Configuration Manual

MSc Research Project
Masters in Data Analytics

Forecasting Tesla prices using ARIMA Models

Manchu Leela Prakash
Student ID: X23214210

School of Computing
National College of Ireland

Supervisor: EAMON NOLAN

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name:	Manchu Leela Prakash
Student ID:	X23214210
Programme:	Masters in Data Analytics **Year:** 2024
Module:	MSC RESEARCH PROJECT
Lecturer: **Submission Due Date:**	12 Dec
Project Title:	Forecasting Tesla prices using ARIMA Models
Word Count:	448

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: MANCHU LEELA PRAKASH

Date: 12 Dec

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.

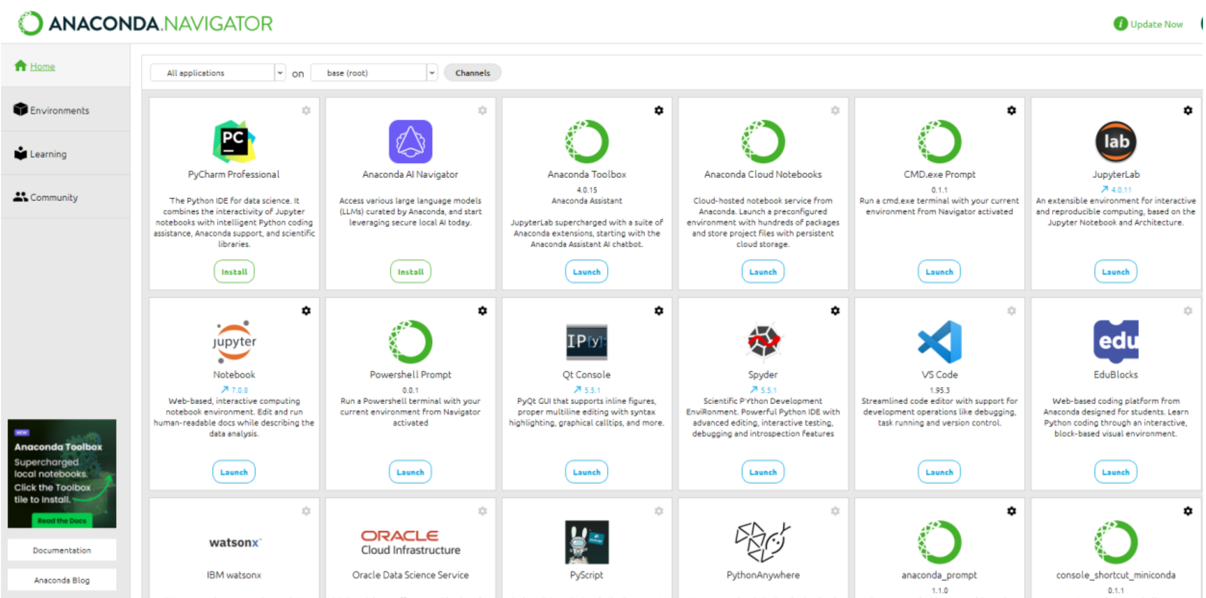
Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual

MANCHU LEELA PRAKASH
X23214210

1 Tools and Libraries used

Tool	Version	Purpose
Python	3.12.5	Core language for implementing data preprocessing, analysis, and ARIMA modeling.
Anaconda	2024.03	Manages Python environments and packages for data science workflows.
Jupyter Notebook	6.5.4	Interactive environment for coding and visualizing results.
yfinance	0.2.30	Fetches Tesla stock price data from Yahoo Finance for analysis.
pandas	2.1.1	Handles data manipulation, cleaning, and structuring for ARIMA analysis.
numpy	1.25.3	Provides numerical operations required in ARIMA modeling.
statsmodels	0.14.0	Implements ARIMA models for time-series analysis and forecasting.
sklearn	1.4.0	Assists with preprocessing and evaluation metrics in machine learning workflows.
tensorflow	2.13.0	Supports neural network comparisons if integrated into forecasting experiments.



2 Dataset used

For the project the dataset that is used is accessed through yahoo finance using yfinance library in python. With the help of this library last 5 year of stock price data of tesla is downloaded.

3 Libraries and Pre-processing steps

```
import yfinance as yf
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.stattools import adfuller
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score
from statsmodels.tsa.statespace.sarimax import SARIMAX
from pmdarima import auto_arima
from statsmodels.tsa.arima.model import ARIMA
from statsmodels.graphics.tsaplots import plot_acf, plot_pacf
```

```
# Download Tesla stock data for the last 5 years
tesla = yf.download("TSLA", period="5y")

# Display the first few rows of the data
tesla.head()
```

```
[*********************100%***********************]  1 of 1 completed
```

Price	Adj Close	Close	High	Low	Open	Volume
Ticker	TSLA	TSLA	TSLA	TSLA	TSLA	TSLA
Date						
2019-11-27	22.086000	22.086000	22.261999	21.904667	22.074667	83334000
2019-11-29	21.996000	21.996000	22.084000	21.833332	22.073999	36984000
2019-12-02	22.324667	22.324667	22.425333	21.912666	21.959999	91117500
2019-12-03	22.413334	22.413334	22.527332	22.146000	22.174667	98605500
2019-12-04	22.202000	22.202000	22.524000	22.190001	22.516666	82995000

```
tesla.isnull().sum()
```

```
Price      Ticker
Adj Close  TSLA      0
Close      TSLA      0
High       TSLA      0
Low        TSLA      0
Open       TSLA      0
Volume     TSLA      0
dtype: int64
```

```
# Feature Engineering: Adding Moving Averages
tesla['7-day MA'] = tesla['Close'].rolling(window=7).mean()
tesla['30-day MA'] = tesla['Close'].rolling(window=30).mean()
```

```
# Adding daily returns (percentage change)
tesla['Daily Return'] = tesla['Close'].pct_change()
```

```
import numpy as np
from sklearn.preprocessing import MinMaxScaler

# Normalize features (MinMaxScaler)
scaler = MinMaxScaler(feature_range=(0, 1))
tesla_scaled = tesla[['Close', 'Volume', '7-day MA', '30-day MA']].copy()
tesla_scaled[['Close', 'Volume', '7-day MA', '30-day MA']] = scaler.fit_transform(tesla_scaled[['Close', 'Volume', '7-day MA', '30-day MA']])
```

```
tesla_scaled.head()
```

Price	Close	Volume	7-day MA	30-day MA
Ticker	TSLA	TSLA		
Date				
2019-11-27	0.000232	0.060962	NaN	NaN
2019-11-29	0.000000	0.008571	NaN	NaN
2019-12-02	0.000847	0.069760	NaN	NaN
2019-12-03	0.001076	0.078225	NaN	NaN
2019-12-04	0.000531	0.060579	NaN	NaN

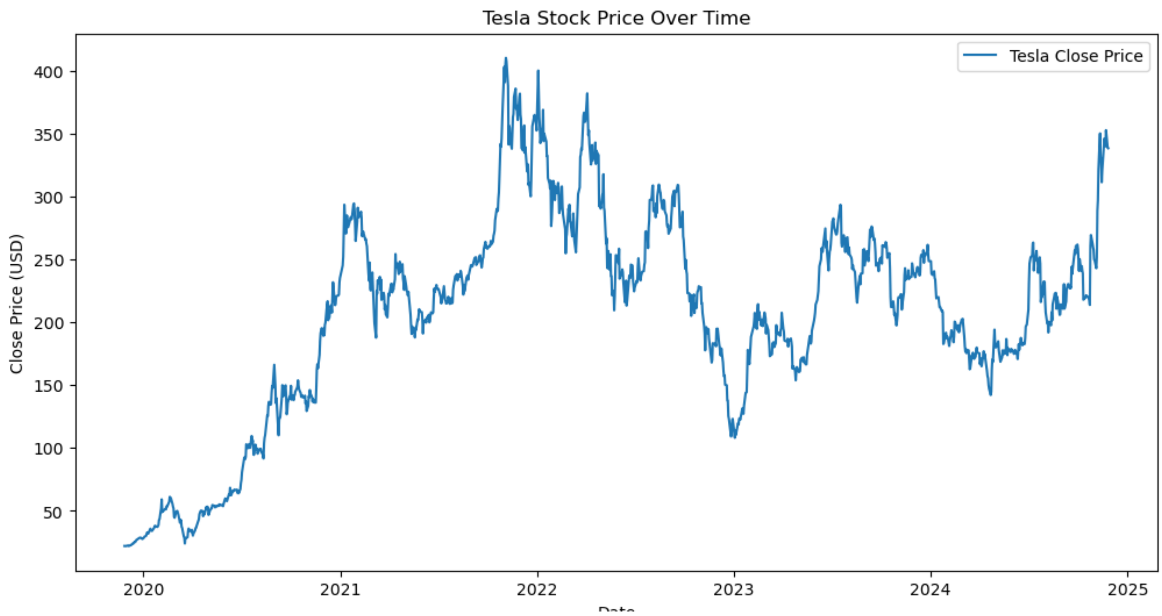
```
tesla_scaled.isnull().sum()
```

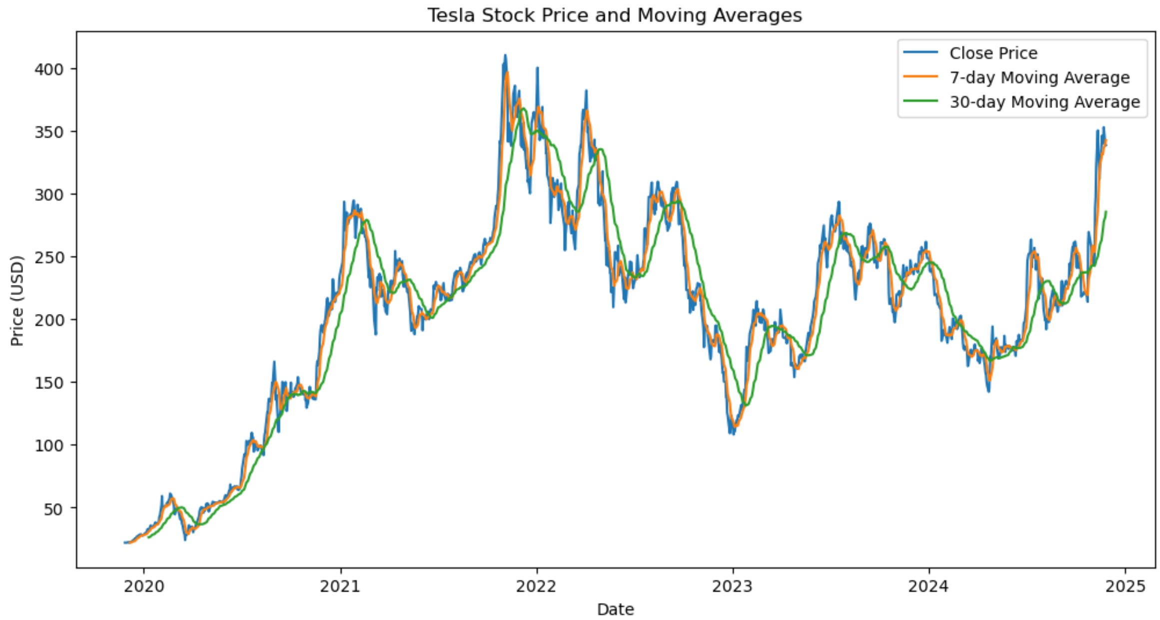
```
Price       Ticker
Close       TSLA       0
Volume      TSLA       0
7-day MA               6
30-day MA             29
dtype: int64
```

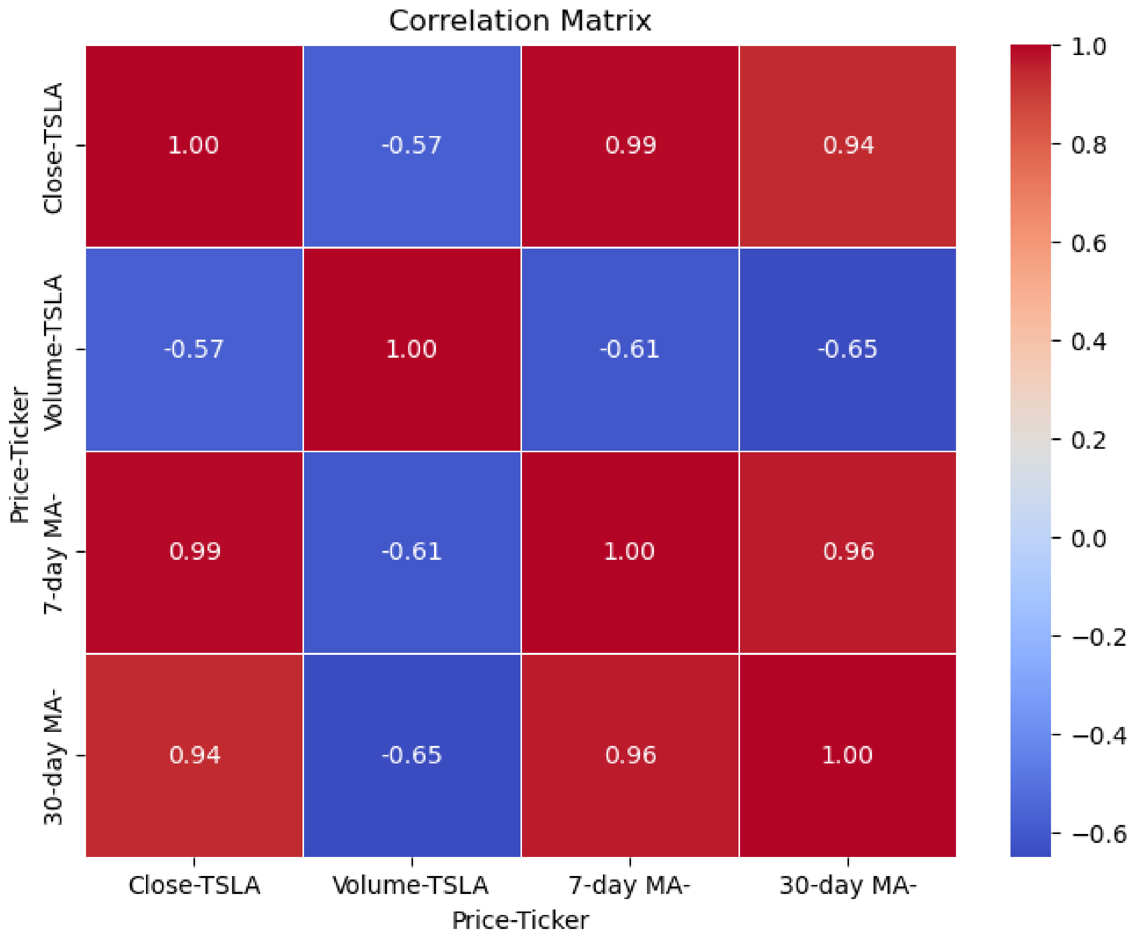
```
tesla_scaled.info()
```

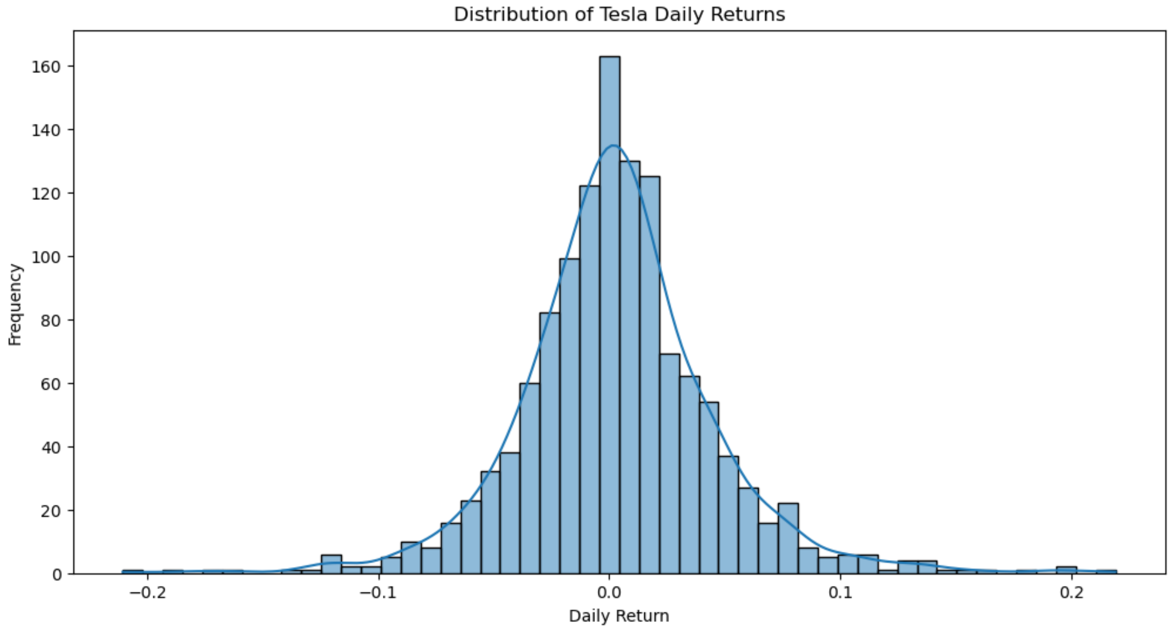
```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1258 entries, 2019-11-27 to 2024-11-26
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   (Close, TSLA)   1258 non-null   float64
 1   (Volume, TSLA)  1258 non-null   float64
 2   (7-day MA, )    1252 non-null   float64
 3   (30-day MA, )   1229 non-null   float64
dtypes: float64(4)
memory usage: 49.1 KB
```

4 EDA



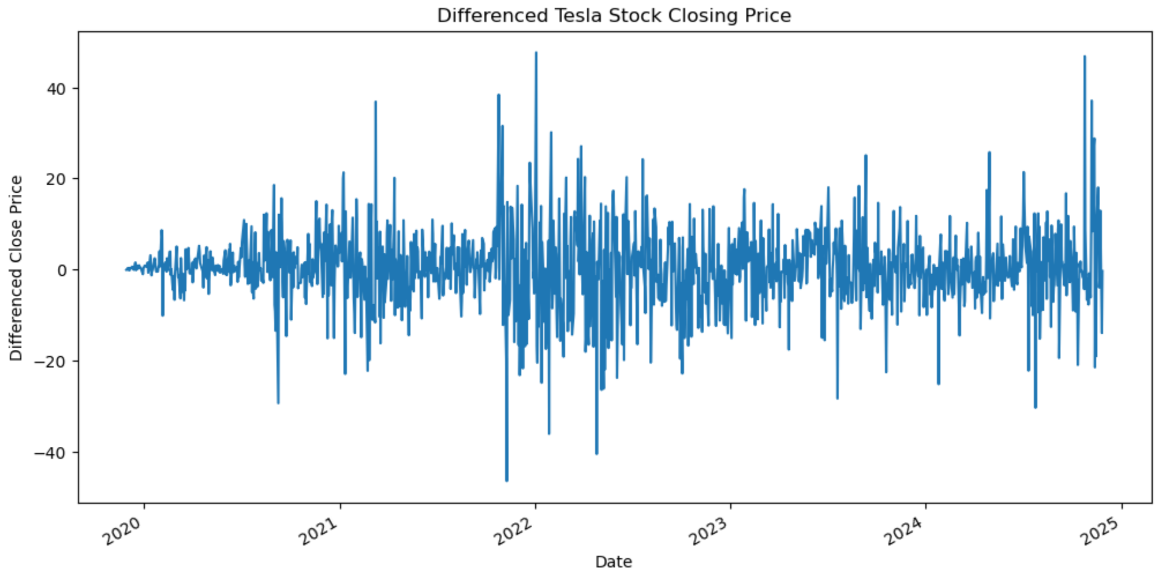

Tesla Stock Price and Moving Averages
Close Price
7-day Moving Average
30-day Moving Average
Price (USD)
400
350
300
250
200
150
100
50
2020
2021
2022
2023
2024
2025
Date


Correlation Matrix
Close-TSLA
Volume-TSLA
7-day MA-
30-day MA-
Price-Ticker
1.00
-0.57
0.99
0.94
-0.57
1.00
-0.61
-0.65
0.99
-0.61
1.00
0.96
0.94
-0.65
0.96
1.00
Close-TSLA
Volume-TSLA
7-day MA-
30-day MA-
Price-Ticker
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6




```python
from statsmodels.tsa.stattools import adfuller

# Perform Augmented Dickey-Fuller test on 'Close' column
adf_result = adfuller(tesla['Close'].dropna())

# Print the results
print('ADF Statistic:', adf_result[0])
print('p-value:', adf_result[1])
print('Critical Values:', adf_result[4])
```

```
ADF Statistic: -2.369513369639033
p-value: 0.15052652863242327
Critical Values: {'1%': -3.4356048614183443, '5%': -2.8638605461891617, '10%': -2.5680054872544145}
```

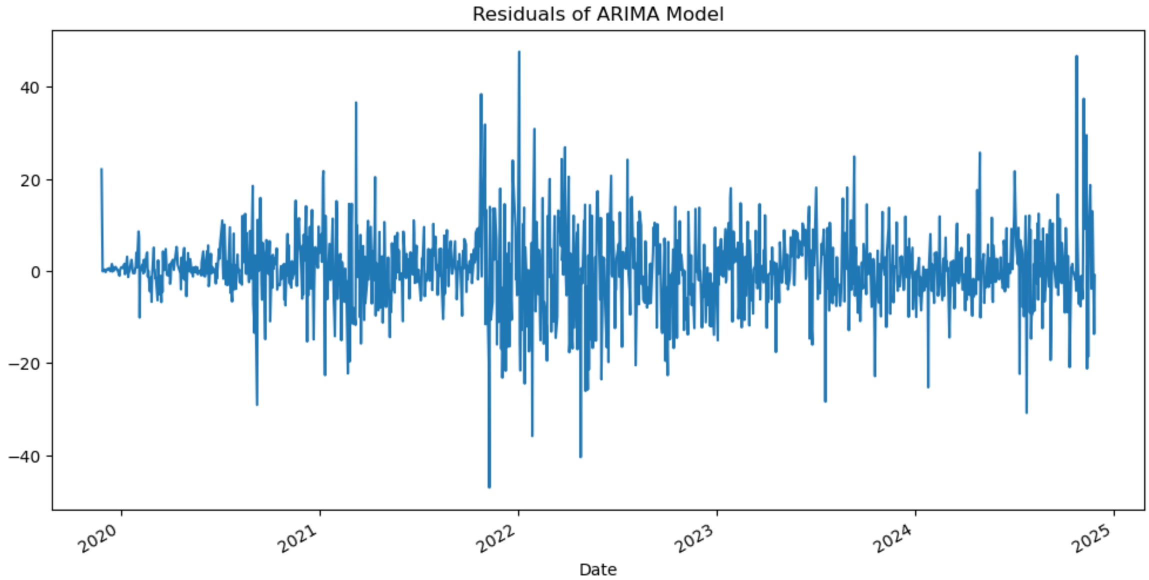


5 Model Implementation

5.1 Arima Model

```
from statsmodels.tsa.arima.model import ARIMA

# Fit the ARIMA model
model = ARIMA(tesla['Close'], order=(1, 1, 1))
model_fit = model.fit()

# Summary of the model
print(model_fit.summary())
```

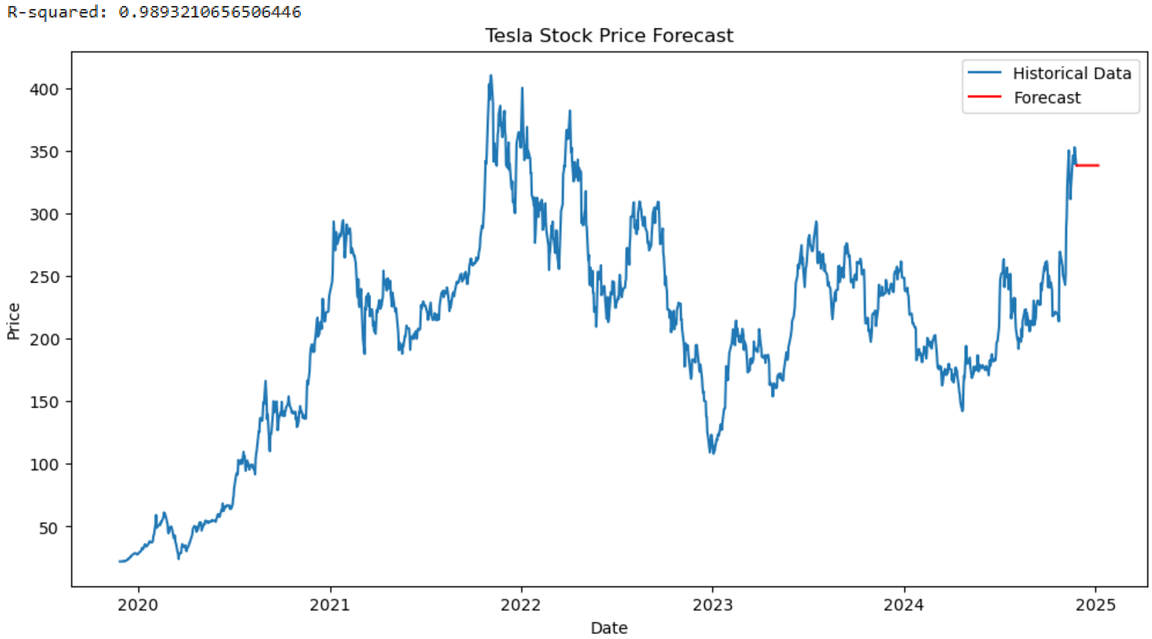
```
                               SARIMAX Results
==============================================================================
Dep. Variable:                   TSLA   No. Observations:                 1258
Model:                 ARIMA(1, 1, 1)   Log Likelihood               -4477.593
Date:                Wed, 27 Nov 2024   AIC                           8961.186
Time:                        14:52:09   BIC                           8976.595
Sample:                             0   HQIC                          8966.977
                               - 1258
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1         -0.5894      0.515     -1.145      0.252      -1.598       0.420
ma.L1          0.5641      0.527      1.070      0.285      -0.469       1.598
sigma2        72.7019      1.662     43.752      0.000      69.445      75.959
===================================================================================
Ljung-Box (L1) (Q):                   0.02   Jarque-Bera (JB):               904.01
Prob(Q):                              0.90   Prob(JB):                         0.00
Heteroskedasticity (H):               1.59   Skew:                             0.06
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.15
===================================================================================
```




```
AIC:  8961.18572663116
BIC:  8976.595176256931
```

```python
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
from sklearn.metrics import mean_squared_error

# Get the actual values
actual_values = tesla['Close'].dropna()

# Get the fitted (predicted) values from the ARIMA model
fitted_values = model_fit.fittedvalues

# The fitted values have an extra value due to the model's initialization; drop the first value of fitted_values
fitted_values = fitted_values[1:]

# Now, both actual_values and fitted_values should have the same length
mse = mean_squared_error(actual_values[1:], fitted_values)
print(f'Mean Squared Error (MSE): {mse}')

# Calculate R-squared between the actual and fitted values
r2 = r2_score(actual_values[1:], fitted_values)
print(f'R-squared: {r2}')
```

```
Mean Squared Error (MSE): 72.70241189574938
R-squared: 0.9893210656506446
```



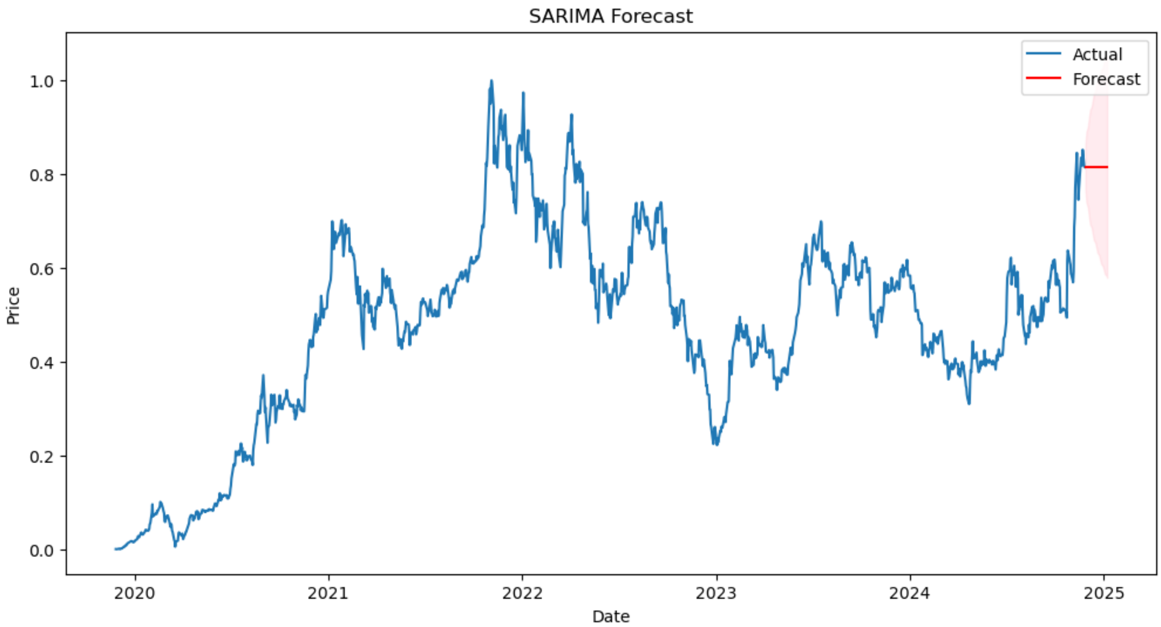
5.2 SARIMAX

```
import pandas as pd
import numpy as np
from statsmodels.tsa.statespace.sarimax import SARIMAX
from sklearn.metrics import mean_squared_error
import matplotlib.pyplot as plt
from pmdarima import auto_arima

data = tesla_scaled['Close']

# Automatically determine SARIMA parameters
auto_model = auto_arima(data, seasonal=True, m=12, trace=True, error_action='ignore', suppress_warnings=True)
print(auto_model.summary())
```

```
Performing stepwise search to minimize aic
 ARIMA(2,1,2)(1,0,1)[12] intercept   : AIC=-6019.368, Time=3.52 sec
 ARIMA(0,1,0)(0,0,0)[12] intercept   : AIC=-6026.478, Time=0.19 sec
 ARIMA(1,1,0)(1,0,0)[12] intercept   : AIC=-6023.706, Time=0.33 sec
 ARIMA(0,1,1)(0,0,1)[12] intercept   : AIC=-6023.570, Time=0.49 sec
 ARIMA(0,1,0)(0,0,0)[12]             : AIC=-6027.385, Time=0.08 sec
 ARIMA(0,1,0)(1,0,0)[12] intercept   : AIC=-6025.072, Time=0.23 sec
 ARIMA(0,1,0)(0,0,1)[12] intercept   : AIC=-6024.959, Time=0.30 sec
 ARIMA(0,1,0)(1,0,1)[12] intercept   : AIC=-6024.317, Time=0.62 sec
 ARIMA(1,1,0)(0,0,0)[12] intercept   : AIC=-6025.109, Time=0.21 sec
 ARIMA(0,1,1)(0,0,0)[12] intercept   : AIC=-6025.089, Time=0.18 sec
 ARIMA(1,1,1)(0,0,0)[12] intercept   : AIC=-6023.714, Time=0.32 sec

Best model:  ARIMA(0,1,0)(0,0,0)[12]
Total fit time: 6.476 seconds
                               SARIMAX Results
==============================================================================
Dep. Variable:                      y   No. Observations:                 1258
Model:               SARIMAX(0, 1, 0)   Log Likelihood                3014.692
Date:                Wed, 27 Nov 2024   AIC                          -6027.385
Time:                        14:52:26   BIC                          -6022.248
Sample:                             0   HQIC                         -6025.454
                               - 1258
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0005    1.1e-05     43.840      0.000       0.000       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.63   Jarque-Bera (JB):               885.95
Prob(Q):                              0.43   Prob(JB):                         0.00
Heteroskedasticity (H):               1.59   Skew:                             0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.11
```

```python
# Define SARIMA model
model = SARIMAX(data, order=(0, 1, 0), seasonal_order=(0, 0, 0, 12))

# Fit the model
sarima_model = model.fit(disp=False)

# Print model summary
print(sarima_model.summary())
```

```
                               SARIMAX Results
==============================================================================
Dep. Variable:                   TSLA   No. Observations:                 1258
Model:               SARIMAX(0, 1, 0)   Log Likelihood                3014.692
Date:                Wed, 27 Nov 2024   AIC                          -6027.385
Time:                        14:52:26   BIC                          -6022.248
Sample:                             0   HQIC                         -6025.454
                               - 1258
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
sigma2         0.0005    1.1e-05     43.840      0.000       0.000       0.001
===================================================================================
Ljung-Box (L1) (Q):                   0.63   Jarque-Bera (JB):               885.95
Prob(Q):                              0.43   Prob(JB):                         0.00
Heteroskedasticity (H):               1.59   Skew:                             0.05
Prob(H) (two-sided):                  0.00   Kurtosis:                         7.11
===================================================================================
```



```python
from sklearn.metrics import mean_squared_error, r2_score

# Get in-sample predictions
in_sample_preds = sarima_model.fittedvalues
in_sample_actuals = data

# Compute Mean Squared Error
mse_in_sample = mean_squared_error(in_sample_actuals, in_sample_preds)

# Compute R^2 Score
r2_in_sample = r2_score(in_sample_actuals, in_sample_preds)

# Display results
print(f"Mean Squared Error (MSE): {mse_in_sample:.4f}")
print(f"R^2 Score: {r2_in_sample:.4f}")
```

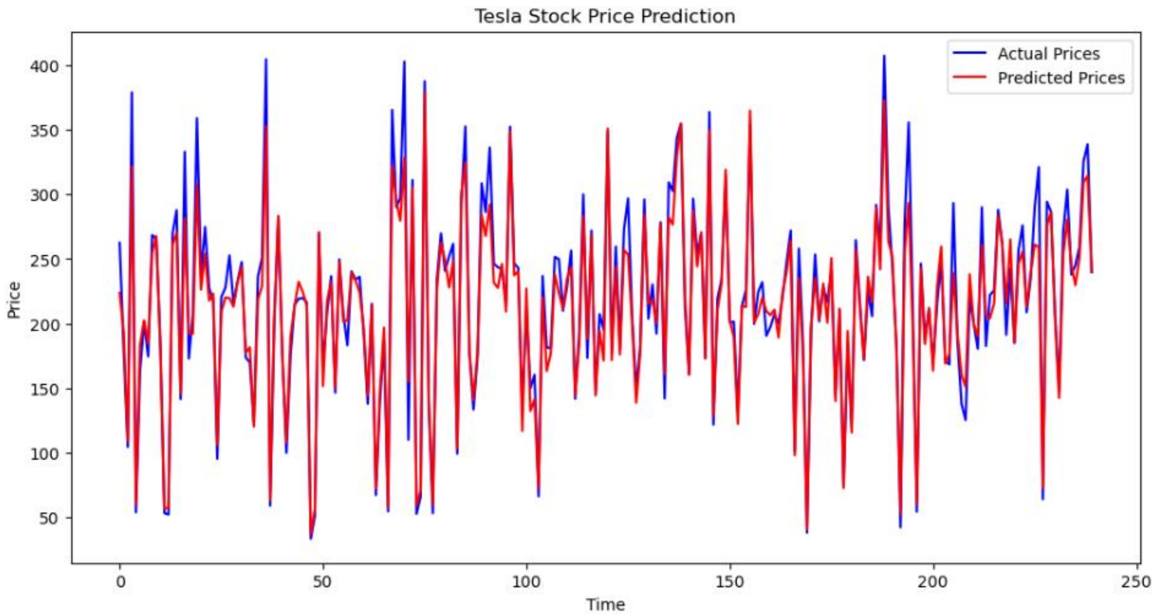
```
Mean Squared Error (MSE): 0.0005
R^2 Score: 0.9894
```

5.3 LSTM

```python
# Preparing the data for LSTM
def create_sequences(data, sequence_length):
    sequences = []
    for i in range(len(data) - sequence_length):
        seq = data[i:i + sequence_length]
        label = data[i + sequence_length]
        sequences.append((seq, label))
    return sequences

# Define the sequence length
sequence_length = 60

# Extracting Close prices for LSTM input
data = tesla_scaled['Close'].values.reshape(-1, 1)

# Creating sequences
sequences = create_sequences(data, sequence_length)
X, y = zip(*sequences)
X = np.array(X)
y = np.array(y)

# Splitting into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Build the LSTM model
model = Sequential([
    LSTM(50, return_sequences=True, input_shape=(X_train.shape[1], 1)),
    Dropout(0.2),
    LSTM(50, return_sequences=False),
    Dropout(0.2),
    Dense(25),
    Dense(1)
])

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

```
# Train the model
history = model.fit(X_train, y_train, batch_size=32, epochs=20, validation_data=(X_test, y_test))
# Predictions
predictions = model.predict(X_test)
# Rescaling predictions back to original scale
predicted_prices = scaler.inverse_transform(np.concatenate([predictions, np.zeros((predictions.shape[0], 3))], axis=1))[:, 0]
# Rescale y_test
actual_prices = scaler.inverse_transform(np.concatenate([y_test.reshape(-1, 1), np.zeros((y_test.shape[0], 3))], axis=1))[:, 0]
```

```
Epoch 1/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 7s 61ms/step - loss: 0.0887 - val_loss: 0.0065
Epoch 2/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0082 - val_loss: 0.0040
Epoch 3/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0068 - val_loss: 0.0030
Epoch 4/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0046 - val_loss: 0.0028
Epoch 5/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0043 - val_loss: 0.0026
Epoch 6/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0044 - val_loss: 0.0025
Epoch 7/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0035 - val_loss: 0.0025
Epoch 8/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0038 - val_loss: 0.0027
Epoch 9/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0037 - val_loss: 0.0021
Epoch 10/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 42ms/step - loss: 0.0038 - val_loss: 0.0021
Epoch 11/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0031 - val_loss: 0.0021
Epoch 12/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0028 - val_loss: 0.0019
Epoch 13/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0028 - val_loss: 0.0018
Epoch 14/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0026 - val_loss: 0.0021
Epoch 15/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0027 - val_loss: 0.0019
Epoch 16/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0030 - val_loss: 0.0020
Epoch 17/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0028 - val_loss: 0.0017
Epoch 18/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 41ms/step - loss: 0.0024 - val_loss: 0.0019
Epoch 19/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 38ms/step - loss: 0.0027 - val_loss: 0.0017
Epoch 20/20
30/30 ━━━━━━━━━━━━━━━━━━━━ 1s 39ms/step - loss: 0.0025 - val_loss: 0.0020
8/8 ━━━━━━━━━━━━━━━━━━━━ 1s 56ms/step
```



```python
from sklearn.metrics import mean_squared_error, r2_score

# Calculate MSE
mse = mean_squared_error(actual_prices, predicted_prices)

# Calculate R^2 Score
r2 = r2_score(actual_prices, predicted_prices)

# Display results
print(f"Mean Squared Error (MSE): {mse:.2f}")
print(f"R^2 Score: {r2:.2f}")
```

```
Mean Squared Error (MSE): 295.07
R^2 Score: 0.95
```

References

Anaconda, (2024). *Download Now.* [Online.] Accessed through: <https://www.anaconda.com/download/success>