024]. Available From: https://facebook.github.io/prophet/

HAYES, B., 2021. Time Series Analysis With Facebook Prophet [Viewed on 12th January 2024]. Available From: https://benhay.es/posts/time-series-facebook-prophet/

Bibliography

O'CONNELL, R., 2023. Free Stock Prices in Python Made Easy | yFinance [Viewed on 12th November 2023]. Available From: https://www.youtube.com/watch?v=PVhecjQcfV4&ab_channel=RyanO%27Connell%2CCFA%2CFRM
ANON., 2023. Nasdaq 100 Companies [Viewed on 12th November 2023]. Available From: https://www.slickcharts.com/nasdaq100
ANON., 2023. NASDAQ 100(^NDX) [Viewed on 12th November 2023]. Available From: https://finance.yahoo.com/quote/%5ENDX/history?p=%5ENDX
MONDAL, A., 2022. Download Financial Dataset Using Yahoo Finance in Python | A Complete Guide [Viewed on 12th November 2023]. Available From: https://www.analyticsvidhya.com/blog/2021/06/download-financial-dataset-using-yahoo-finance-in-python-a-complete-guide/
AROUSSI, R., date unknown. yfinance 0.2.31 [Viewed on 12th November 2023]. Available From: https://pypi.org/project/yfinance/
RAJTT, P., 2023. Get Financial Data from Yahoo Finance with Python [Viewed on 12th November 2023]. Available From: https://www.geeksforgeeks.org/get-financial-data-from-yahoo-finance-with-python/
DEV, U., 2020. EDA of Stock Market using Time Series [Viewed on 19th December 2023]. Available From: https://usharbudha-dev09.medium.com/eda-of-stock-market-using-time-series-9662fd18bfc5
MEGHNA, K., 2023. EDA| Exploratory Data Analysis in Python [Viewed on 19th December 2023]. Available From: https://www.geeksforgeeks.org/exploratory-data-analysis-in-python/
SILVA, J., 2023. Predicting stock prices with FBProphet in python[Viewed on 20th December 2023]. Available From: https://www.linkedin.com/pulse/predicting-stock-prices-fbprophet-python-jo%C3%A3o-gabriel-corr%C3%AAa-da-silva/
DUONG, C., 2023. Quick Start [Viewed on 20th December 2023]. Available From: https://facebook.github.io/prophet/docs/quick_start.html#python-api
KUMAR, R., 2021. Predicting Stock Prices Using Facebook's Prophet Model [Viewed on 20th December 2023]. Available From: https://medium.com/analytics-vidhya/predicting-stock-prices-using-facebooks-prophet-model-b1716c733ea6
NEURALNINE., 2022. Predicting Stock Prices with FBProphet in Python [Viewed on 19th December 2023]. Available From: https://www.youtube.com/watch?v=03H2_ekdv2I&t=199s&ab_channel=NeuralNine
ANON., date unknown. Machine Learning (COM624) Weekly Activities [Viewed on 21st December 2023]. Available From: https://learn.solent.ac.uk/course/view.php?id=52725&section=5#tabs-tree-start
AWAN, A., 2023. Time Series Analysis: ARIMA Models in Python [Viewed on 23rd December 2023]. Available From: https://www.kdnuggets.com/2023/08/times-series-analysis-arima-models-python.html#:~:text=The%20ARIMA%20model%20is%20a,for%20AutoRegressive%20Integrated%20Moving%20Average.

IVANYUK, B., 2018. AR, ARIMA, LSTM [Viewed on 23rd December 2023]. Available From: https://www.kaggle.com/code/bogdanbaraban/ar-arima-lstm#ARIMA-model
HEBBAR, N., 2020. ARIMA Model in Python| Time Series Forecasting #6| [Viewed on 25th December 2023] Available From: https://www.youtube.com/watch?v=8FCDpFhd1zk&ab_channel=NachiketaHebbar
HEBBAR, N., 2020. Time Series Forecasting With ARIMA Model in Python for Temperature Prediction [Viewed on 25th December 2023]. Available From: https://medium.com/swlh/temperature-forecasting-with-arima-model-in-python-427b2d3bcb53
ANON., date unknown. Setting the Font, Title, Legend Entries and Axis Titles in Python [Viewed on 26th December 2023]. Available From: https://plotly.com/python/figure-labels/
ANON., date unknown. Streamlit documentation [Viewed on 28th December 2023]. Available From: https://docs.streamlit.io/
PIXEGAMI., 2023. Streamlit: The Fastest Way to Build Pyhton Apps? [Viewed on 28th December 2023]. Available From: https://www.youtube.com/watch?v=D0D4Pa22iG0&ab_channel=pixegami
CODING IS FUN., 2022. Build a Website in only 12 minutes using Python & Streamlit [Viewed on 28th December 2023]. Available From: https://www.youtube.com/watch?v=VqgUkExPvLY&ab_channel=CodingIsFun
ANON., date unknown. Streamlit [Viewed on 28th December 2023]. Available From: https://streamlit.io/
ANON., date unknown. Streamlit Community Cloud [Viewed on 28th December 2023]. Available From: https://streamlit.io/cloud
STREAMLIT., 2022. How to Deploy your App to Streamlit Community Cloud [Viewed on 30th December 2023]. Available From: https://www.youtube.com/watch?v=HKoOBiAaHGg&ab_channel=Streamlit
M, SHASHANKA., 2018. Predicting Stock Prices with Linear Regression [Viewed on 7th January 2024]. Available From: https://github.com/shashimanyam/PREDICTING-STOCK-PRICES-WITH-LINEAR-REGRESSION/blob/master/Predicting%20Stock%20Prices%20with%20Linear%20Regression.ipynb
EDUCATION ECOSYSTEM., 2018. Understanding K-Means Clustering in Machine Learning [Viewed on 10th January 2024]. Available From: https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1
MALI, K., 2023. Everything you need to know about Linear Regression! [Viewed on 10th January 2024]. Available From: https://www.analyticsvidhya.com/blog/2021/10/everything-you-need-to-know-about-linear-regression/#h-what-is-linear-regression
ANON., 2021. What are the downsides of ARIMA models? [Viewed on 11th January 2024]. Available From: https://stats.stackexchange.com/questions/558252/what-are-the-downsides-of-arima-models
ANON., date unknown. What is ARIMA Modelling? [Viewed on 11th January 2024]. Available From: https://www.mastersindatascience.org/learning/statistics-data-science/what-is-arima-modeling/

ANON., date unknown. What are the advantages and disadvantages of using LSTMs for time series prediction? [Viewed on 12th January 2024]. Available From: https://typeset.io/questions/what-are-the-advantages-and-disadvantages-of-using-lstms-for-29rc9pw9zp

AHMED, R., 2022. What is Facebook Prophet? Time Series Forecasting [Viewed on 12th January 2024]. Available From: https://www.youtube.com/watch?v=4YwZYHfCAJk&ab_channel=Prof.RyanAhmed

BROWNLEE, J., 2020. Time Series Forecasting with Prophet in Python [Viewed on 12th January 2024]. Available From: https://machinelearningmastery.com/time-series-forecasting-with-prophet-in-python/

Appendices

My reasoning and choice of my selected stocks was purely a personal choice of mine, I chose NVDA and AMD purely on myself liking their computer components and being a consumer of those two companies, BKNG was the only forced choice as it is the only one to pick from Cluster Group 1 from my solution and finally ORLY was a random choice out of Cluster Group 3.