Stock Market Prediction

Stock market prediction is a complex endeavor that draws upon various theories and concepts within the realm of finance and investment. One foundational theory is the Efficient Market Hypothesis (EMH), which posits that stock prices already incorporate all known information, making it nearly impossible to consistently outperform the market through analysis or prediction. It comes in three forms: weak, semi-strong, and strong, each reflecting different degrees of information efficiency. The Random Walk Theory, closely related to EMH, suggests that stock prices follow a random pattern, making price changes unpredictable and independent from past prices. Fundamental and technical analysis theories focus on evaluating a company's financial health, market position, and historical price patterns to estimate future stock movements, while behavioral finance delves into the psychological factors that influence investment decisions. Market sentiment and valuation models play crucial roles in forecasting, as do risk-reward tradeoffs, information asymmetry, and the impact of specific events on stock prices. By understanding these theories and concepts, investors and analysts can navigate the complexities of financial markets and make informed decisions, even though predicting stock prices remains inherently uncertain.

Linear Regression

Linear regression is a fundamental statistical method used to establish a linear relationship between one or more independent variables and a dependent variable. It is widely applied in various fields, including economics, finance, and social sciences, to understand and predict how changes in one or more factors influence the outcome of interest. In a simple linear regression model, there is a single independent variable, while multiple independent variables are considered in multiple or multivariate regression. The goal is to find the best-fitting straight line (the regression line) that minimizes the difference between the predicted values and the actual data points. This line is defined by a slope and an intercept, which provide insights into the direction and strength of the relationship between the variables. Linear regression is not only used for prediction but also for understanding the extent to which different factors affect the target variable, making it a powerful tool for data analysis and decision-making in various fields.

Implementation

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

import chart_studio.plotly as py
import plotly.graph_objs as go
from plotly.offline import plot

#for offline plotting
from plotly.offline import download_plotlyjs,
init_notebook_mode, plot, iplot
init_notebook_mode(connected=True)
```

```
import pandas as pd
tesla = pd.read_csv('C:\\Users\\OM\\Downloads\\New
folder\\tesla.csv')
tesla.head()
```

output:

	Date	Open	High	Low	Close	Adj Close	Volume
0	29-06-2010	19.000000	25.00	17.540001	23.889999	23.889999	18766300
1	30-06-2010	25.790001	30.42	23.299999	23.830000	23.830000	17187100
2	01-07-2010	25.000000	25.92	20.270000	21.959999	21.959999	8218800
3	02-07-2010	23.000000	23.10	18.709999	19.200001	19.200001	5139800
4	06-07-2010	20.000000	20.00	15.830000	16.110001	16.110001	6866900

```
tesla.info()
```

output:

```
[3]
...  <class 'pandas.core.frame.DataFrame'>
     RangeIndex: 2193 entries, 0 to 2192
     Data columns (total 7 columns):
      #   Column     Non-Null Count  Dtype
     ---  ------     --------------  -----
      0   Date       2193 non-null   object
      1   Open       2193 non-null   float64
      2   High       2193 non-null   float64
      3   Low        2193 non-null   float64
      4   Close      2193 non-null   float64
      5   Adj Close  2193 non-null   float64
      6   Volume     2193 non-null   int64
     dtypes: float64(5), int64(1), object(1)
     memory usage: 120.1+ KB
```

```
tesla['Date'] = pd.to_datetime(tesla['Date'])
```

```
print(f'Dataframe contains stock prices between {tesla.Date.min()} {tesla.Date.max()}')
print(f'Total days = {(tesla.Date.max()  - tesla.Date.min()).days} days')
```

output:

```
Dataframe contains stock prices between 2010-06-29 00:00:00 2019-03-15 00:00:00
Total days = 3181 days
```

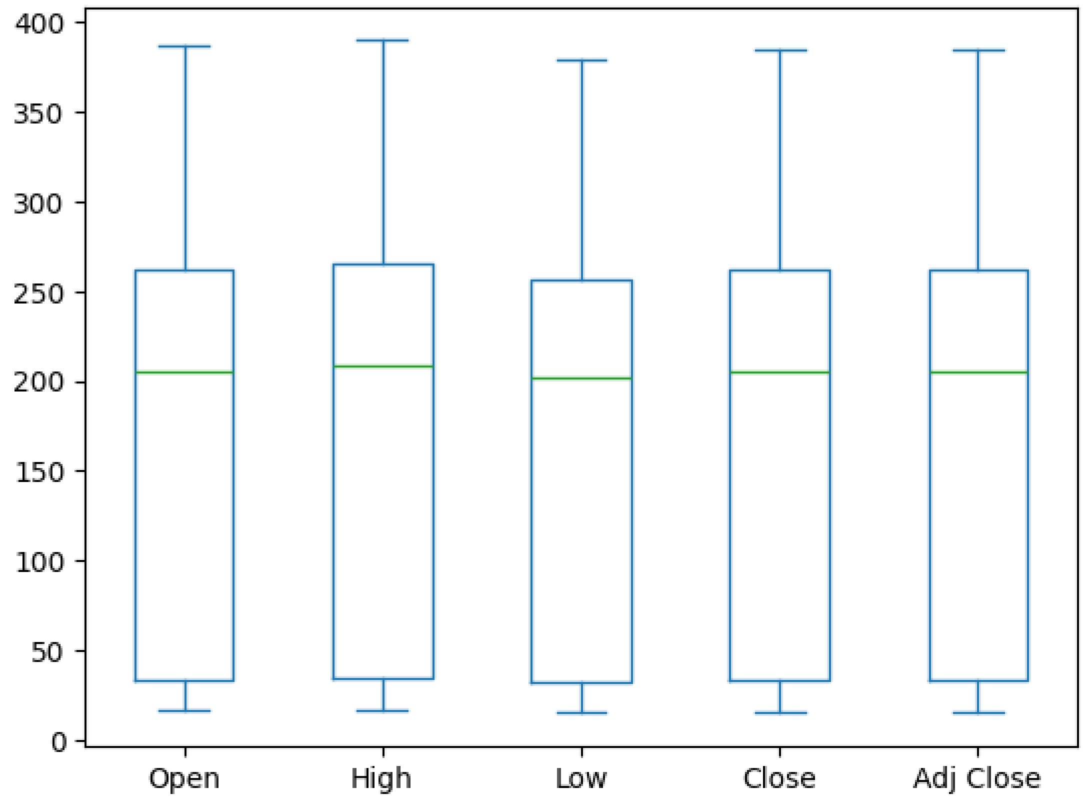
```
tesla.describe()
```

Output:

	Date	Open	High	Low	Close	Adj Close	Volume
count	2193	2193.000000	2193.000000	2193.000000	2193.000000	2193.000000	2.193000e+03
mean	2014-11-04 14:37:15.841313024	175.652882	178.710262	172.412075	175.648555	175.648555	5.077449e+06
min	2010-06-29 00:00:00	16.139999	16.629999	14.980000	15.800000	15.800000	1.185000e+05
25%	2012-08-29 00:00:00	33.110001	33.910000	32.459999	33.160000	33.160000	1.577800e+06
50%	2014-11-04 00:00:00	204.990005	208.160004	201.669998	204.990005	204.990005	4.171700e+06
75%	2017-01-09 00:00:00	262.000000	265.329987	256.209991	261.739990	261.739990	6.885600e+06
max	2019-03-15 00:00:00	386.690002	389.609985	379.350006	385.000000	385.000000	3.716390e+07
std	NaN	115.580903	117.370092	113.654794	115.580771	115.580771	4.545398e+06

```
import pandas as pd
tesla = pd.read_csv('C:\\Users\\OM\\Downloads\\New folder\\tesla.csv')
tesla[['Open','High','Low','Close','Adj Close']].plot(kind='box')
```

Output:




```python
# Setting the layout for our plot
import plotly.graph_objs as go
import pandas as pd
tesla = pd.read_csv('C:\\Users\\OM\\Downloads\\New folder\\tesla.csv')
layout = go.Layout(
    title='Stock Prices of Tesla',
    xaxis=dict(
        title='Date',
        titlefont=dict(
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    ),
    yaxis=dict(
        title='Price',
        titlefont=dict(
```

```
            family='Courier New, monospace',
            size=18,
            color='#7f7f7f'
        )
    )
)

tesla_data = [{'x':tesla['Date'],
'y':tesla['Close']}]
plot = go.Figure(data=tesla_data, layout=layout)
```

```
#plot(plot) #plotting offline
import plotly.offline as pyo
pyo.plot(plot)
```

Output:

```
# Building the regression model
from sklearn.model_selection import
train_test_split

#For preprocessing
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler

#For model evaluation
from sklearn.metrics import mean_squared_error as
mse
from sklearn.metrics import r2_score
```

```
#Split the data into train and test sets
import numpy as np
import pandas as pd
tesla = pd.read_csv('C:\\Users\\OM\\Downloads\\New
folder\\tesla.csv')
from sklearn.model_selection import
train_test_split
X = np.array(tesla.index).reshape(-1,1)
Y = tesla['Close']
X_train, X_test, Y_train, Y_test =
train_test_split(X, Y, test_size=0.3,
random_state=101)
```

```
# Feature scaling
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
```

```
from sklearn.linear_model import LinearRegression
```

```python
#Creating a linear model
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, Y_train)
```

```python
#Plot actual and predicted values for train dataset
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train, Y_train)
import plotly.graph_objs as go
layout=go.Layout(
    xaxis=dict(title='day'),
    yaxis=dict(title='price'),
    title = 'stock prices of tesla'
)
trace0 = go.Scatter(
    x = X_train.T[0],
    y = Y_train,
    mode = 'markers',
    name = 'Actual'
)
trace1 = go.Scatter(
    x = X_train.T[0],
    y = lm.predict(X_train).T,
    mode = 'lines',
    name = 'Predicted'
)
tesla_data = [trace0,trace1]
layout.xaxis.title.text = 'Day'
plot2 = go.Figure(data=tesla_data, layout=layout)
```

```python
import plotly.offline as pyo
pyo.plot(plot2)
```

Output:


```
#Calculate scores for model evaluation
scores = f'''
{'Metric'.ljust(10)}{'Train'.center(20)}{'Test'.center(20)}
{'r2_score'.ljust(10)}{r2_score(Y_train, lm.predict(X_train))}\t{r2_score(Y_test, lm.predict(X_test))}
{'MSE'.ljust(10)}{mse(Y_train, lm.predict(X_train))}\t{mse(Y_test, lm.predict(X_test))}
'''
print(scores)
```

Output:

```
0]

Metric          Train                 Test
r2_score  0.8658871776828707    0.8610649253244574
MSE       1821.3833862936174    1780.987539418845
```

Conclusion:

The task of predicting Tesla, Inc. (TSLA) stock prices is a challenging and complex undertaking. While this study employed various machine learning models, including time series analysis and regression, to forecast Tesla's future stock prices, it is important to acknowledge the inherent unpredictability of financial markets. Our analysis has provided valuable insights into historical price trends and their potential relevance for future forecasts. Nevertheless, it is essential to recognize that external factors, market sentiment, and unforeseen events can significantly impact stock prices, making precise predictions elusive.

The results of our models demonstrate the potential for informed decision-making, particularly in terms of identifying trends and patterns that may guide investment strategies. However, these models are not infallible, and there are limitations to their accuracy and reliability. Investors and analysts should approach stock market predictions with caution and use them as one of many tools for financial decision-making.

As financial markets continue to evolve and new data sources become available, the field of stock price prediction will continue to advance. Future research may explore the integration of sentiment analysis, alternative data sources, and improved machine learning algorithms to enhance forecasting accuracy. Ultimately, a comprehensive investment strategy should consider a wide range of factors, including risk management, diversification, and a long-term perspective, in addition to predictive models. Successful investment decisions require a combination of data analysis, market understanding, and prudent risk management in the ever-changing landscape of the stock market.