**The NETFLIX Revolution**

**Appendix**

1. Introduction
2. Data Description
3. Exploratory Data Analysis
   1. Importing the libraries
   2. Importing the data
   3. Describing the data
   4. Finding the missing values
   5. Visualizations
   6. Corelations
4. Data transformations and cleaning
5. K-Nearest Neighbors (KNN) algorithm
6. Summary
7. References
8. Introduction:

Predicting stock prices is an important and difficult subject of study. The economies of the developed countries are measured according to their economies of power. In this assignment, I applied k-nearest neighbor algorithm in order to predict stock prices for a population of Netflix’s stock market data. According to the results, the kNN algorithm is robust with small error ratio; consequently the results were rational and also reasonable.

In addition, depending on the actual stock prices data; the prediction results were close and almost close to actual stock prices.

1. Data Description:

The sample data was extracted from ‘The Wall Street Journal’. The study sample includes stock data of the company Netflix from the period 01/04/10 to 05/01/20 . The dataset population has approximately 2600 records with six attributes including Date, Open, High, Low, Close, Volume. A brief data analysis is presented with the fundamental concepts of data attributes. All the attributes are included in the data analysis. ‘Close’ is the main factor that affects the prediction process for a specific stock based on kNN algorithm. The kNN algorithm is applied on all the records to estimate predicted values for each stock.

1. Exploratory Data Analysis:
   1. Importing the libraries:

import numpy as np

import pandas as pd

from pandas import Series, DataFrame

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.neighbors import KNeighborsClassifier

from sklearn.metrics import accuracy\_score

from sklearn.model\_selection import GridSearchCV

plt.style.use('ggplot')

%matplotlib inline

* 1. Importing the data:

Code:

Netflix=pd.read\_csv('/content/HistoricalPrices (1).csv')

Netflix.shape

Output:

A picture containing drawing

Description automatically generated

Code:

Netflix.head()

Output:

A screenshot of a cell phone

Description automatically generated

* 1. Describing the data:

Code:

Netflix.describe()

Output:

A screenshot of a cell phone

Description automatically generated

* 1. Finding the missing values:

Code:

Netflix.isnull().sum()

Output:

A screenshot of a cell phone

Description automatically generated

Code:

Netflix.dtypes

Output:

A screenshot of a cell phone

Description automatically generated

* 1. Visualizations:

The below histograms give us the information about the attributes in the dataset. These attributes are sufficient to perform our analysis on the dataset.

Code:

Netflix.hist(figsize=(10,8), color = "crimson")

Output:

A picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generated

A picture containing drawing

Description automatically generated

A picture containing drawing, clock

Description automatically generated

Code:

Netflix.set\_index(['Date\_f'])[' High'].plot(kind = 'line')

Output:

A close up of a mans face

Description automatically generated

Code:

Netflix.set\_index(['Date\_f'])[' Low'].plot(kind = 'line')

Output:

A close up of a mans face

Description automatically generated

Code:

Netflix.plot(x='Date\_f',y=[' Open',' Close']).plot(kind = 'line')

Output:

A screenshot of a cell phone

Description automatically generated

Code:

Netflix.set\_index(['Date\_f'])[' Volume'].plot(kind = 'line')

Output:

A close up of a logo

Description automatically generated

* 1. Corelations:

Code:

Netflix\_df\_num\_corr = Netflix\_df.corr()[' Close'][:-1]

list = Netflix\_df\_num\_corr[abs(Netflix\_df\_num\_corr) > 0.5].sort\_values(ascending=False)

print("There is {} strongly correlated values with Close:\n{}".format(len(list), list))

Output:

A picture containing bird

Description automatically generated

Code:

for i in range(0, len(Netflix\_df.columns), 5):

sns.pairplot(data=Netflix\_df,

x\_vars=Netflix\_df.columns[i:i+5],

y\_vars=[' Close'])

Output:

A close up of a map

Description automatically generated

1. Data transformations and cleaning:

Code:

Netflix\_df = Netflix.dropna()

Netflix\_df = Netflix\_df[['Date\_f', ' Open', ' High', ' Low', ' Close']]

Netflix\_df.set\_index(['Date\_f'], inplace=True)

Netflix\_df.head()

Output:

A screenshot of a cell phone

Description automatically generated

1. K-Nearest Neighbors (KNN) algorithm:

K-Nearest Neighbors (KNN) is one of the simplest algorithms used in Machine Learning for regression and classification problem. KNN algorithms use data and classify new data points based on similarity measures. Classification is done by a majority vote to its neighbors. The data is assigned to the class which has the nearest neighbors. As you increase the number of nearest neighbors, the value of k, accuracy might increase.

So for the algorithm x is used to denote a predictor while y is used to denote the target that is trying to be predicted. A training dataset is used to capture the relationship between x and y so that unseen observations of x can be used to confidently predict corresponding y outputs.

Define Predictor Variable:

Code:

Netflix\_df['Open-Close'] = Netflix\_df[' Open'] - Netflix\_df[' Close']

Netflix\_df['High-Low'] = Netflix\_df[' High'] - Netflix\_df[' Low']

Netflix\_df= Netflix\_df.dropna()

x= Netflix\_df[['Open-Close', 'High-Low']]

x.head()

Output:

A screenshot of a cell phone

Description automatically generated

Code:

x.tail()

Output:

A screenshot of a cell phone

Description automatically generated

Define Target Variables

Code:

y= np.where(Netflix\_df[' Close'].shift(-1)> Netflix\_df[' Close'], 1, -1)

x

Output:

A screenshot of a cell phone

Description automatically generated

Split the Dataset

Code:

split\_percentage= 0.7

split= int(split\_percentage\*len(Netflix\_df))

x\_train= x[-split:]

y\_train= y[-split:]

x\_test= x[:-split]

y\_test= y[:-split]

x\_test

Output:

A screenshot of a cell phone

Description automatically generated

Scores for Test and Train sets:

Code:

test\_scores = []

train\_scores = []

for i in range(1,15):

knn = KNeighborsClassifier(i)

knn.fit(x\_train,y\_train)

train\_scores.append(knn.score(x\_train,y\_train))

test\_scores.append(knn.score(x\_test,y\_test))

print(train\_scores)

print(test\_scores)

Output:

[1.0, 0.9021440351841671, 0.8982957669048928, 0.880153930731171, 0.8851017042330951, 0.8763056624518967, 0.8763056624518967, 0.8746564046179219, 0.8752061572292469, 0.8746564046179219, 0.880153930731171, 0.8735568993952721, 0.8768554150632215, 0.8735568993952721]

[0.8271446862996159, 0.8476312419974392, 0.8553137003841229, 0.8629961587708067, 0.8693982074263764, 0.8719590268886044, 0.8745198463508322, 0.8719590268886044, 0.8655569782330346, 0.8668373879641486, 0.8693982074263764, 0.8681177976952625, 0.8668373879641486, 0.8681177976952625]

Code:

#In case of classifier like knn the parameter to be tuned using cross validation is n\_neighbors

param\_grid = {'n\_neighbors':np.arange(1,70)}

knn = KNeighborsClassifier()

knn\_cv= GridSearchCV(knn,param\_grid,cv=10)

knn\_cv.fit(x\_train,y\_train)

print("Best Score:" + str(knn\_cv.best\_score\_))

print("Best Parameters: " + str(knn\_cv.best\_params\_))

Output:

A close up of a logo

Description automatically generated

Result Visualization:

Code:

sns.set()

plt.figure(figsize=(12,5))

p = sns.lineplot(range(1,15),train\_scores,marker='\*',label='Train Score')

p = sns.lineplot(range(1,15),test\_scores,marker='o',label='Test Score')

Output:

A picture containing table, white, lot, room

Description automatically generated

Code:

knn = KNeighborsClassifier(7)

knn.fit(x\_train,y\_train)

knn.score(x\_test,y\_test)

Output:

A picture containing drawing

Description automatically generated

Code:

Netflix\_df['Predicted']= knn.predict(x)

Netflix\_df['Netflix\_Returns']= np.log(Netflix\_df[' Close'] / Netflix\_df[' Close'].shift(1))

Cummulative\_Netflix\_Returns= Netflix\_df[:-split]['Netflix\_Returns'].cumsum()\*100

Netflix\_df['Strategy\_Returns']= Netflix\_df['Netflix\_Returns'] \* Netflix\_df['Predicted'].shift(1)

Cummulative\_Strategy\_Returns= Netflix\_df[:-split]['Strategy\_Returns'].cumsum()\*100

Cummulative\_Strategy\_Returns

Output:

A screenshot of a cell phone

Description automatically generated

Code:

plt.figure(figsize=(10,5))

plt.plot(Cummulative\_Netflix\_Returns, color='r', label= 'Netflix\_Returns')

plt.plot(Cummulative\_Strategy\_Returns, color='b', label='Strategy\_Returns')

plt.legend()

plt.tight\_layout()

plt.show()

Output:

A close up of a map

Description automatically generated

Performance and fault measure:

Code:

y\_test\_pad = np.pad(y\_test, (0, len(knn.predict(x))-len(y\_test)), 'constant')

len(y\_test\_pad)

Output:

A picture containing drawing

Description automatically generated

Code:

rms=np.sqrt(np.mean(np.power((y\_test\_pad-np.array(knn.predict(x))),2)))

rms

Output:

A picture containing drawing

Description automatically generated

Code:

from sklearn.metrics import confusion\_matrix

#let us get the predictions using the classifier we had fit above. Creating the confusion Matrix

y\_pred = knn.predict(x\_test)

y\_pred

confusion\_matrix(y\_test,y\_pred)

pd.crosstab(y\_test, y\_pred, rownames=['True'], colnames=['Predicted'], margins=True)

Output:

A screenshot of a cell phone

Description automatically generated

Code:

y\_pred = knn.predict(x\_test)

y\_pred = knn.predict(x\_test)

from sklearn import metrics

cnf\_matrix = metrics.confusion\_matrix(y\_test, y\_pred)

p = sns.heatmap(pd.DataFrame(cnf\_matrix), annot=True, cmap="gist\_heat" ,fmt='g')

plt.title('Confusion matrix', y=1.1)

plt.ylabel('Actual ')

plt.xlabel('Predicted ')

Output:

A screenshot of a cell phone

Description automatically generated

1. Summary:

It has been two decades that Netflix has set its foot in the market, but it has never failed to surprise us with its popularity among the people. While Netflix (NFLX) continues to be a market leader in the streaming market. The below plot gives a gist of the opening and closing stocks of Netflix in the past decade

A screenshot of a cell phone

Description automatically generated

The trading strategy that I have used to predict the market for Netflix during the COVID 19 pandemic is reasonably accurate. As we can see, the model has the highest accuracy when k = 7. It is often extremely hard to predict the price of stocks. Even the 2.5% improvement over random guessing can make a difference given the amount of money at stake.

A picture containing table, white, lot, room

Description automatically generated

Netflix accounts for 19% of all worldwide online video subscriptions. Based on its current worldwide subscription numbers (167 million), that means Netflix commands 19% of the global digital streaming market.

Netflix has remained true to itself and to the vision of its founders, building it’s reputation as an impeccable innovator. It has also managed to hold the flexibility that all growing companies need to effectively respond to a rapidly changing marketplace without losing the focus it needed to reach to the pinnacle.

Netflix’s stock dropped over 22%, from 383.79 per share to 298.84 per share between March 4 and March 16. However, the stock price had jumped back up to 375.50 by March 31, recovering the vast majority of what it had lost. The reason behind Netflix’s recovery is the hundreds of millions of individuals worldwide found themselves at home in quarantine. The below plot shows that the prediction that we made is almost flexible during the pandemic.

A close up of a map

Description automatically generated

The below confusion matrix concludes the accuracy of our prediction. It describes the performance of a the classification model that we have used

A screenshot of a cell phone

Description automatically generated

1. References:

Dataset: <https://www.wsj.com/market-data/quotes/NFLX/historical-prices>