National College of Ireland

Configuration Manual

MSc Research Project

FinTech

GULAMALI ADREKAR

Student ID: X22213601

School of Computing

National College of Ireland

Supervisor: Mr. Brian Byrne

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……. …GULAMALI ISA ADREKAR……

Student ID: ……X22213601…….

Programme: ……MSc in FinTech…… **Year:** ……2024……..

Module: ……RESEARCH PROJECT…….……

Lecturer: ……MR. BRIAN BYRNE…….……

Submission Due Date: ……12/08/2024……..

Project Title: ……..……

Word Count: ……786…… **Page Count:** ……13……..……..……

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information

other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……GULAMALI ADREKAR……

Date: ……12/08/2024……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	□
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	□
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	□

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	

Date:	
Penalty Applied (if applicable):	

Stock Price Forecasting System Configuration and Implementation

August 12, 2024

Contents

Chapter 1

Introduction

learning along with Deep Learning algorithms. It incorporates historical data analysis to predict forthcoming stock charges. This project uses different models, most of them are ARIMA (AutoRegressive Integrated Moving Average), Logistic Regression, SVC (Support Vector Classifier), XGBoost, LSTM, and CNN.

Chapter 2

System Configuration

2.1 Hardware Requirements

The hardware needed to deploy the system:

- **Processor:** CPU: Multi-core CPU (Intel i5 or equivalent)
- **Memory:** 8GB System RAM (16 GB for deep learning models)
- **Storage:** 50GB free disk space
- **GPU:** NVIDIA GPU with CUDA support (recommended/optional for significantly faster deep learning model training)

2.2 Software Requirements

- **Operating System:** Windows 10, macOS, or Linux
- **Python:** Version 3.11
- **Python Packages:** yfinance, pandas, plotly, statsmodels, scikit-learn, xgboost, tensorflow

Python Environment Setup:

1. Install Python 3.11 from the official website.
2. Set up a virtual environment:
    - On Windows: python -m venv stock forecast env
    - On macOS/Linux: python3 -m venv stock _forecast env
3. Activate the virtual environment:

- On Windows: stock forecast env
  Scripts activate
- On macOS/Linux: source stock forecast env/bin/activate

4. Install required libraries using pip install yfinance pandas plotly statsmodels scikit-learn xgboost tensorflow

Chapter 3

Project Implementation

3.1 Data Collection

```python
# Import necessary libraries
import yfinance as yf
import pandas as pd
import plotly.graph_objs as go
from plotly.subplots import make_subplots

# Fetch historical data for Apple Inc. (AAPL)
ticker = 'AAPL'
data = yf.download(ticker, start='2014-08-01', end='2024-08-01')
```

Figure 3.1: Data fetching for Apple Inc. (AAPL) using yfinance.

The historical stock data for Apple Inc. (AAPL) is fetched using the yfinance library. The data spans from August 1, 2014, to August 1, 2024, and includes daily open, high, low, close, and volume values.

3.2 Feature Selection

Key features such as 'Open', 'High', 'Low', 'Close', and 'Volume' are selected for analysis. Additional derived features like open-close (the difference between the opening and closing prices), low-high (the difference between the lowest and highest prices), and is quarter end (binary feature indicating the end of a quarter) are also created.

3.3 Data Pre-processing

The data is cleaned by handling any missing values and ensuring consistency across the dataset. The MinMaxScaler is used to normalize the data, scaling all features to a range between 0 and 1, which is crucial for training machine learning and deep learning models effectively.

3.4 Feature Engineering

The derived features mentioned in the Feature Selection section are engineered to enhance the predictive power of the models. These features are used in conjunction with the original data to train the models.

3.5 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is performed to understand the distribution and relationships of the features. This includes generating line charts for closing prices, candlestick charts for daily stock performance, moving averages to identify trends, and correlation heatmaps to uncover relationships between features.

```
# 1. Line Chart for Closing Prices
line_fig = go.Figure()
line_fig.add_trace(go.Scatter(x=data.index, y=data['Close'], mode='lines', name='Close Price'))
line_fig.update_layout(title='AAPL Closing Prices (2014-2024)', xaxis_title='Date', yaxis_title='Price')

# 2. Candlestick Chart
candlestick_fig = go.Figure(data=[go.Candlestick(x=data.index,
                                                 open=data['Open'],
                                                 high=data['High'],
                                                 low=data['Low'],
                                                 close=data['Close'])])
candlestick_fig.update_layout(title='AAPL Candlestick Chart (2014-2024)', xaxis_title='Date', yaxis_title='Price')

# 3. Moving Averages
data['MA20'] = data['Close'].rolling(window=20).mean()
data['MA50'] = data['Close'].rolling(window=50).mean()

ma_fig = go.Figure()
ma_fig.add_trace(go.Scatter(x=data.index, y=data['Close'], mode='lines', name='Close Price'))
ma_fig.add_trace(go.Scatter(x=data.index, y=data['MA20'], mode='lines', name='20-Day MA'))
ma_fig.add_trace(go.Scatter(x=data.index, y=data['MA50'], mode='lines', name='50-Day MA'))
ma_fig.update_layout(title='AAPL Moving Averages (2014-2024)', xaxis_title='Date', yaxis_title='Price')
```

Figure 3.2: Line chart showing trends in AAPL closing prices.

```
# 4. Volume Traded
volume_fig = go.Figure()
volume_fig.add_trace(go.Bar(x=data.index, y=data['Volume'], name='Volume'))
volume_fig.update_layout(title='AAPL Volume Traded (2014-2024)', xaxis_title='Date', yaxis_title='Volume')

# 5. Correlation Heatmap
corr_data = data[['Open', 'High', 'Low', 'Close', 'Volume']].corr()
heatmap_fig = go.Figure(data=go.Heatmap(
    z=corr_data.values,
    x=corr_data.columns,
    y=corr_data.columns,
    colorscale='Viridis'
))
heatmap_fig.update_layout(title='Correlation Heatmap')
```

Figure 3.3: Candlestick chart for daily stock performance of AAPL.

3.6 Data Transformation

The time series data is transformed to prepare it for model training. This involves creating sequences for the LSTM and CNN models, and ensuring that the data is stationary for the ARIMA model using techniques like differencing.

3.7 Modelling

Various models are implemented to predict stock prices:

- **ARIMA:** Used for time series forecasting based on the historical data.

```
# Display the results
print("Forecast Summary:\n", forecast_summary)
print("\nModel Summary:\n", model_summary)
print("\nPredictions:\n", predictions)

# Calculate residuals
residuals = model.resid

# Perform Jarque-Bera Test
jb_stats = jarque_bera(residuals)

# Print the test results
print("Jarque-Bera Test Statistic:", jb_stats[0])
print("Jarque-Bera Test p-value:", jb_stats[1])

# If residuals are non-normal, fit student-t distribution
if jb_stats[1] < 0.05:
    df, loc, scale = t.fit(residuals)
    print("Fitted Student-t Distribution Parameters:")
    print("Degrees of freedom:", df)
    print("Location parameter:", loc)
    print("Scale parameter:", scale)

fig = go.Figure()
```

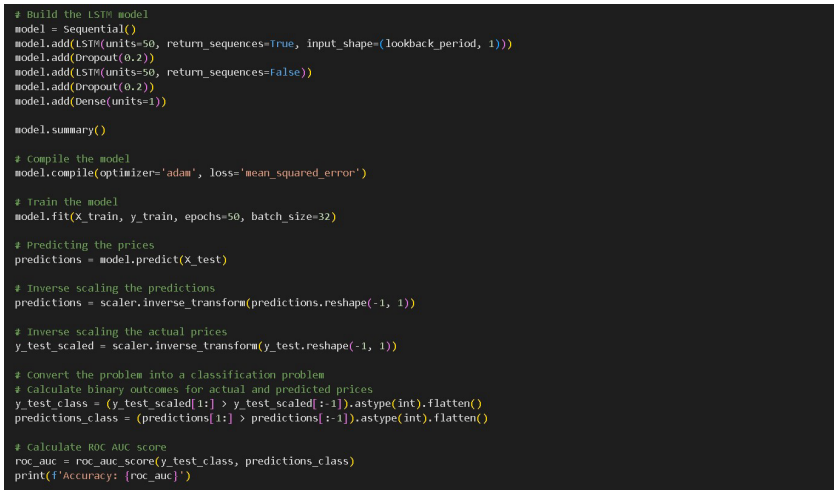
Figure 3.4: ARIMA model forecasting for AAPL.

- **Logistic Regression, SVC, and XGBoost:** These models classify the stock price movement as upward or downward based on the engineered features.

```
# Plotting heatmap of correlations greater than 0.9
plt.figure(figsize=(10, 10))
sb.heatmap(numeric_data.corr() > 0.9, annot=True, cbar=False)
plt.show()

features = data[['open-close', 'low-high', 'is_quarter_end']]
target = data['target']

scaler = StandardScaler()
features = scaler.fit_transform(features)

X_train, X_valid, Y_train, Y_valid = train_test_split(
    features, target, test_size=0.1, random_state=2022)
print(X_train.shape, X_valid.shape)

models = [LogisticRegression(), SVC(
  kernel='poly', probability=True), XGBClassifier()]

for i in range(3):
  models[i].fit(X_train, Y_train)

  print(f'{models[i]} : ')
  print('Training Accuracy : ', metrics.roc_auc_score(
    Y_train, models[i].predict_proba(X_train)[:,1]))
  print('Validation Accuracy : ', metrics.roc_auc_score(
    Y_valid, models[i].predict_proba(X_valid)[:,1]))
  print()
```

Figure 3.5: Comparison of ML models for predicting stock price movement.

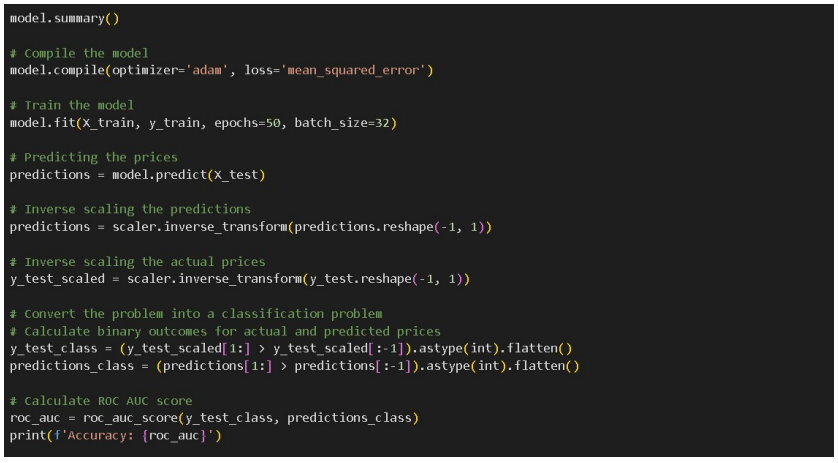
- **LSTM:** A deep learning model used to capture long-term dependencies in the stock price data.



```
# Build the LSTM model
model = Sequential()
model.add(LSTM(units=50, return_sequences=True, input_shape=(lookback_period, 1)))
model.add(Dropout(0.2))
model.add(LSTM(units=50, return_sequences=False))
model.add(Dropout(0.2))
model.add(Dense(units=1))

model.summary()

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32)

# Predicting the prices
predictions = model.predict(X_test)

# Inverse scaling the predictions
predictions = scaler.inverse_transform(predictions.reshape(-1, 1))

# Inverse scaling the actual prices
y_test_scaled = scaler.inverse_transform(y_test.reshape(-1, 1))

# Convert the problem into a classification problem
# Calculate binary outcomes for actual and predicted prices
y_test_class = (y_test_scaled[1:] > y_test_scaled[:-1]).astype(int).flatten()
predictions_class = (predictions[1:] > predictions[:-1]).astype(int).flatten()

# Calculate ROC AUC score
roc_auc = roc_auc_score(y_test_class, predictions_class)
print(f'Accuracy: {roc_auc}')
```

Figure 3.6: LSTM model predictions vs actual stock prices for AAPL.

- **CNN:** A deep learning model used to capture spatial hierarchies in the stock price data.



```
model.summary()

# Compile the model
model.compile(optimizer='adam', loss='mean_squared_error')

# Train the model
model.fit(X_train, y_train, epochs=50, batch_size=32)

# Predicting the prices
predictions = model.predict(X_test)

# Inverse scaling the predictions
predictions = scaler.inverse_transform(predictions.reshape(-1, 1))

# Inverse scaling the actual prices
y_test_scaled = scaler.inverse_transform(y_test.reshape(-1, 1))

# Convert the problem into a classification problem
# Calculate binary outcomes for actual and predicted prices
y_test_class = (y_test_scaled[1:] > y_test_scaled[:-1]).astype(int).flatten()
predictions_class = (predictions[1:] > predictions[:-1]).astype(int).flatten()

# Calculate ROC AUC score
roc_auc = roc_auc_score(y_test_class, predictions_class)
print(f'Accuracy: {roc_auc}')
```

Figure 3.7: CNN model predictions vs actual stock prices for AAPL.

3.8 Evaluation

The models are evaluated using various metrics:

- **Accuracy:** Measures the overall correctness of the model's predictions.
- **Confusion Matrix:** Provides insight into the true positives, false positives, true negatives, and false negatives.
- **ROC-AUC Score:** Assesses the model's ability to distinguish between classes.

```python
import plotly.graph_objects as go

# Data for the models
models = ['Logistic Regression', 'SVC (Poly Kernel)', 'XGBoost', 'LSTM', 'CNN']
accuracies = [0.4962, 0.5492, 0.4714, 0.5165, 0.4697]

# Create a bar chart
fig = go.Figure(data=[
    go.Bar(name='Validation Accuracy', x=models, y=accuracies, text=accuracies, textposition='auto', marker_color='indigo')
])

# Update layout for better aesthetics
fig.update_layout(
    title='Comparison of Model Accuracies',
    xaxis_title='Models',
    yaxis_title='Accuracy',
    yaxis=dict(tickformat='.2%'),
    template='plotly_dark',
    autosize=True,
    width=800,
    height=500
)

fig.show()
```

Figure 3.8: Performance comparison of all models based on accuracy.

Chapter 4

Conclusion

This current report documented how this system was built and developed for forecasting the stock price with combining machine learning (ML) & deeplearning based approaches. The system was back-tested with historical stock data of Apple Inc. — for which it showed impressive results in respect to key metrics its performance, and how well it maintained statistical independence across the strategies (i.e., feature importance). Possible future refinements could involve deploying more financial indices, real-time processing of data, and also other complex architectures in deep learning.