Sri Lanka Institute of Information Technology

Assignment – 02

Machine Learning (IT4060)

2022

Stock Price Prediction using Different Machine Learning Models and Compare the Performance of the Models

Submitted by:

Student ID	Student Name
IT18233704	Yamasinghe N. R
IT18156652	Cooray M.D.D.M
IT18213072	Ranasinghe Y. S
IT18209280	Dissanayake D.M.Y. S

TABLE OF CONTENT

1 DESCRIPTION OF THE PROBLEM ADDRESSED

The stock price forecast is one of the most preferred topics and most interesting topics in the science industry. However, stock market price forecasts are challenging compared to other price forecasting case studies. Many scholars and industry experts have come to a greater consensus a decade. They have studied stocks in various fields such as Computer Science, Economics, Business Arithmetic, and Marketing price forecasts. The stock has been identified according to them as a random walking behavior at market prices. The sudden rises and falls have been the main reason behind stock market price forecasting being a big challenge. An efficient and accurate.

stock market forecasting model will help managers, investors, and decision-makers make the right decisions regarding their investments. Machine learning techniques used to predict stock prices include the analysis of historical data to predict the likelihood of a future event or to predict future performance. This is done by looking at patterns of data that include current and past information and finding the most suitable predictive models.

1.1 What is the Stock Market?

A stock market is a public market where you can buy and sell shares for publicly listed companies. Shares, also known as shares, represent the ownership of a company. A stock exchange is an intermediary that allows you to buy and sell shares.

1.2 Importance of Stock Market

- Stock markets help companies to raise capital.
- It helps generate personal wealth.
- Stock markets serve as an indicator of the state of the economy.
- It is a widely used source for people to invest money in companies with high growth potential.

1.3 Stock Price Prediction

Stock Price Prediction using machine learning helps you discover the future value of company stock and other financial assets traded on an exchange. The entire idea of predicting stock prices is to gain significant profits. Predicting how the stock market will perform is a hard task to do. There are other factors involved in the prediction, such as physical and psychological factors, rational and irrational behavior, and so on. All these factors combine to make share prices dynamic and volatile. This makes it very difficult to predict stock prices with high accuracy.

2 DATASET

2.1 Description of the Data Set

The yahoo finance data has information from 2014 to 218. There are ten columns. Yahoo! Finance is a media property that is part of the Yahoo! network. It provides financial news, data and commentary including stock quotes, press releases, financial reports, and original content. It also offers some online tools for personal finance management.

The Open column tells the price at which a stock started trading when the market opened on a particular day. The Close column refers to the price of an individual stock when the stock exchange closed the market for the day. The High column depicts the highest price at which a stock traded during a period. The Low column tells the lowest price of the period. Volume is the total amount of trading activity during a period of time.

Description	This dataset is contain in yahoo finance site.
Data Set Link	https://finance.yahoo.com/quotes/OCR,dataset/view/v1/
Size of Data	12570
Related Task	Stock Market Prediction
Data Set Stored In	Github

2.2 Data Set Parameters

Characteristic	Information
Number of columns	10
Number of rows	1257
Columns	Open, High, Low, Close, Adj Close, Volume, Increase/Decrease, Buy Sell on Open, Buy Sell, Return

2.3 Sample Images of the Dataset

```
#View Dataset
dataset
```

Date	Open	High	Low	Close	Adj Close	Volume	Increase/Decrease	Buy_Sell_on_Open	Buy_Sell	Return
2014-01-03	3.980000	4.000000	3.880000	4.000000	4.000000	22887200	1	1	1	0.012658
2014-01-06	4.010000	4.180000	3.990000	4.130000	4.130000	42398300	1	1	1	0.032500
2014-01-07	4.190000	4.250000	4.110000	4.180000	4.180000	42932100	0	1	-1	0.012106
2014-01-08	4.230000	4.260000	4.140000	4.180000	4.180000	30678700	0	-1	-1	0.000000
2014-01-09	4.200000	4.230000	4.050000	4.090000	4.090000	30667600	0	-1	1	-0.021531
...	...	...	...	...	...	...	...	...	...	...
2018-12-24	16.520000	17.219999	16.370001	16.650000	16.650000	62933100	1	1	1	-0.016539
2018-12-26	16.879999	17.910000	16.030001	17.900000	17.900000	108811800	1	1	-1	0.075075
2018-12-27	17.430000	17.740000	16.440001	17.490000	17.490000	111373000	0	1	1	-0.022905
2018-12-28	17.530001	18.309999	17.139999	17.820000	17.820000	109214400	0	1	1	0.018868
2018-12-31	18.150000	18.510000	17.850000	18.459999	18.459999	84732200	0	-1	-1	0.035915

1257 rows × 10 columns

```
# See data types of the Columns

dataset.dtypes

Open                   float64
High                   float64
Low                    float64
Close                  float64
Adj Close              float64
Volume                   int64
Increase/Decrease        int64
Buy_Sell_on_Open         int64
Buy_Sell                 int64
Return                 float64
dtype: object
```

3 SELECTED MACHINE LEARNING ARCHITECTURES

To predict the stock price, we use the Machine Learning Models below and finally compare the accuracy values and the loss function values to find the best model.

1. Linear Regression
2. Decision Tree Regression
3. Logistic Regression
4. Bayesian Ridge Regression

Linear Regression: Linear regression attempts to model the relationship between two variables by fitting a linear equation to observed data. One variable is considered to be an explanatory variable, and the other is considered to be a dependent variable. For example, a modeler might want to relate the weights of individuals to their heights using a linear regression model.

Decision Tree Regression: Decision tree builds regression or classification models in the form of a tree structure. It breaks down a dataset into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes.

Logistic Regression: In statistics, the logistic model is a statistical model that models the probability of one event taking place by having the log-odds for the event be a linear combination of one or more independent variables. In regression analysis, logistic regression is estimating the parameters of a logistic model.

Bayesian Ridge Regression: Bayesian regression allows a natural mechanism to survive insufficient data or poorly distributed data by formulating linear regression using probability distributors rather than point estimates.

4 METHODOLOGY

We used TensorFlow, Keras as main framework for this and use normal python libraries like matplotlib, NumPy, pandas, sklearn to other purposes. They are an open-source machine learning library for Python, mainly developed by the Facebook AI Research team. These are the main steps we followed to build and train the models.

1. Import Libraries
2. Load data into a Data Frame
3. Data set Cleaning
4. Data set Preposing
5. Analyze the Data
6. Prepare Dataset
7. Create Model
8. Instantiate Model
9. Instantiate Loss
10. Instantiate Optimizer
11. Training the Model
12. Prediction

5 IMPLEMENTATION

5.1 Import Libraries

Linear Regression

```
# yahoo_finance is used to fetch data
!pip install yfinance
```

```
# Import Libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt

import warnings
warnings.filterwarnings("ignore")

# yahoo_finance is used to fetch data
import yfinance as yf
yf.pdr_override()
```

Decision Tree Regression

```
# yahoo_finance is used to fetch data
!pip install yfinance
```

```
# Import Libraries
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

# yahoo_finance is used to fetch data
import yfinance as yf
yf.pdr_override()

# MATPLOTLIB & SEABORN FOR GRAPH-PLOTTING
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Logistic Regression

```
# yahoo_finance is used to fetch data
!pip install yfinance
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

!pip install yfinance
import yfinance as yf
yf.pdr_override()
```

```
# MATPLOTLIB & SEABORN FOR GRAPH-PLOTTING
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Bayesian Ridge Regression

```
# yahoo_finance is used to fetch data
!pip install yfinance


import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

import warnings
warnings.filterwarnings("ignore")

!pip install yfinance
import yfinance as yf
yf.pdr_override()

# MATPLOTLIB & SEABORN FOR GRAPH-PLOTTING
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

5.2 Load data into a Data Frame

```
# input
symbol = 'AMD'
start = '2014-01-01'
end = '2019-01-01'

# Read data
dataset = yf.download(symbol,start,end)

# View data set
dataset.head()
```

```
# View data set
dataset.head()
```

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

Date	Open	High	Low	Close	Adj Close	Volume
2014-01-02	3.85	3.98	3.84	3.95	3.95	20548400
2014-01-03	3.98	4.00	3.88	4.00	4.00	22887200
2014-01-06	4.01	4.18	3.99	4.13	4.13	42398300
2014-01-07	4.19	4.25	4.11	4.18	4.18	42932100
2014-01-08	4.23	4.26	4.14	4.18	4.18	30678700

5.3 Data Preprocessing

Data pre-processing is the practice of processing raw data using a machine learning model. It is similar to the most important stages in data cleaning and building a machine learning model.

```
# Create more data
dataset['Increase/Decrease'] = np.where(dataset['Volume'].shift(-
1) > dataset['Volume'],1,0)
dataset['Buy_Sell_on_Open'] = np.where(dataset['Open'].shift(-
1) > dataset['Open'],1,-1)
dataset['Buy_Sell'] = np.where(dataset['Adj Close'].shift(-
1) > dataset['Adj Close'],1,-1)
dataset['Return'] = dataset['Adj Close'].pct_change()
dataset = dataset.dropna()
dataset.head()
```

Date	Open	High	Low	Close	Adj Close	Volume	Increase/Decrease	Buy_Sell_on_Open	Buy_Sell	Return
2014-01-03	3.98	4.00	3.88	4.00	4.00	22887200	1	1	1	0.012658
2014-01-06	4.01	4.18	3.99	4.13	4.13	42398300	1	1	1	0.032500
2014-01-07	4.19	4.25	4.11	4.18	4.18	42932100	0	1	-1	0.012106
2014-01-08	4.23	4.26	4.14	4.18	4.18	30678700	0	-1	-1	0.000000
2014-01-09	4.20	4.23	4.05	4.09	4.09	30667600	0	-1	1	-0.021531

5.4 View Dataset

```
#View Dataset

Dataset
```

```
#View Dataset
dataset
```

Date	Open	High	Low	Close	Adj Close	Volume	Increase/Decrease	Buy_Sell_on_Open	Buy_Sell	Return
2014-01-03	3.980000	4.000000	3.880000	4.000000	4.000000	22887200	1	1	1	0.012658
2014-01-06	4.010000	4.180000	3.990000	4.130000	4.130000	42398300	1	1	1	0.032500
2014-01-07	4.190000	4.250000	4.110000	4.180000	4.180000	42932100	0	1	-1	0.012106
2014-01-08	4.230000	4.260000	4.140000	4.180000	4.180000	30678700	0	-1	-1	0.000000
2014-01-09	4.200000	4.230000	4.050000	4.090000	4.090000	30667600	0	-1	1	-0.021531
...	...	...	...	...	...	...	...	...	...	...
2018-12-24	16.520000	17.219999	16.370001	16.650000	16.650000	62933100	1	1	1	-0.016539
2018-12-26	16.879999	17.910000	16.030001	17.900000	17.900000	108811800	1	1	-1	0.075075
2018-12-27	17.430000	17.740000	16.440001	17.490000	17.490000	111373000	0	1	1	-0.022905
2018-12-28	17.530001	18.309999	17.139999	17.820000	17.820000	109214400	0	1	1	0.018868
2018-12-31	18.150000	18.510000	17.850000	18.459999	18.459999	84732200	0	-1	-1	0.035915

1257 rows × 10 columns

5.5 Null value testing and data clearance

We check for null values in the data frame to ensure that there are none. The existence of null values in the dataset causes issues during training since they function as outliers, creating a wide variance in the training process.

Data cleaning is the method of finding the parts of data that are faulty, incomplete, unreliable, obsolete, or unavailable, and then updating, removing, or replacing them, as necessary. Data cleaning is considered as a fundamental component of model training on machine learning. As a good practice when doing modification on data frames, we done the modifications to new data frames.

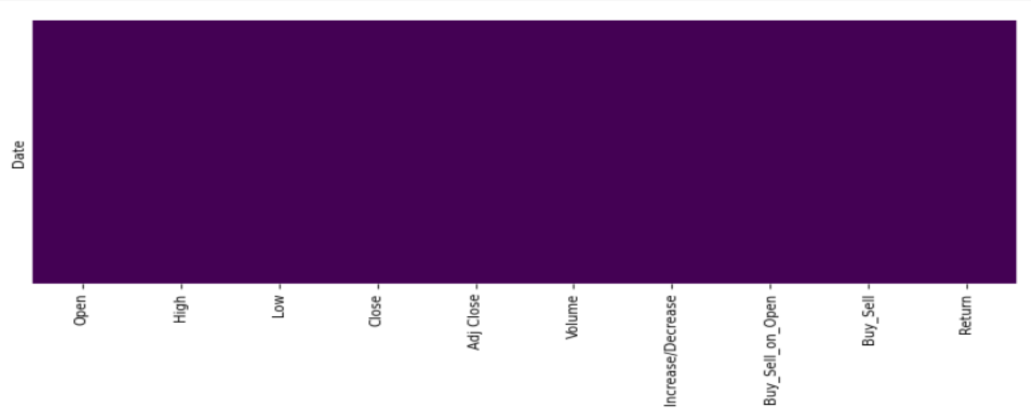
```
# See how many null values in each column
dataset.isnull().sum()
```

```
# See how many null values in each column

dataset.isnull().sum()
```

```
Open                 0
High                 0
Low                  0
Close                0
Adj Close            0
Volume               0
Increase/Decrease    0
Buy_Sell_on_Open     0
Buy_Sell             0
Return               0
dtype: int64
```

```
plt.figure(figsize=(12,4))
sns.heatmap(dataset.isnull(), yticklabels=False,cbar=False,cmap='viridis')
plt.savefig("Figure 1: Heatmap for Null Values")
```

```
plt.figure(figsize=(12,4))
sns.heatmap(dataset.isnull(), yticklabels=False,cbar=False,cmap='viridis')
plt.savefig("Figure 1: Heatmap for Null Values")
```



5.6 Analyze the Data

Before doing anything with the data it is better to have an analysis on them. Get the size of the data. This shows 1257 rows and 10 columns. We can say this dataset is large enough to use to train a machine learning model.

```
# see number of rows, number of columns
dataset.shape
```


```
# see number of rows, number of columns
dataset.shape
```


```
(1257, 10)
```

```
#TOTAL NUMBER OF RECORDS
dataset.size
print("Total number of records = ",dataset.size)
```


```
#TOTAL NUMBER OF RECORDS
dataset.size
print("Total number of records = ",dataset.size)
```


```
Total number of records =  12570
```

```
# see columns names

dataset.columns
```


```
# see columns names

dataset.columns
```


```
Index(['Open', 'High', 'Low', 'Close', 'Adj Close', 'Volume',
       'Increase/Decrease', 'Buy_Sell_on_Open', 'Buy_Sell', 'Return'],
      dtype='object')
```

```
# See data types of the Columns

dataset.dtypes
```

```
# See data types of the Columns

dataset.dtypes
```

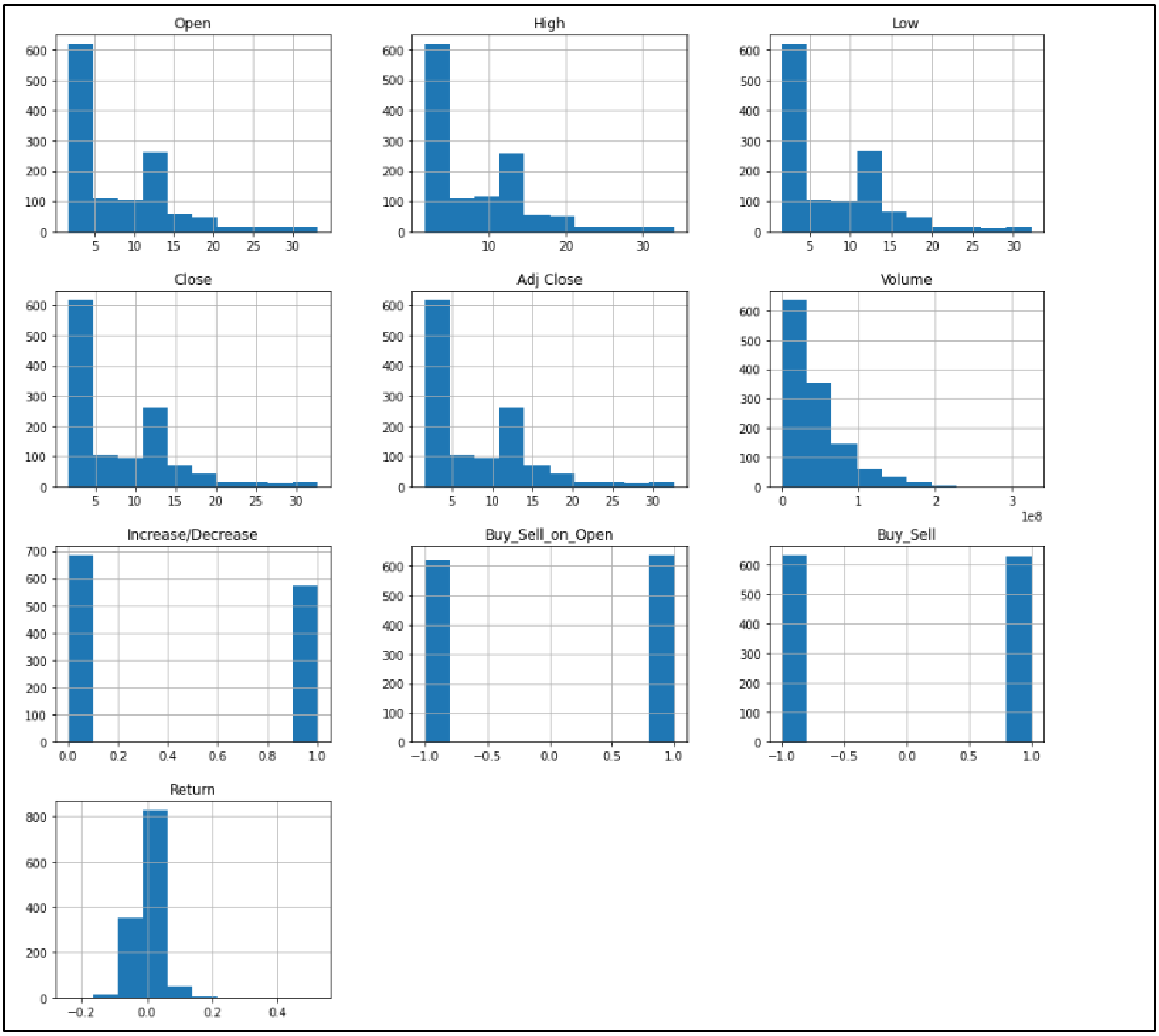
```
Open                 float64
High                 float64
Low                  float64
Close                float64
Adj Close            float64
Volume                 int64
Increase/Decrease      int64
Buy_Sell_on_Open       int64
Buy_Sell               int64
Return               float64
dtype: object
```

```
#View Data Info

dataset.info()
```

```
#View Data Info
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1257 entries, 2014-01-03 to 2018-12-31
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Open               1257 non-null   float64
 1   High               1257 non-null   float64
 2   Low                1257 non-null   float64
 3   Close              1257 non-null   float64
 4   Adj Close          1257 non-null   float64
 5   Volume             1257 non-null   int64  
 6   Increase/Decrease  1257 non-null   int64  
 7   Buy_Sell_on_Open   1257 non-null   int64  
 8   Buy_Sell           1257 non-null   int64  
 9   Return             1257 non-null   float64
dtypes: float64(6), int64(4)
memory usage: 108.0 KB
```

```python
# Histogram per each numerical column

dataset.hist(figsize=(15, 15))
```



```
# The statistics per each column

dataset.describe()
```

```
# The statistics per each column

dataset.describe()
```

	Open	High	Low	Close	Adj Close	Volume	Increase/Decrease	Buy_Sell_on_Open	Buy_Sell	Return
count	1257.000000	1257.000000	1257.000000	1257.000000	1257.000000	1.257000e+03	1257.000000	1257.000000	1257.000000	1257.000000
mean	8.169578	8.358266	7.972641	8.167780	8.167780	4.353588e+07	0.455052	0.011933	-0.003978	0.001971
std	6.482962	6.659404	6.280754	6.476459	6.476459	4.089003e+07	0.498174	1.000327	1.000390	0.039211
min	1.620000	1.690000	1.610000	1.620000	1.620000	0.000000e+00	0.000000	-1.000000	-1.000000	-0.242291
25%	2.760000	2.800000	2.710000	2.760000	2.760000	1.374570e+07	0.000000	-1.000000	-1.000000	-0.016523
50%	5.100000	5.190000	5.000000	5.100000	5.100000	3.179370e+07	0.000000	1.000000	-1.000000	0.000000
75%	12.400000	12.660000	12.120000	12.300000	12.300000	5.820150e+07	1.000000	1.000000	1.000000	0.018692
max	33.180000	34.139999	32.189999	32.720001	32.720001	3.250584e+08	1.000000	1.000000	1.000000	0.522901

5.7 Define X and Y

Linear Regression

```
X = dataset[['Open', 'High', 'Low','Volume', 'Open_Close', 'High_Low', 'Re
turns']]
y = dataset['Adj Close']
```

Decision Tree Regression

```
X = dataset.drop(['Adj Close', 'Close'], axis=1)
y = dataset['Adj Close']
```

Logistic Regression

```
# Define X
X = np.asarray(dataset[['Open', 'High', 'Low', 'Adj Close', 'Volume']])
X[0:5]

# Define y
y = np.asarray(dataset['Buy_Sell'])
y[0:5]
```

Bayesian Ridge Regression

```
X = dataset['Open'].values.reshape(1170,-1)
y = dataset['Adj Close'].values.reshape(1170,-1)
```

5.8 Split Train data and Test data

Linear Regression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                   test_size=0.4, random_state=101)
```

Decision Tree Regression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, r
andom_state=0)
```

Logistic Regression

```
# Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25
, random_state = 0)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

Bayesian Ridge Regression

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
 random_state = 0)
```

5.9 Dataset Training and Model Training

Linear Regression

```
from sklearn.linear_model import LinearRegression
lm = LinearRegression()
lm.fit(X_train,y_train)
```

Decision Tree Regression

```
from sklearn.tree import DecisionTreeRegressor
regressor = DecisionTreeRegressor()
regressor.fit(X_train, y_train)
```

Logistic Regression

```
# Fitting Logistic Regression to the Training set
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
LR = LogisticRegression(C=0.01, solver='liblinear').fit(X_train,y_train)
LR
```

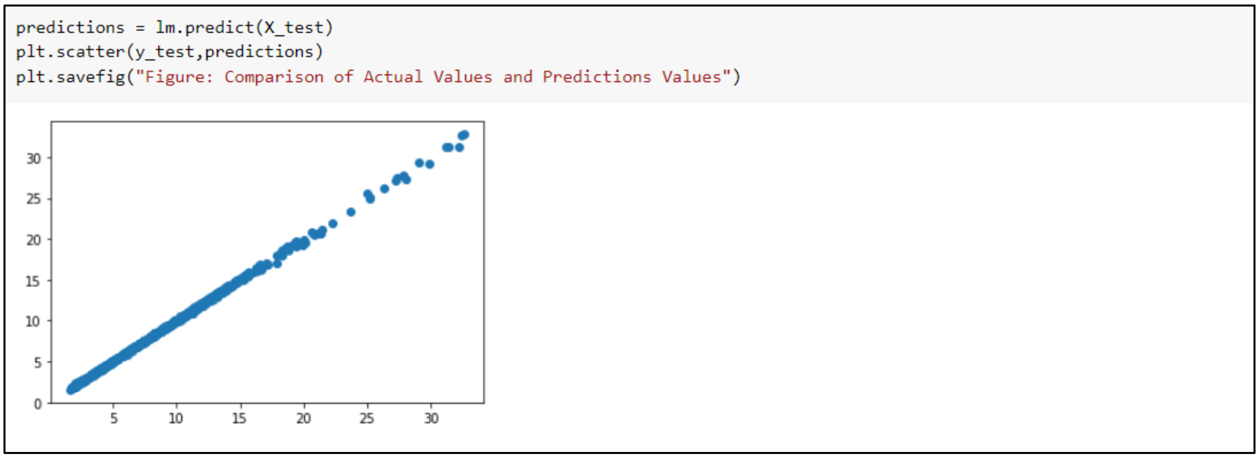
Bayesian Ridge Regression

```
model = BayesianRidge(compute_score=True)
model.fit(X_train, y_train)
```

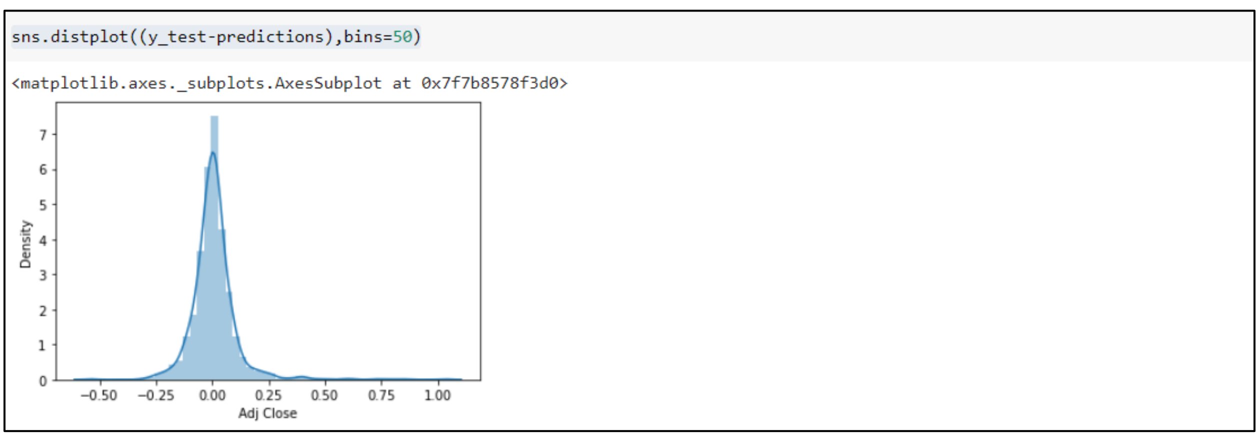
5.10 Comparison of Actual Values and Predictions Values

Linear Regression

```
predictions = lm.predict(X_test)
plt.scatter(y_test,predictions)
plt.savefig("Figure: Comparison of Actual Values and Predictions Values")
```




```
sns.distplot((y_test-predictions),bins=50)
```



Decision Tree Regression

```
y_pred = regressor.predict(X_test)

df = pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
print(df.head())
print(df.tail())
```

```
y_pred = regressor.predict(X_test)

df = pd.DataFrame({'Actual':y_test, 'Predicted':y_pred})
print(df.head())
print(df.tail())

            Actual  Predicted
Date
2017-08-04   13.12      13.02
2017-06-22   14.38      14.29
2016-05-17    3.79       3.86
2015-04-24    2.30       2.32
2015-09-11    2.01       2.09
               Actual  Predicted
Date
2017-04-20  13.110000      13.00
2014-03-11   3.850000       3.82
2018-07-09  16.610001      16.66
2018-02-21  11.720000      11.88
2018-08-20  19.980000      19.58
```

Logistic Regression

```
# Predicting the Test set results
yhat = LR.predict(X_test)
yhat
```

```
# Predicting the Test set results
yhat = LR.predict(X_test)
yhat

array([ 1,  1, -1, -1, -1, -1,  1,  1, -1,  1, -1,  1, -1,  1, -1,  1,  1,
       -1,  1, -1, -1, -1, -1, -1,  1, -1, -1,  1,  1,  1,  1,  1,  1, -1,
        1,  1, -1, -1, -1,  1, -1, -1,  1,  1,  1, -1,  1, -1, -1,  1,  1,
        1, -1,  1,  1,  1,  1,  1,  1, -1, -1, -1, -1,  1, -1, -1,  1, -1,
       -1, -1,  1, -1,  1,  1, -1, -1,  1, -1, -1,  1, -1,  1, -1, -1,  1,
        1,  1, -1, -1, -1, -1, -1,  1,  1,  1, -1,  1, -1,  1,  1,  1,  1,
        1,  1,  1,  1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1, -1,  1,
        1,  1,  1,  1, -1, -1,  1,  1, -1,  1, -1, -1,  1, -1,  1,  1,  1,
       -1, -1,  1, -1, -1, -1, -1, -1, -1, -1,  1,  1, -1,  1, -1,  1, -1,
        1, -1,  1, -1,  1, -1,  1,  1,  1, -1,  1,  1, -1,  1,  1,  1, -1,
        1,  1, -1, -1, -1,  1,  1, -1, -1, -1, -1, -1,  1, -1, -1,  1,  1,
       -1, -1, -1, -1,  1, -1, -1,  1,  1, -1,  1,  1,  1,  1, -1, -1,  1,
       -1, -1, -1, -1, -1, -1, -1,  1, -1, -1,  1,  1,  1,  1, -1,  1,  1,
        1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1, -1,  1, -1,  1,  1,  1,
        1, -1, -1,  1,  1,  1, -1,  1, -1,  1,  1, -1,  1,  1,  1, -1,  1,
       -1,  1,  1, -1,  1, -1, -1, -1,  1,  1,  1,  1,  1,  1, -1,  1,  1,
        1, -1, -1,  1,  1, -1, -1,  1, -1,  1,  1,  1,  1,  1,  1, -1, -1,
       -1,  1,  1, -1])
```

Bayesian Ridge Regression

```
y_pred = model.predict(X_test)
```

```
[24] y_pred = model.predict(X_test)

y_pred

array([13.19758088, 14.097286  ,  3.80066688,  2.36113955,  1.88129722,
        2.18119881, 12.03796184,  2.73101821,  2.39112991, 13.23756771,
       16.49649774,  6.44979679,  2.92095589,  7.6094163 ,  2.43111674,
       15.77673383, 13.41750892,  3.74068663, 11.99797501,  4.34048981,
        3.60073249,  2.79099845,  4.09057188, 19.96535791, 12.75772574,
        2.32115272,  2.99093284, 12.65775819, 11.60810269,  4.75035506,
       13.69741674, 12.84769635, 13.42750587,  1.87130052, 12.76772269,
       13.19758088,  3.96061444,  2.10122491,  2.14121198, 10.44848366,
        2.00125783,  2.41112332,  6.22986922,  7.2395374 ,  5.48011543,
        2.67103796, 10.84835291,  1.80132344, 16.70642788, 11.24822122,
        7.78935704, 11.81803379, 11.60810269, 13.97732455, 10.89833574,
       13.20757782,  6.30984288,  6.73970155, 10.50846438,  2.32115272,
        1.82131698,  3.56074566,  2.12121833,  1.93128076,  3.63072285,
        3.0609098 , 12.27788282,  2.5110904 ,  2.69103138,  1.82131698,
        7.48945533,  2.13121527, 12.80770952, 10.70839853,  2.34114613,
        3.71069651, 14.98699321,  1.92128405,  2.31115601, 12.10793855,
        2.44111345,  6.99961642,  2.75101162,  2.77100504,  4.40047005,
       12.60777441, 10.27853939,  2.28116589,  2.0812315 , 14.27722626,
        3.88064078,  4.29050603,  8.94897608, 10.59843498,  2.70102808,
        2.71102479,  9.96864169,  2.63105113,  4.28050956, 13.11760722,
       13.2575616 , 13.84736807, 14.2872232 , 13.21757477,  7.0995835 ,
       12.81770551,  2.20119223,  4.10056835,  4.27051261,  3.51076212,
        2.33114942,  2.28116589,  3.30083102,  2.36113955,  2.26117247,
```

5.11 Accuracy and Loss Function Values of the Model

Linear Regression

```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, pre
dictions))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, predi
ctions))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error
(y_test, predictions)))

print("Accuracy score: {:.7f}".format(lm.score(X_test, y_test)))
```

```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, predictions))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, predictions))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, predictions)))

Mean_Absolute_Error(MAE): 0.0617037525685526
Mean_Squared_Error(MSE): 0.010497023921536569
Root_Mean_Squared_Error(RMSE): 0.10245498485450363

print("Accuracy score: {:.7f}".format(lm.score(X_test, y_test)))

Accuracy score: 0.9995713
```

Decision Tree Regression

```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, dt_fit.predict(X_test)))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, dt_fit.predict(X_test)))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, dt_fit.predict(X_test))))

print("Accuracy score: {:.7f}".format(regressor.score(X_test, y_test)))
```

```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, dt_fit.predict(X_test)))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, dt_fit.predict(X_test)))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, dt_fit.predict(X_test))))

Mean_Absolute_Error(MAE): 0.10346154015288393
Mean_Squared_Error(MSE): 0.034138895615638154
Root_Mean_Squared_Error(RMSE): 0.1847671388955248

print("Accuracy score: {:.7f}".format(regressor.score(X_test, y_test)))

Accuracy score: 0.9985958
```

Logistic Regression

```
print ("LogLoss: : %.2f" % log_loss(y_test, yhat_prob2))

print("Accuracy:",metrics.accuracy_score(y_test, yhat))
```

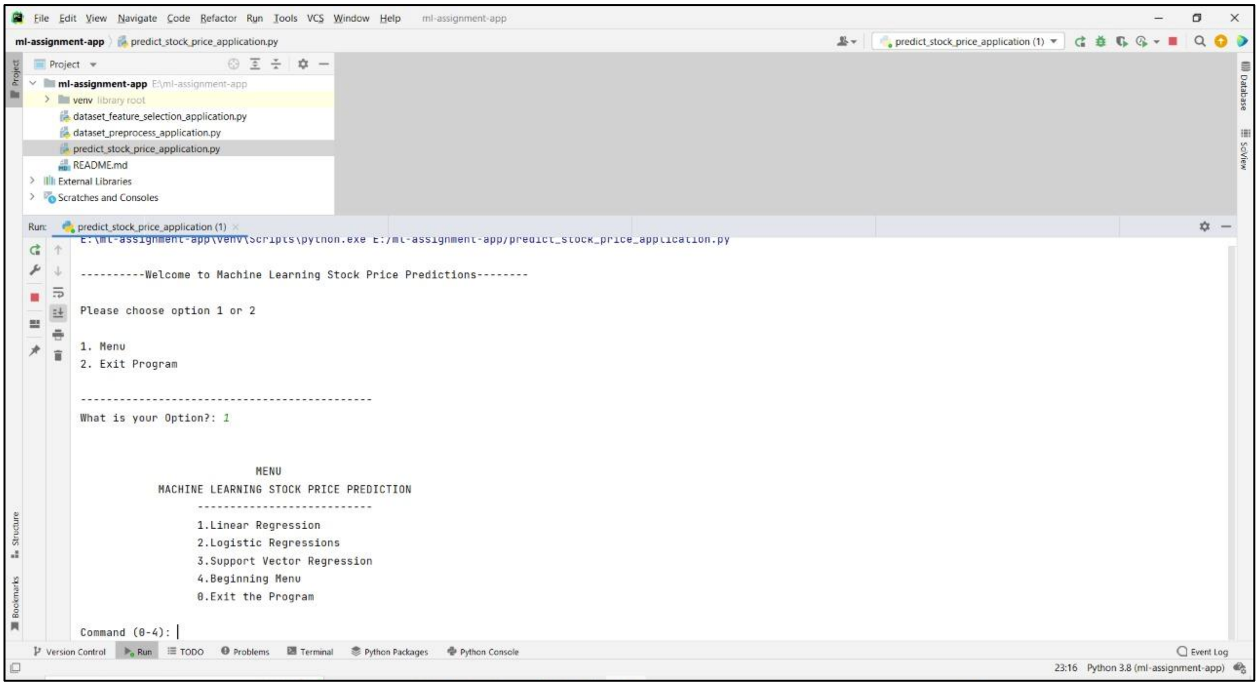
```
print ("LogLoss: : %.2f" % log_loss(y_test, yhat_prob2))

LogLoss: : 0.70

print("Accuracy:",metrics.accuracy_score(y_test, yhat))

Accuracy: 0.5426621160409556
```

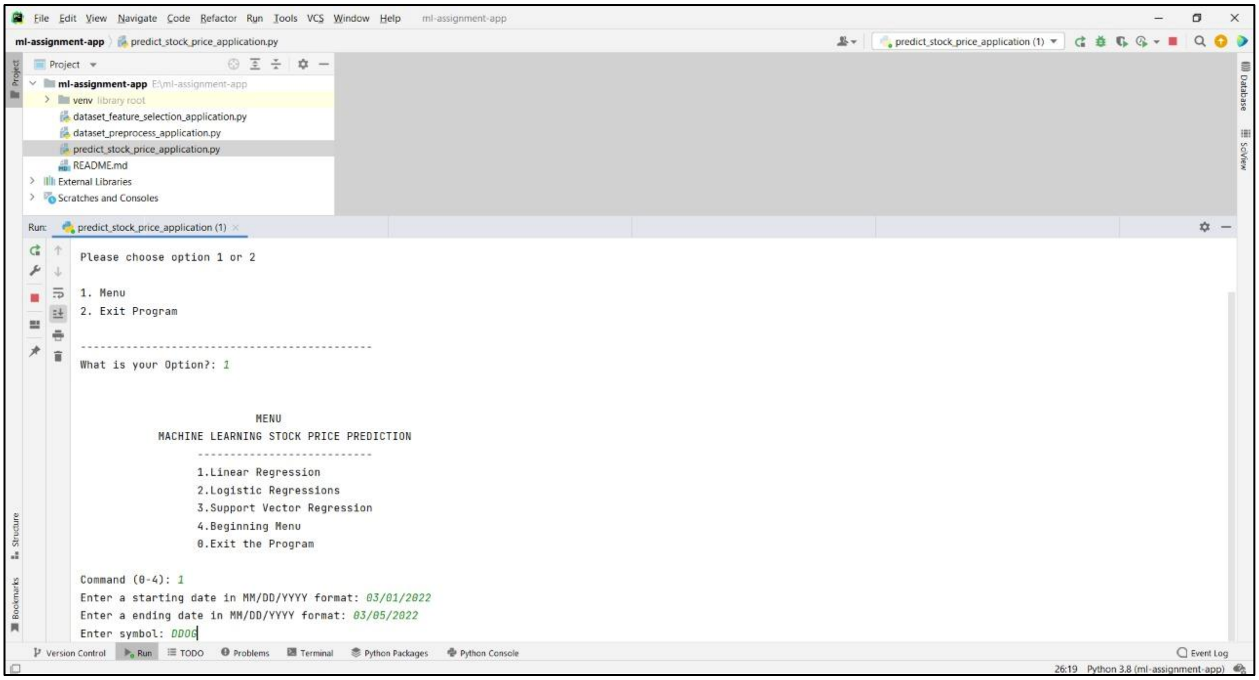
Bayesian Ridge Regression

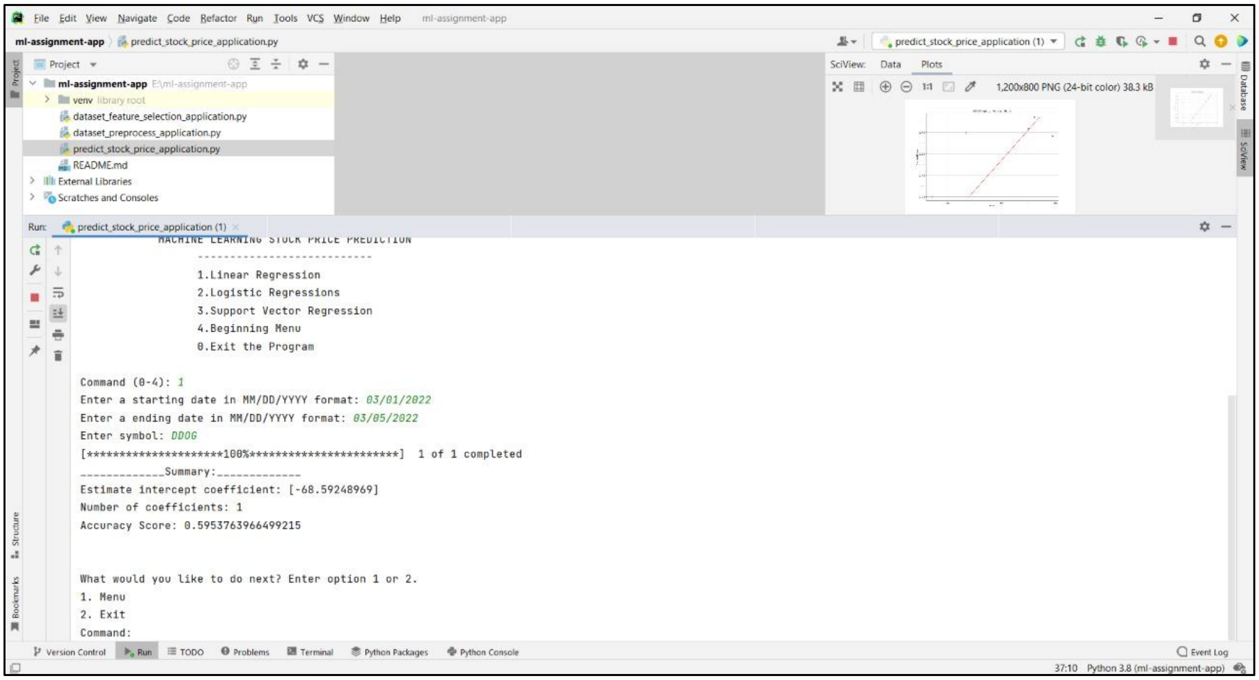
```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

print('Accuracy Score:', model.score(X_test, y_test))
```

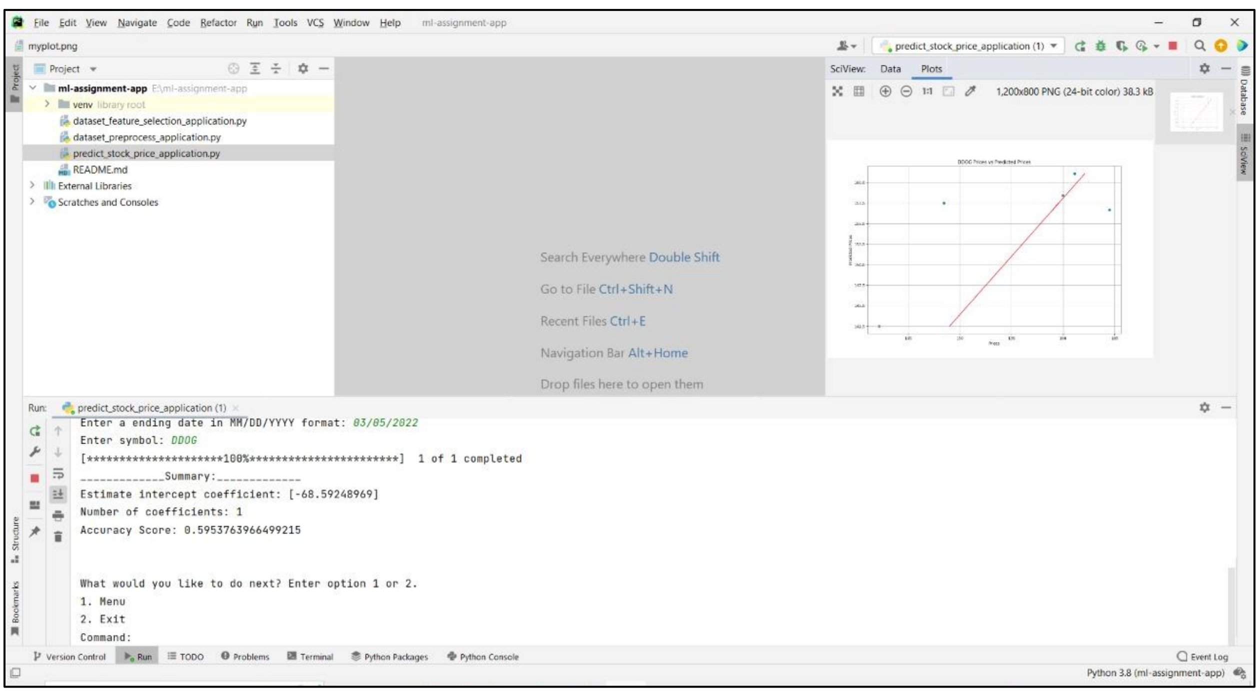
```
from sklearn import metrics
print('Mean_Absolute_Error(MAE):', metrics.mean_absolute_error(y_test, y_pred))
print('Mean_Squared_Error(MSE):', metrics.mean_squared_error(y_test, y_pred))
print('Root_Mean_Squared_Error(RMSE):', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

Mean_Absolute_Error(MAE): 0.1466702616961777
Mean_Squared_Error(MSE): 0.051846556663474076
Root_Mean_Squared_Error(RMSE): 0.22769838968133718

print('Accuracy Score:', model.score(X_test, y_test))

Accuracy Score: 0.9978674317701605
```

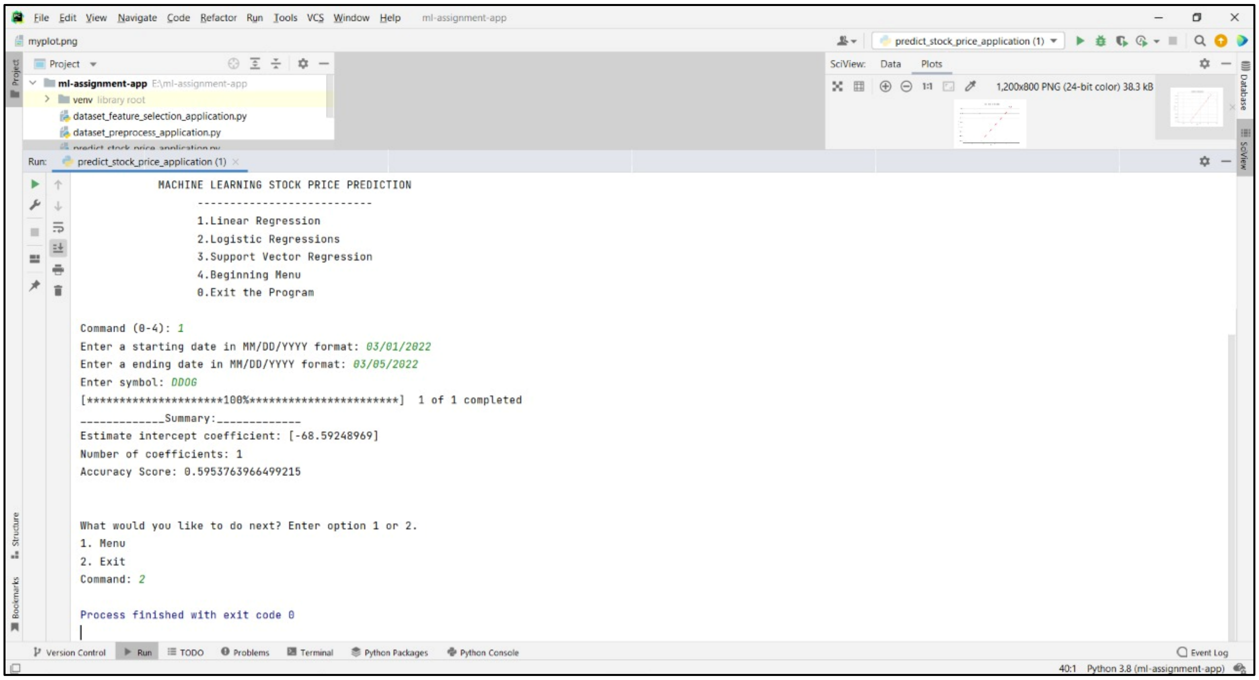
6 OUTPUT APPLICATION SCREENSHOTS



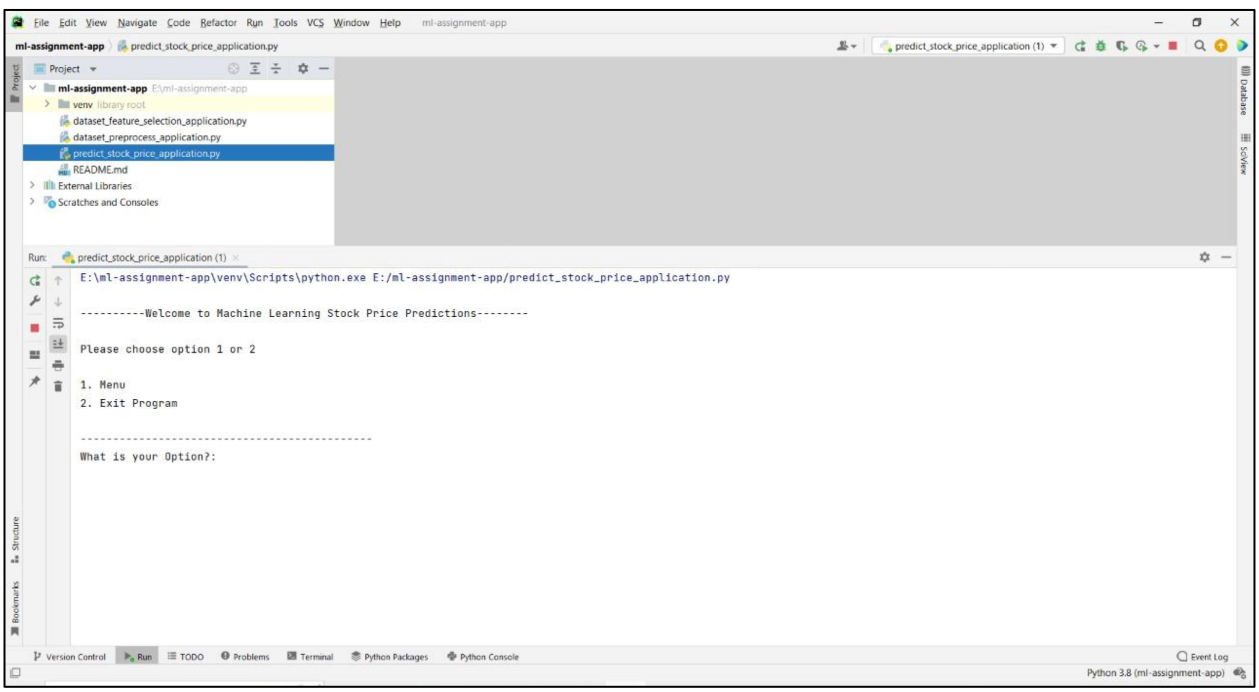



File Edit View Navigate Code Refactor Run Tools VCS Window Help ml-assignment-app
ml-assignment-app predict_stock_price_application.py
predict_stock_price_application (1)
Project
ml-assignment-app E:\ml-assignment-app
venv library root
dataset_feature_selection_application.py
dataset_preprocess_application.py
predict_stock_price_application.py
README.md
External Libraries
Scratches and Consoles
SciView: Data Plots
1,200x800 PNG (24-bit color) 38.3 kB
Run: predict_stock_price_application (1)
MACHINE LEARNING STOCK PRICE PREDICTION
1.Linear Regression
2.Logistic Regressions
3.Support Vector Regression
4.Beginning Menu
0.Exit the Program
Command (0-4): 1
Enter a starting date in MM/DD/YYYY format: 03/01/2022
Enter a ending date in MM/DD/YYYY format: 03/05/2022
Enter symbol: DDOG
[*********************100%***********************] 1 of 1 completed
____________Summary:____________
Estimate intercept coefficient: [-68.59248969]
Number of coefficients: 1
Accuracy Score: 0.5953763966499215
What would you like to do next? Enter option 1 or 2.
1. Menu
2. Exit
Command:
Version Control Run TODO Problems Terminal Python Packages Python Console
Event Log
37:10 Python 3.8 (ml-assignment-app)


File Edit View Navigate Code Refactor Run Tools VCS Window Help ml-assignment-app
myplot.png
predict_stock_price_application (1)
Project
ml-assignment-app E:\ml-assignment-app
venv library root
dataset_feature_selection_application.py
dataset_preprocess_application.py
predict_stock_price_application.py
README.md
External Libraries
Scratches and Consoles
Search Everywhere Double Shift
Go to File Ctrl+Shift+N
Recent Files Ctrl+E
Navigation Bar Alt+Home
Drop files here to open them
SciView: Data Plots
1,200x800 PNG (24-bit color) 38.3 kB
Run: predict_stock_price_application (1)
Enter a ending date in MM/DD/YYYY format: 03/05/2022
Enter symbol: DDOG
[*********************100%***********************] 1 of 1 completed
____________Summary:____________
Estimate intercept coefficient: [-68.59248969]
Number of coefficients: 1
Accuracy Score: 0.5953763966499215
What would you like to do next? Enter option 1 or 2.
1. Menu
2. Exit
Command:
Version Control Run TODO Problems Terminal Python Packages Python Console
Event Log
Python 3.8 (ml-assignment-app)


MACHINE LEARNING STOCK PRICE PREDICTION

1.Linear Regression
2.Logistic Regressions
3.Support Vector Regression
4.Beginning Menu
0.Exit the Program
Command (0-4): 1
Enter a starting date in MM/DD/YYYY format: 03/01/2022
Enter a ending date in MM/DD/YYYY format: 03/05/2022
Enter symbol: DDOG
[*********************100%***********************] 1 of 1 completed
____________Summary:____________
Estimate intercept coefficient: [-68.59248969]
Number of coefficients: 1
Accuracy Score: 0.5953763966499215
What would you like to do next? Enter option 1 or 2.
1. Menu
2. Exit
Command: 2
Process finished with exit code 0


E:\ml-assignment-app\venv\Scripts\python.exe E:/ml-assignment-app/predict_stock_price_application.py
----------Welcome to Machine Learning Stock Price Predictions--------
Please choose option 1 or 2
1. Menu
2. Exit Program

What is your Option?:

7 RESULTS AND DISCUSSION

7.1 Accuracy Comparison of the Models

The performance of a model depends on its loss function values and accuracy values.

Three common loss functions:

- **Mean Absolute Error (MAE)** is the mean of the absolute value of the errors. MAE is the easiest to understand because it's the average error
- **Mean Squared Error (MSE)** is the mean of the squared errors. MSE is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world
- **Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors. RMSE is even more popular than MSE, because RMSE is interpretable in the "y" units

All of these are loss functions because we want to minimize them to increase the performance of model.

Machine learning model accuracy is the measurement used to determine which model is best at identifying relationships and patterns between variables in a dataset based on the input, or training, data.

The table below compares the values and accuracy of the loss functions applicable to the models we used.

Model	MAE	MSE	RMSE	Log Loss	Accuracy
Linear Regression	0.061703	0.010497	0.102454	-	0.999571
Decision Tree Regression	0.103461	0.034138	0.184767	-	0.998595
Logistic Regression	-	-	-	0.70	0.542662
Bayesian Ridge Regression	0.146670	0.051846	0.227698	-	0.997867

Based on the results of both loss functions value and accuracy values approaches, we can say that Linear Regression, Decision Tree Regression and Bayesian Ridge Regression give better prediction in this scenario. Comparing loss functions value and accuracy value shows that the Linear Regression Model is best suited for stock price prediction.

8 CRITICAL ANALYSIS & DISCUSSION

8.1 Possible Limitations

Model that are being trained and maintained for one business does not fit another business. When considering the studies have done most of the studies does not use a large dataset which compromises different types of data. Some studies used only one company database for more than 14 years. One of the biggest challenges in this research area is an accurate prediction. It is due to multiple reasons, mainly micro and macro, such as global economic conditions, unforeseen events, the financial performance of a company, and politics. However, there is a huge load of data, which we can use the identify trends and patterns. Therefore, researchers, analysts in finance, data science, and data scientists use this data to explore different analytical techniques. We Can't Predict All Contingencies. For example, predicting that something will rise when prices fall can disrupt a trader's finances, especially since we do not know exactly how the market will react to ongoing news or information. When prices fall, even the good news is that prices will not rise significantly, and when prices rise, even bad news does not have a long-term negative impact on prices. Individual stocks do not necessarily follow the overall market. Unexpected changes in stuff price prediction can occur all at once. In such a case, a previously trained model may make some mistakes in making decisions.

8.2 Challenges in Stock Price Prediction

- Biggest challenge is achieving a higher accuracy amid the different reasons and problems.
- Unexpected global and local situation like COVID-19, and war will lead to less accuracy of the models even we trained models to predict profit, gain, and loss.
- Most of the models are overmounting.
- Difficulties introduced on identifying quality data because fake databases and bot generated datasets.
- When using binary features sometimes important data might be lost while converting to binary features.

8.3 Future Work (Areas of Possible Improvement)

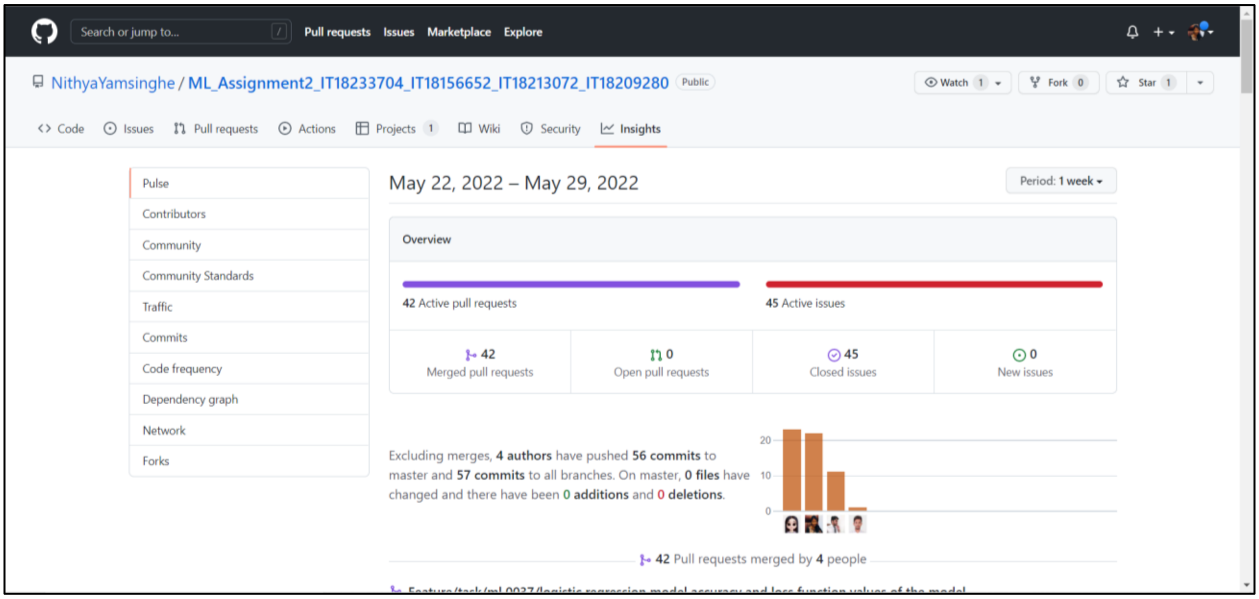
- Adding important information from the original database an enrich the research database while improving forecast accuracy.
- Although we have many models predict stock prices, there are a smaller number of models to predict the changes in stock prices.
- Use the approaches to predict the inventory costs in the future.
- Using hybrid forecast models can improve the accuracy.
- Most of the times different variables of dataset considered for training model can be unique to relevant country, income, and education of that country. Therefore, training a universal accurate model is really challenging.
- Use more feature selection and feature engineering to help enrich the database and mine important information from the original database to improve forecast accuracy.

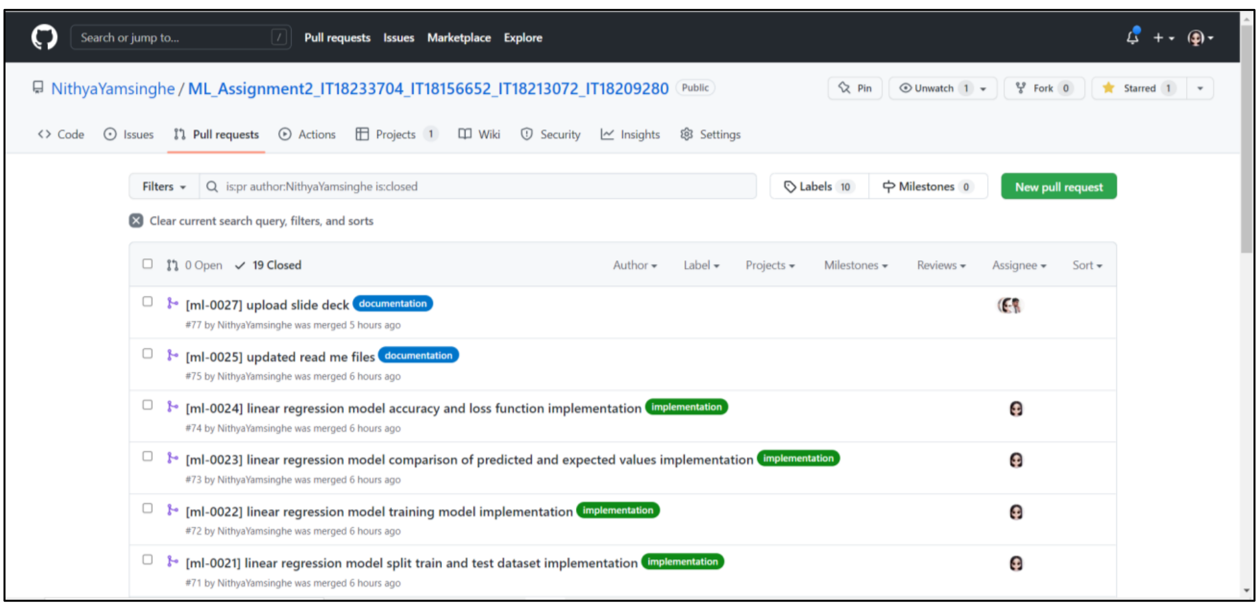
8.4 Solutions for Stock Price Prediction

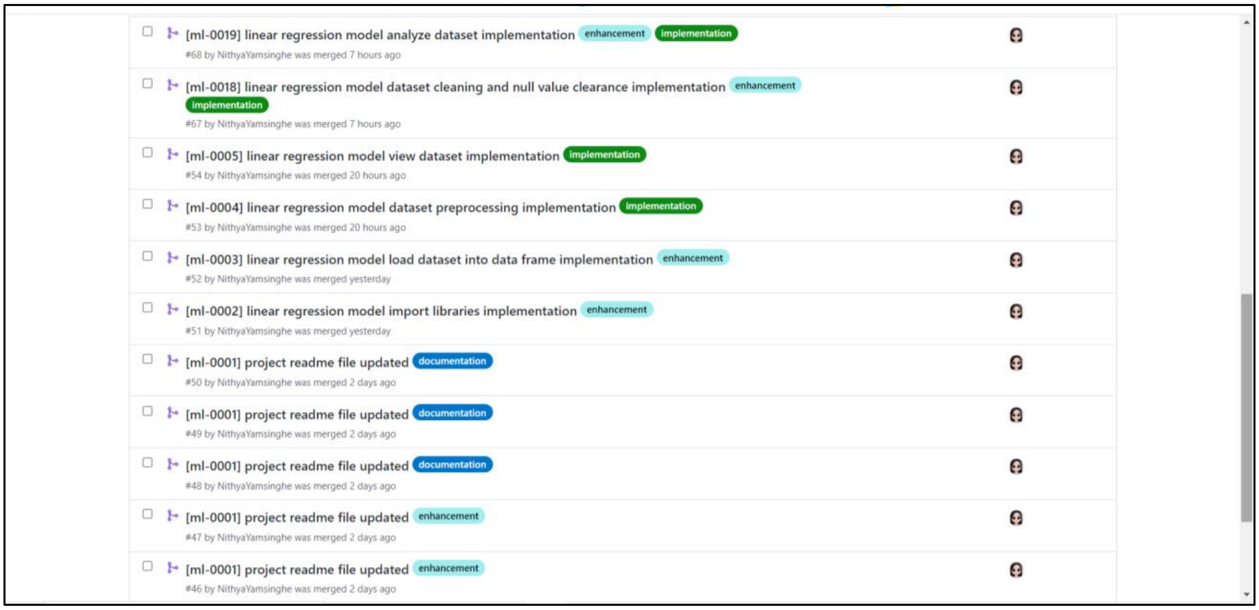
- Determining stock patterns
- Construct pattern networks
- Extract essential characteristic variables
- Determine the optimal parameter values for best suitable model
- Implement in the real market

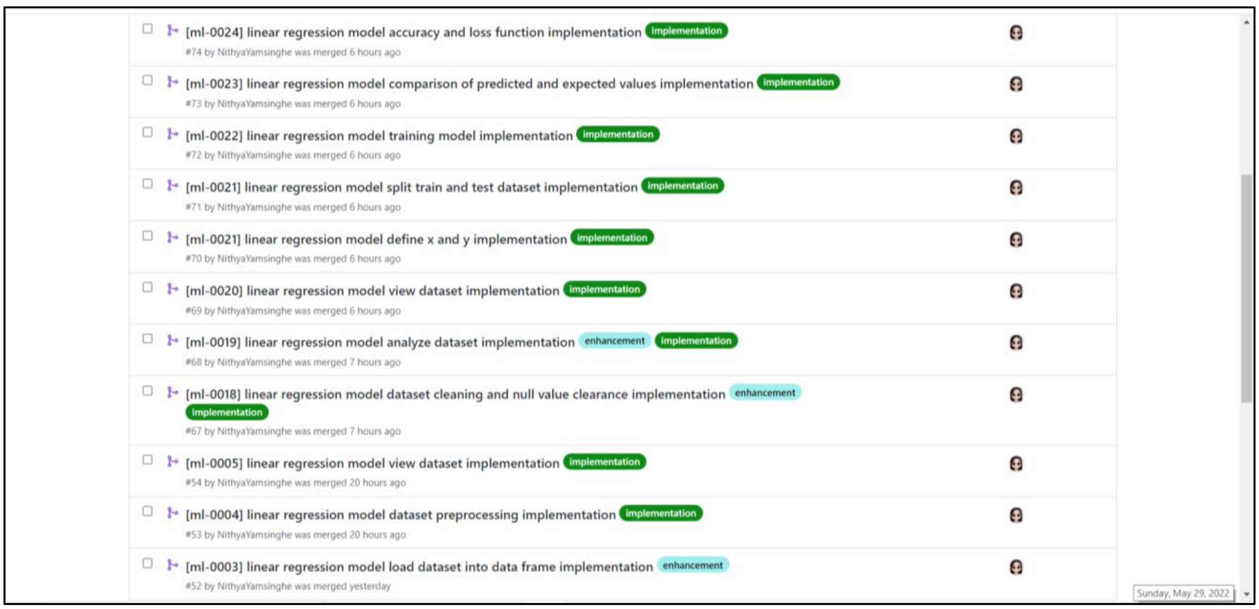
9 INDIVIDUAL CONTRIBUTION

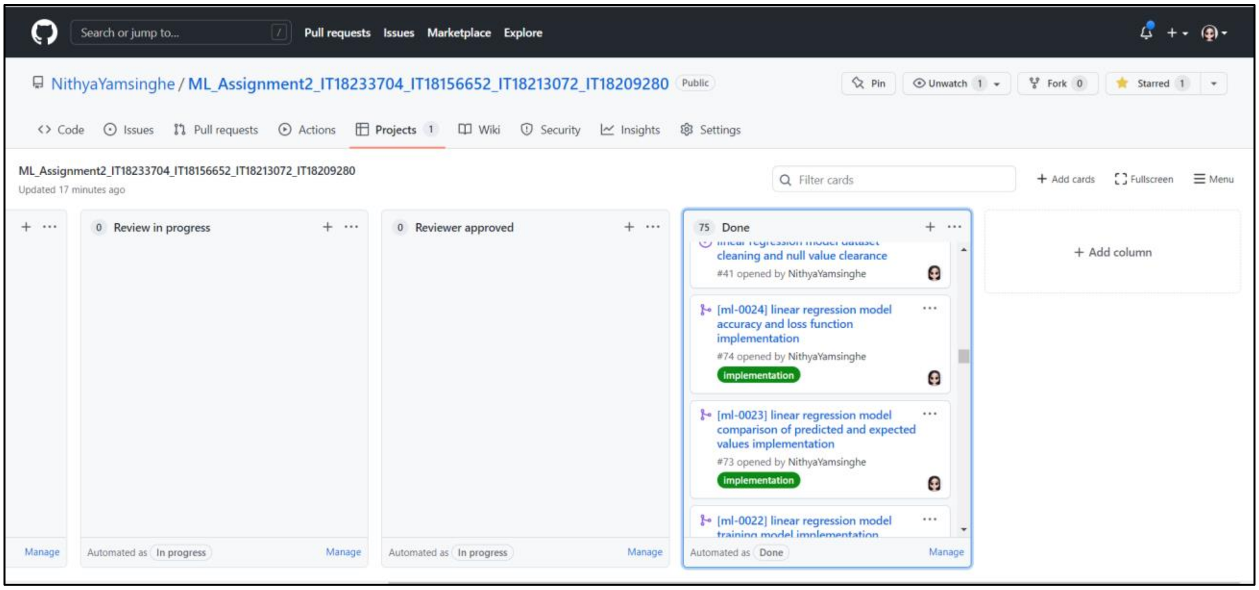
9.1 GitHub Commits Screenshots - IT18233704 | Yamasinghe N.R






[ml-0019] linear regression model analyze dataset implementation enhancement implementation
#68 by NithyaYamsinghe was merged 7 hours ago
[ml-0018] linear regression model dataset cleaning and null value clearance implementation enhancement implementation
#67 by NithyaYamsinghe was merged 7 hours ago
[ml-0005] linear regression model view dataset implementation implementation
#54 by NithyaYamsinghe was merged 20 hours ago
[ml-0004] linear regression model dataset preprocessing implementation implementation
#53 by NithyaYamsinghe was merged 20 hours ago
[ml-0003] linear regression model load dataset into data frame implementation enhancement
#52 by NithyaYamsinghe was merged yesterday
[ml-0002] linear regression model import libraries implementation enhancement
#51 by NithyaYamsinghe was merged yesterday
[ml-0001] project readme file updated documentation
#50 by NithyaYamsinghe was merged 2 days ago
[ml-0001] project readme file updated documentation
#49 by NithyaYamsinghe was merged 2 days ago
[ml-0001] project readme file updated documentation
#48 by NithyaYamsinghe was merged 2 days ago
[ml-0001] project readme file updated enhancement
#47 by NithyaYamsinghe was merged 2 days ago
[ml-0001] project readme file updated enhancement
#46 by NithyaYamsinghe was merged 2 days ago


[ml-0024] linear regression model accuracy and loss function implementation implementation
#74 by NithyaYamsinghe was merged 6 hours ago
[ml-0023] linear regression model comparison of predicted and expected values implementation implementation
#73 by NithyaYamsinghe was merged 6 hours ago
[ml-0022] linear regression model training model implementation implementation
#72 by NithyaYamsinghe was merged 6 hours ago
[ml-0021] linear regression model split train and test dataset implementation implementation
#71 by NithyaYamsinghe was merged 6 hours ago
[ml-0021] linear regression model define x and y implementation implementation
#70 by NithyaYamsinghe was merged 6 hours ago
[ml-0020] linear regression model view dataset implementation implementation
#69 by NithyaYamsinghe was merged 6 hours ago
[ml-0019] linear regression model analyze dataset implementation enhancement implementation
#68 by NithyaYamsinghe was merged 7 hours ago
[ml-0018] linear regression model dataset cleaning and null value clearance implementation enhancement implementation
#67 by NithyaYamsinghe was merged 7 hours ago
[ml-0005] linear regression model view dataset implementation implementation
#54 by NithyaYamsinghe was merged 20 hours ago
[ml-0004] linear regression model dataset preprocessing implementation implementation
#53 by NithyaYamsinghe was merged 20 hours ago
[ml-0003] linear regression model load dataset into data frame implementation enhancement
#52 by NithyaYamsinghe was merged yesterday
Sunday, May 29, 2022



9.2 Individual References - IT18233704 | Yamasinghe N.R

[01] https://vitalflux.com/popular-machine-learning-techniques-for-stock-price-movement-prediction/#:~:text=Machine%20learning%20techniques%20used%20for%20predicting%20stock%20prices%20involve%20analyzing,the%20best%20fit%20predictive%20models.

[02] https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

[03] https://www.sciencedirect.com/science/article/pii/S1877050918307828

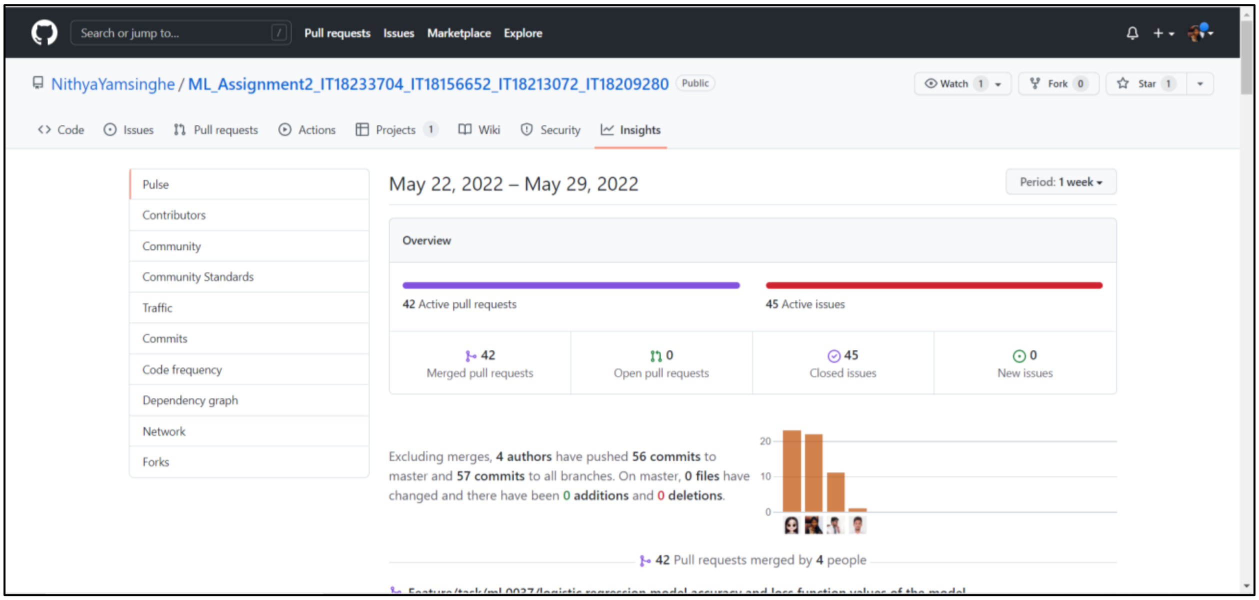
[04] https://towardsdatascience.com/5-reasons-why-stock-prediction-projects-fail-a3dddf30d242

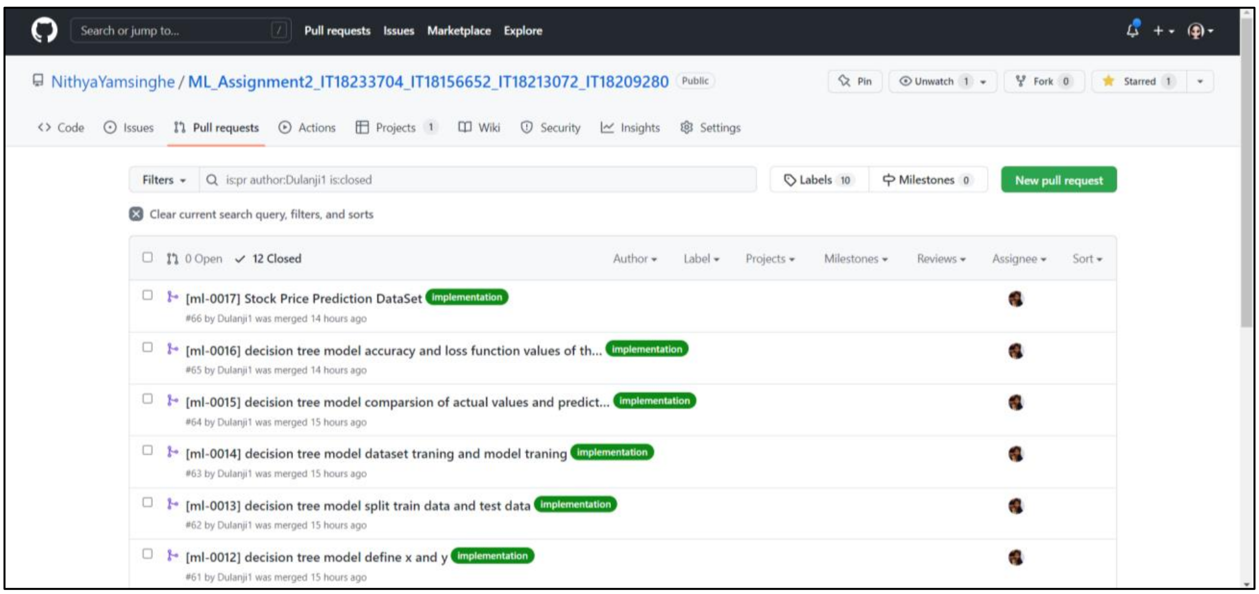
[05] https://www.android-examples.com/category/phpmyadmin/

[06] https://hbr.org/2009/01/why-we-cant-predict-financial

[07] https://github.com/CYBERDEVILZ/Stock-Market-Prediction

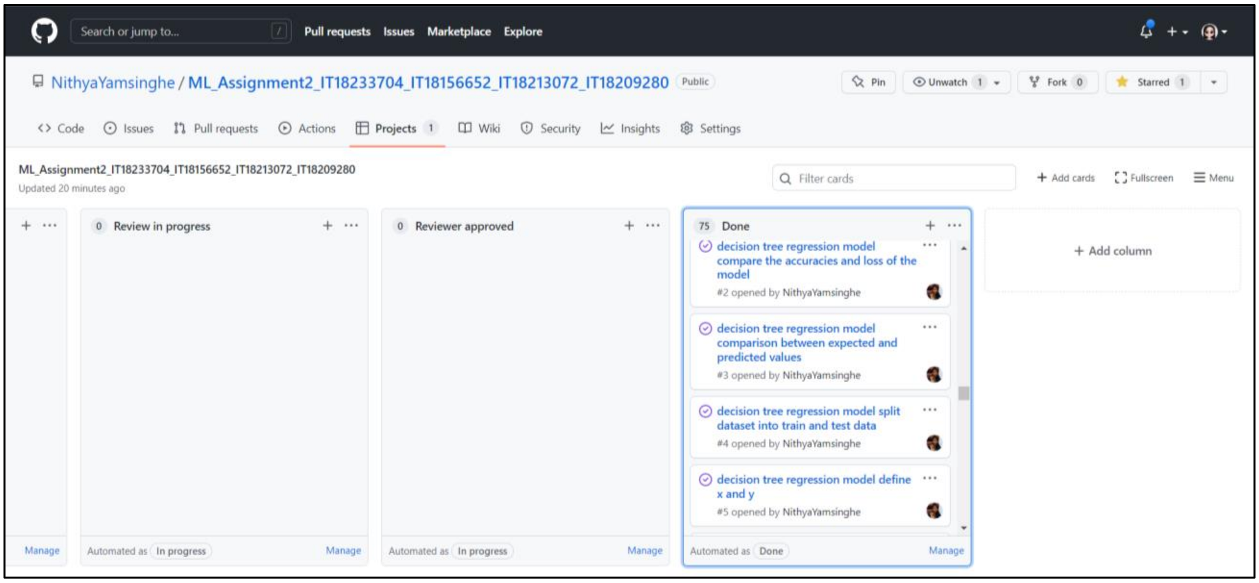
9.3 GitHub Commits Screenshots - IT18156652 | Cooray M.D.D.M






#66 by Dulanji1 was merged 14 hours ago
[ml-0016] decision tree model accuracy and loss function values of th... implementation
#65 by Dulanji1 was merged 14 hours ago
[ml-0015] decision tree model comparsion of actual values and predict... implementation
#64 by Dulanji1 was merged 15 hours ago
[ml-0014] decision tree model dataset traning and model traning implementation
#63 by Dulanji1 was merged 15 hours ago
[ml-0013] decision tree model split train data and test data implementation
#62 by Dulanji1 was merged 15 hours ago
[ml-0012] decision tree model define x and y implementation
#61 by Dulanji1 was merged 15 hours ago
[ml-0011] decision tree model analyze data set implementation
#60 by Dulanji1 was merged 15 hours ago
[ml-0010] decision tree model null value testing and data clearance implementation
#59 by Dulanji1 was merged 15 hours ago
[ml-0009] decision tree model view data set implementation
#58 by Dulanji1 was merged 15 hours ago
[ml-0008] decision tree model data preprocessing implementation
#57 by Dulanji1 was merged 15 hours ago
[ml-0007] decision tree model load data into a data frame implementation
#56 by Dulanji1 was merged 15 hours ago
[ml-0006] decision tree model import libraries implementation
#55 by Dulanji1 was merged 15 hours ago


[ml-0017] Stock Price Prediction DataSet implementation
#66 by Dulanji1 was merged 14 hours ago
[ml-0016] decision tree model accuracy and loss function values of th... implementation
#65 by Dulanji1 was merged 14 hours ago
[ml-0015] decision tree model comparsion of actual values and predict... implementation
#64 by Dulanji1 was merged 15 hours ago
[ml-0014] decision tree model dataset traning and model traning implementation
#63 by Dulanji1 was merged 15 hours ago
[ml-0013] decision tree model split train data and test data implementation
#62 by Dulanji1 was merged 15 hours ago
[ml-0012] decision tree model define x and y implementation
#61 by Dulanji1 was merged 15 hours ago
[ml-0011] decision tree model analyze data set implementation
#60 by Dulanji1 was merged 15 hours ago
[ml-0010] decision tree model null value testing and data clearance implementation
#59 by Dulanji1 was merged 15 hours ago
[ml-0009] decision tree model view data set implementation
#58 by Dulanji1 was merged 15 hours ago
[ml-0008] decision tree model data preprocessing implementation
#57 by Dulanji1 was merged 15 hours ago
[ml-0007] decision tree model load data into a data frame implementation
#56 by Dulanji1 was merged 15 hours ago



9.4 Individual References - IT18156652 | Cooray M.D.D.M

[01] https://vitalflux.com/popular-machine-learning-techniques-for-stock-price-movement-prediction/#:~:text=Machine%20learning%20techniques%20used%20for%20predicting%20stock%20prices%20involve%20analyzing,the%20best%20fit%20predictive%20models.

[02] https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

[03] https://github.com/LastAncientOne/Deep-Learning-Machine-Learning-Stock/tree/master/Stock_Algorithms

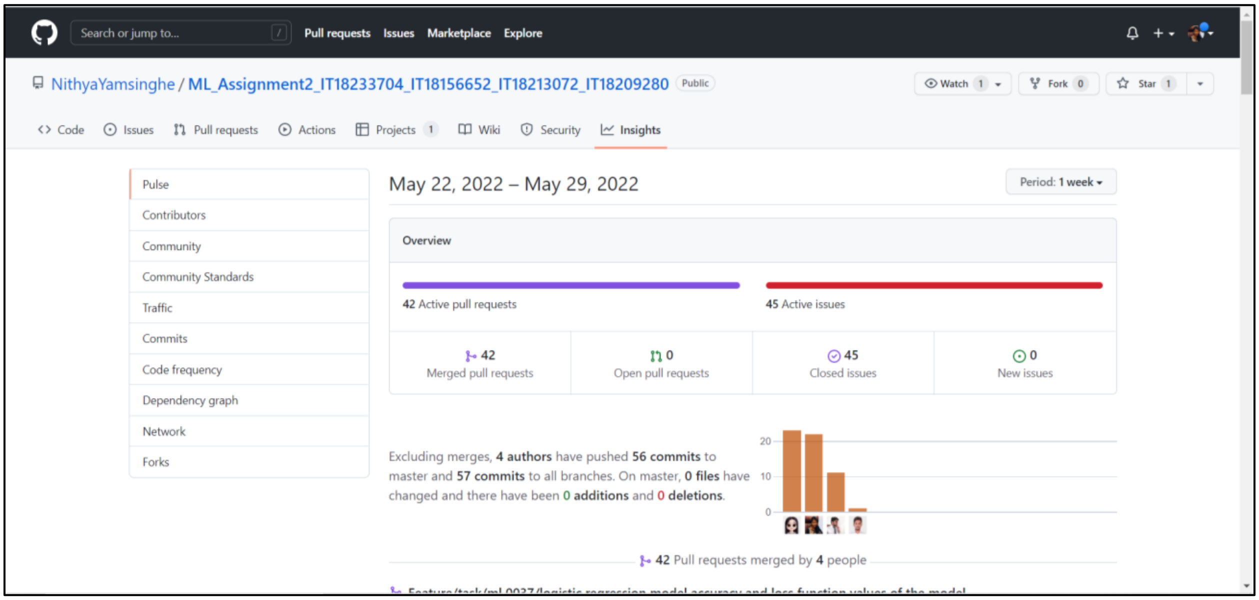
[04] https://towardsdatascience.com/5-reasons-why-stock-prediction-projects-fail-a3dddf30d242

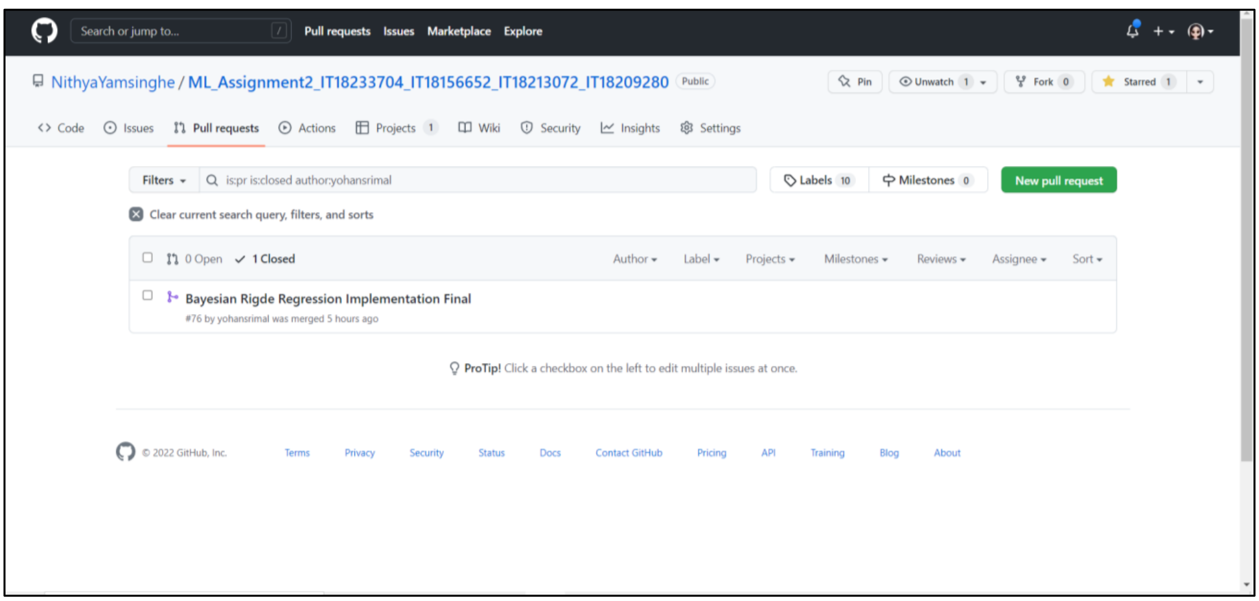
[05] https://github.com/NourozR/Stock-Price-Prediction-LSTM

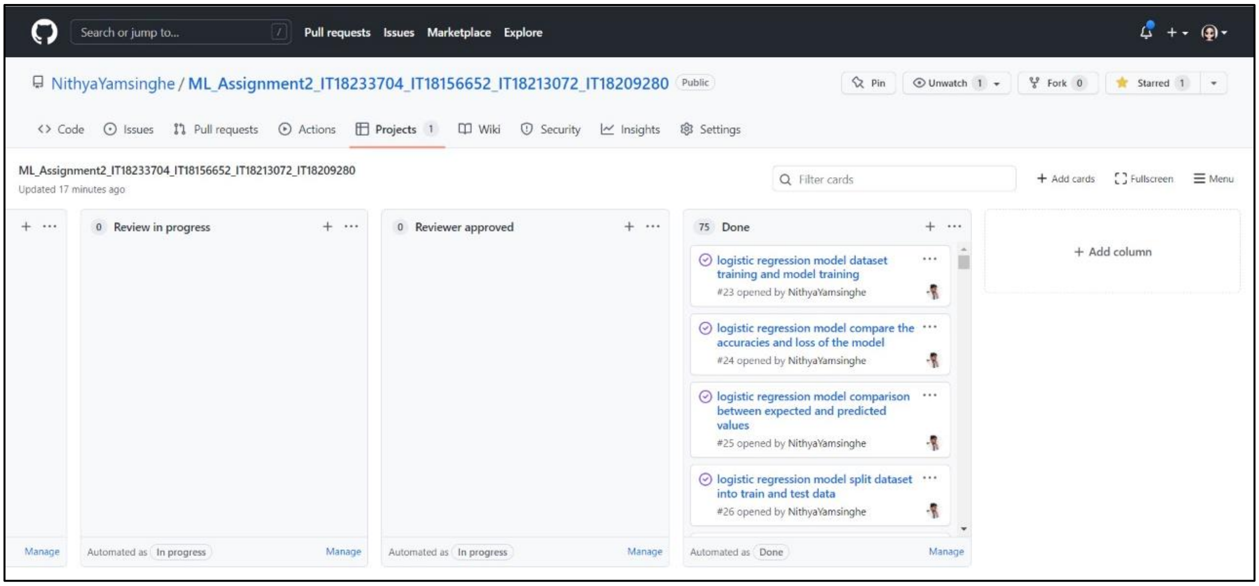
[06] https://www.sciencedirect.com/science/article/pii/S1877050918307828

[07] https://github.com/CYBERDEVILZ/Stock-Market-Prediction

9.5 GitHub Commits Screenshots - IT18213072 | Ranasinghe Y.S







9.6 Individual References - IT18213072 | Ranasinghe Y.S

[01] https://vitalflux.com/popular-machine-learning-techniques-for-stock-price-movement-prediction/#:~:text=Machine%20learning%20techniques%20used%20for%20predicting%20stock%20prices%20involve%20analyzing,the%20best%20fit%20predictive%20models.

[02] https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

[03] https://github.com/LastAncientOne/Deep-Learning-Machine-Learning-Stock/tree/master/Stock_Algorithms

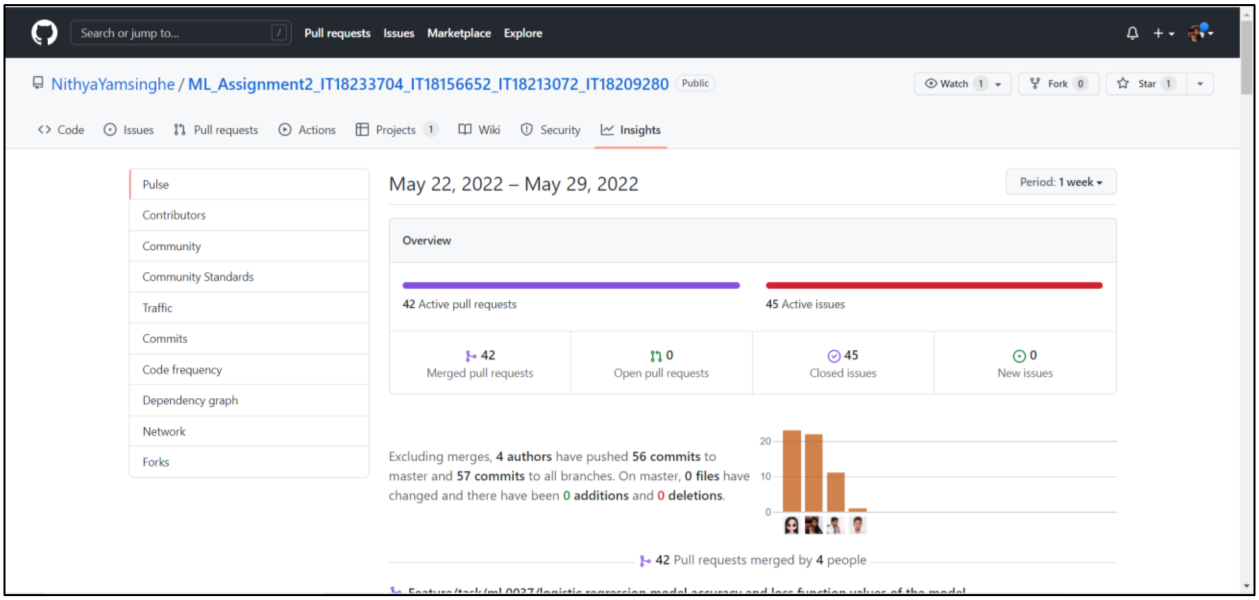
[04] https://towardsdatascience.com/5-reasons-why-stock-prediction-projects-fail-a3dddf30d242

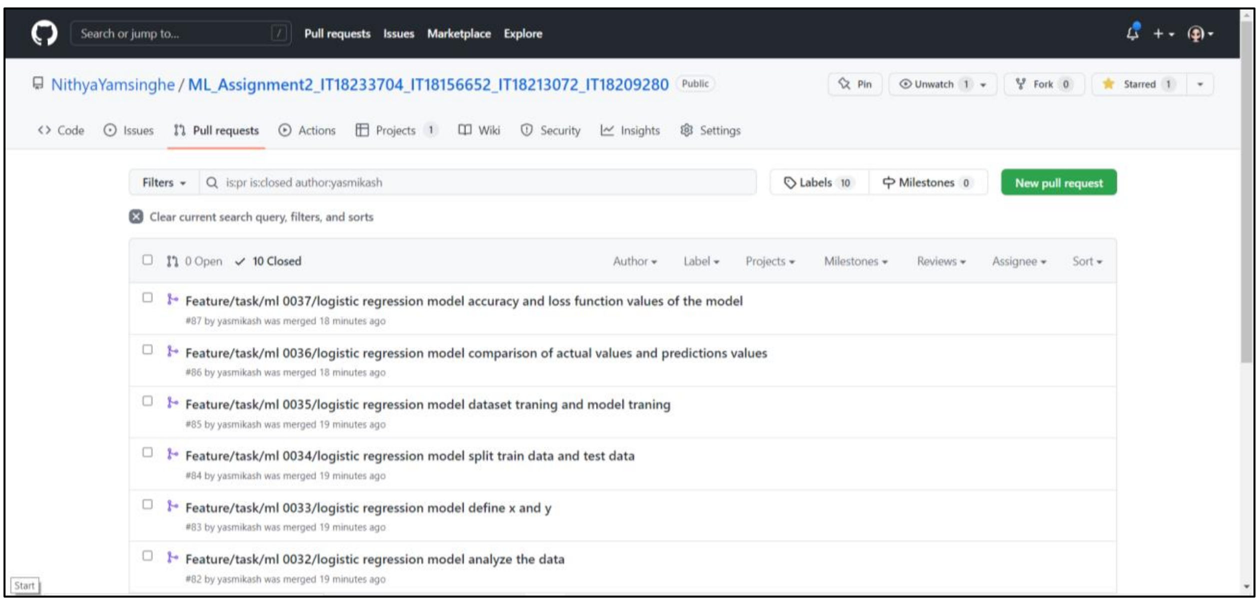
[05] https://www.android-examples.com/category/phpmyadmin/

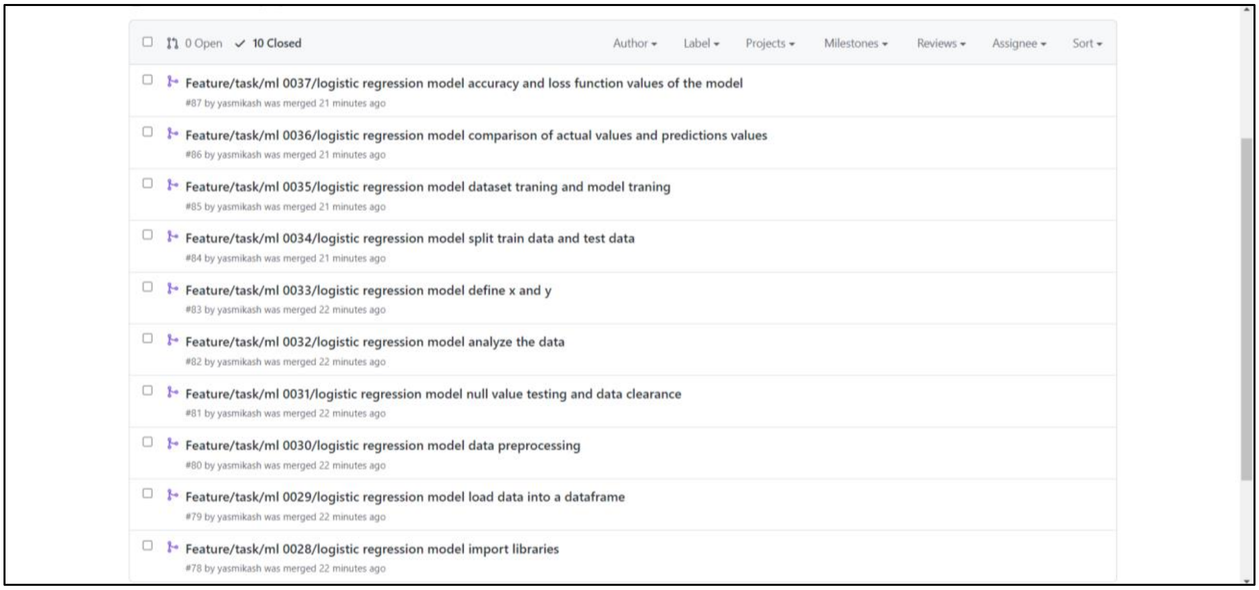
[06] https://hbr.org/2009/01/why-we-cant-predict-financial

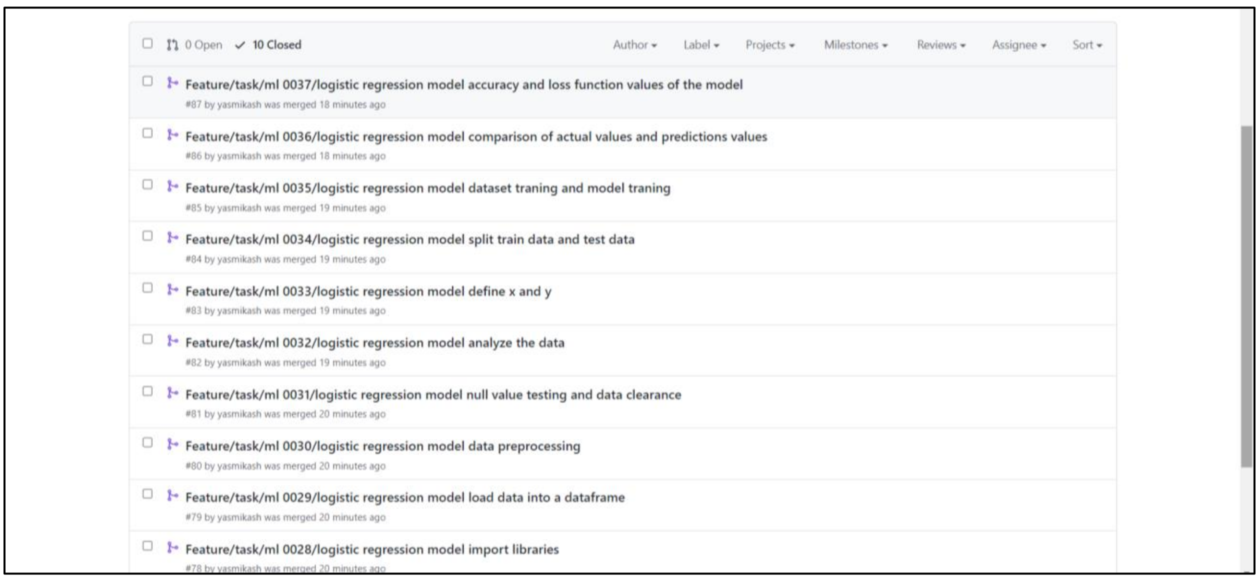
[07] https://github.com/CYBERDEVILZ/Stock-Market-Prediction

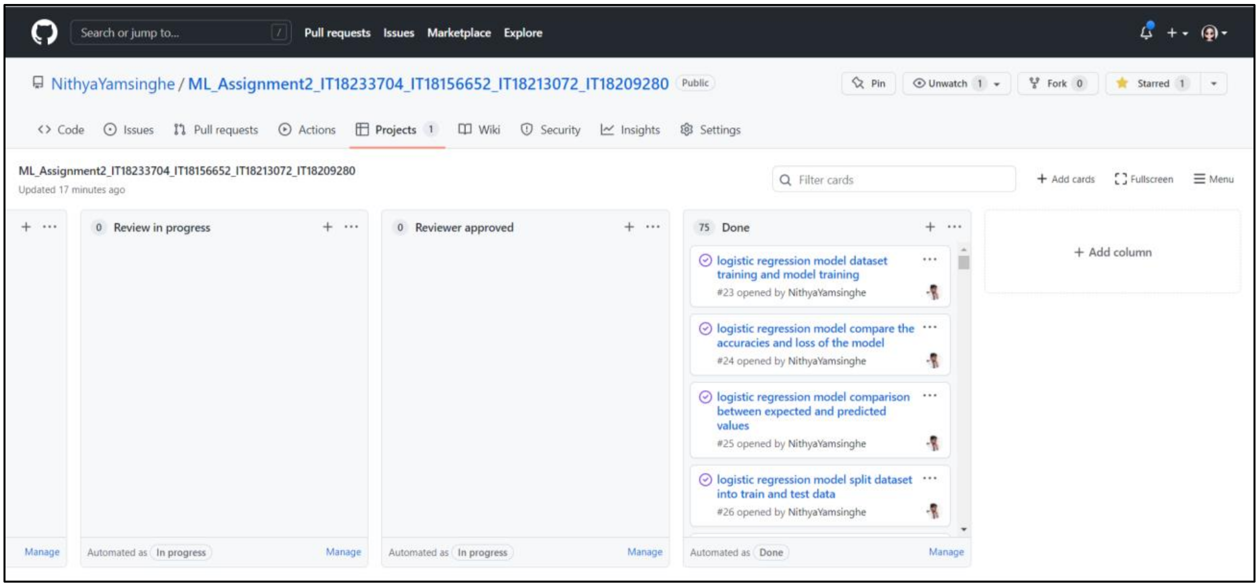
9.7 GitHub Commits Screenshots - IT18209280 | Dissanayake D.M.Y.S






0 Open
10 Closed
Author
Label
Projects
Milestones
Reviews
Assignee
Sort
Feature/task/ml 0037/logistic regression model accuracy and loss function values of the model
#87 by yasmikash was merged 21 minutes ago
Feature/task/ml 0036/logistic regression model comparison of actual values and predictions values
#86 by yasmikash was merged 21 minutes ago
Feature/task/ml 0035/logistic regression model dataset traning and model traning
#85 by yasmikash was merged 21 minutes ago
Feature/task/ml 0034/logistic regression model split train data and test data
#84 by yasmikash was merged 21 minutes ago
Feature/task/ml 0033/logistic regression model define x and y
#83 by yasmikash was merged 22 minutes ago
Feature/task/ml 0032/logistic regression model analyze the data
#82 by yasmikash was merged 22 minutes ago
Feature/task/ml 0031/logistic regression model null value testing and data clearance
#81 by yasmikash was merged 22 minutes ago
Feature/task/ml 0030/logistic regression model data preprocessing
#80 by yasmikash was merged 22 minutes ago
Feature/task/ml 0029/logistic regression model load data into a dataframe
#79 by yasmikash was merged 22 minutes ago
Feature/task/ml 0028/logistic regression model import libraries
#78 by yasmikash was merged 22 minutes ago


0 Open
10 Closed
Author
Label
Projects
Milestones
Reviews
Assignee
Sort
Feature/task/ml 0037/logistic regression model accuracy and loss function values of the model
#87 by yasmikash was merged 18 minutes ago
Feature/task/ml 0036/logistic regression model comparison of actual values and predictions values
#86 by yasmikash was merged 18 minutes ago
Feature/task/ml 0035/logistic regression model dataset traning and model traning
#85 by yasmikash was merged 19 minutes ago
Feature/task/ml 0034/logistic regression model split train data and test data
#84 by yasmikash was merged 19 minutes ago
Feature/task/ml 0033/logistic regression model define x and y
#83 by yasmikash was merged 19 minutes ago
Feature/task/ml 0032/logistic regression model analyze the data
#82 by yasmikash was merged 19 minutes ago
Feature/task/ml 0031/logistic regression model null value testing and data clearance
#81 by yasmikash was merged 20 minutes ago
Feature/task/ml 0030/logistic regression model data preprocessing
#80 by yasmikash was merged 20 minutes ago
Feature/task/ml 0029/logistic regression model load data into a dataframe
#79 by yasmikash was merged 20 minutes ago
Feature/task/ml 0028/logistic regression model import libraries
#78 by yasmikash was merged 20 minutes ago



9.8 Individual References - IT18209280 | Dissanayake D.M.Y.S

[01] https://vitalflux.com/popular-machine-learning-techniques-for-stock-price-movement-prediction/#:~:text=Machine%20learning%20techniques%20used%20for%20predicting%20stock%20prices%20involve%20analyzing,the%20best%20fit%20predictive%20models.

[02] https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

[03] https://github.com/LastAncientOne/Deep-Learning-Machine-Learning-Stock/tree/master/Stock_Algorithms

[04] https://towardsdatascience.com/5-reasons-why-stock-prediction-projects-fail-a3dddf30d242

[05] https://www.android-examples.com/category/phpmyadmin/

[06] https://hbr.org/2009/01/why-we-cant-predict-financial

[07] https://github.com/CYBERDEVILZ/Stock-Market-Prediction

10 REFERANCES

[01] https://vitalflux.com/popular-machine-learning-techniques-for-stock-price-movement-prediction/#:~:text=Machine%20learning%20techniques%20used%20for%20predicting%20stock%20prices%20involve%20analyzing,the%20best%20fit%20predictive%20models.

[02] https://www.sciencedirect.com/science/article/pii/S1877050918307828

[03] https://github.com/LastAncientOne/Deep-Learning-Machine-Learning-Stock/tree/master/Stock_Algorithms

[04] https://github.com/NourozR/Stock-Price-Prediction-LSTM

[05] https://www.projectpro.io/article/stock-price-prediction-using-machine-learning-project/571

[06] https://github.com/CYBERDEVILZ/Stock-Market-Prediction

[07] https://towardsdatascience.com/5-reasons-why-stock-prediction-projects-fail-a3dddf30d242

[08] https://www.android-examples.com/category/phpmyadmin/

[09] https://hbr.org/2009/01/why-we-cant-predict-financial

[10] https://www.emerald.com/insight/content/doi/10.1108/IJCS-05-2020-0012/full/html

11 APPENDICES

11.1 GitHub Repository Link

https://github.com/NithyaYamsinghe/ML_Assignment2_IT18233704_IT18156652_IT18213072_IT18209280

11.2 GitHub Project Link

https://github.com/NithyaYamsinghe/ML_Assignment2_IT18233704_IT18156652_IT18213072_IT18209280/projects/1

11.3 Video Demonstration One Drive Link

https://mysliit-my.sharepoint.com/:f:/g/personal/it18233704_my_sliit_lk/EnHh4V5KZRdHtu4bcQ35rE8BFxawzYm7JimFbU-U7kpD9g?e=kLjNb3

11.4 Video Demonstration Slide Deck Link

https://drive.google.com/drive/folders/1ehb3EBDx0ZLjNj6vViHEMRR1itQnbyPS?usp=sharing

11.5 Report

https://drive.google.com/drive/folders/10k7yfftiB1OSyvz1aK9jlmL5dyDEK5U_?usp=sharing

11.6 Complete Project Link

https://drive.google.com/drive/folders/1Ppob_fOveAhjWn7MG2GwCE2tDiltQ_I1?usp=sharing