MACHINE LEARNING PROJECT

GOLD PRICE PREDICTION

SUBMITTING TO :

1. Mr. Mayur Dev Sewak
   General Manager , Operations
   Eisystems Services

2. Ms. Mallika Srivastava
   Trainer , Programming & Algorithms,
   Eisystems Services

SUBMITTING BY:

SHUBHAM KUMAR

Content Table

List of Figure

Abstract of Project

In this project, we were asked to experiment with a real world dataset, and to explore how machine learning algorithms can be used to find the patterns in data. We were expected to gain experience using a common data-mining and machine learning library, Weka, and were expected to submit a report about the dataset and the algorithms used. After performing the required tasks on a dataset of my choice, herein lies my final report.

Project Summary

Project Title : Gold Price Prediction Using Machine Learning

Machine learning algorithms were used to train and model the collected data. From the data collected, eighty percentage of the data was used for training and remaining twenty percentages for testing the model. The machine learning algorithms used in this study are linear regression, random forest regression and gradient boosting regression.
The statistical process for estimating the relationship between different variables is called regression analysis. Regression analysis is used to understand how the value of the dependent variable changes when one of the independent variables changes, while other variables are fixed.
Linear regression models with more than one independent variable are called multiple linear models. A representation of multiple linear regressions is where, Y is dependent variable and X_1, X_2 … are independent variables are as seen below.

$Y = a + b_1{*}X_1 + b_2{*}X_2 + \ldots + b_p{*}X_p$

As such, linear regression was developed in the field of statistics and is studied as a model for understanding the relationship between input and output numerical variables, but has been borrowed by machine learning. It is both a statistical algorithm and a machine learning algorithm now.
A simple yet crisp definition, to understand what Random Forest Regression Algorithm is, will be, "*Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. It operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees*",

Objectives Of Project

The main objectives of the project are:

1. This project is based on the applicability of the proposed machine learning algorithms that had demonstrated their efficiency to predict gold prices with a better predictive rate.

2. To apply the best appropriate Machine Learning procedure.

3. We proposed the development of a prediction model for predicting future gold prices using Random Forest Regression Algorithm.

Details of project developed

Gold was used for supporting trade transactions around the world besides other modes of payment. Various states maintained and enhanced their gold reserves and were recognized as wealthy and progressive states. Our project will be beneficial for investors, and control banks to decide when to invest in this commodity. Here the commodity is referred to as gold. Various multinational companies and individuals have also invested in gold reserves. Big investors have also been attracted to this precious metal and invest huge amounts in it. We predict future gold rates based on 22 market variables using machine learning techniques. Results show that we can predict the daily gold rates very accurately. For almost 10 years between 2008 and 2018, gold prices barely moved in India. The spot price is the current market price at which a commodity is purchased or sold for immediate payment and delivery. It is differentiated from the futures price, which is the price at which the two parties agree to transact on a future date. Gold spot rates are decided twice a day based on supply and demand in the gold market. Fractional change in gold price may result in huge profit or loss for these investors as well as the banks of the government. Forecasting the rise and fall in the daily gold rates, can help investors to decide when to buy (or sell) the commodity.

System Requirement Used

1. Windows 10 pro
2. Python 3
3. PyCharm IDE
4. Command prompt

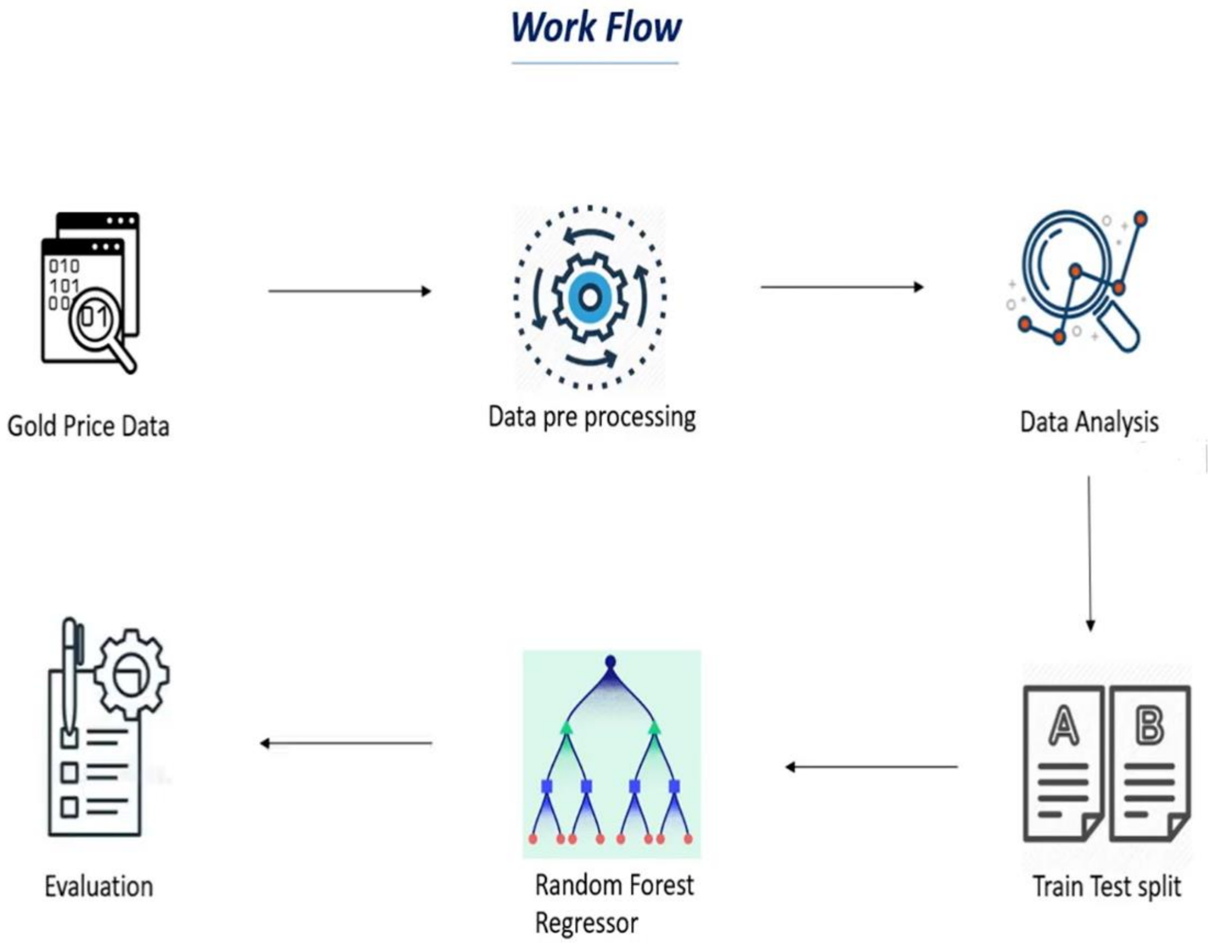
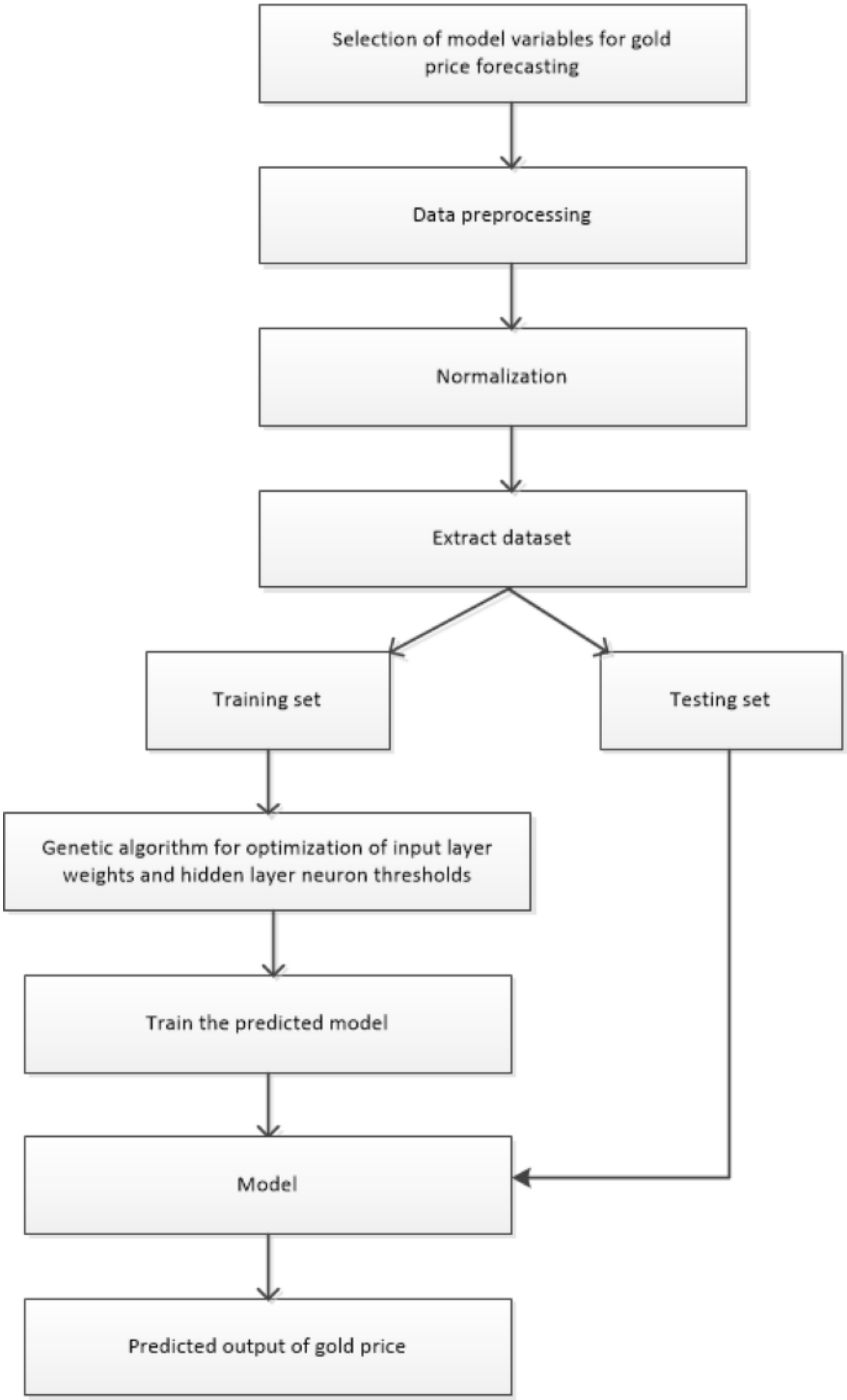


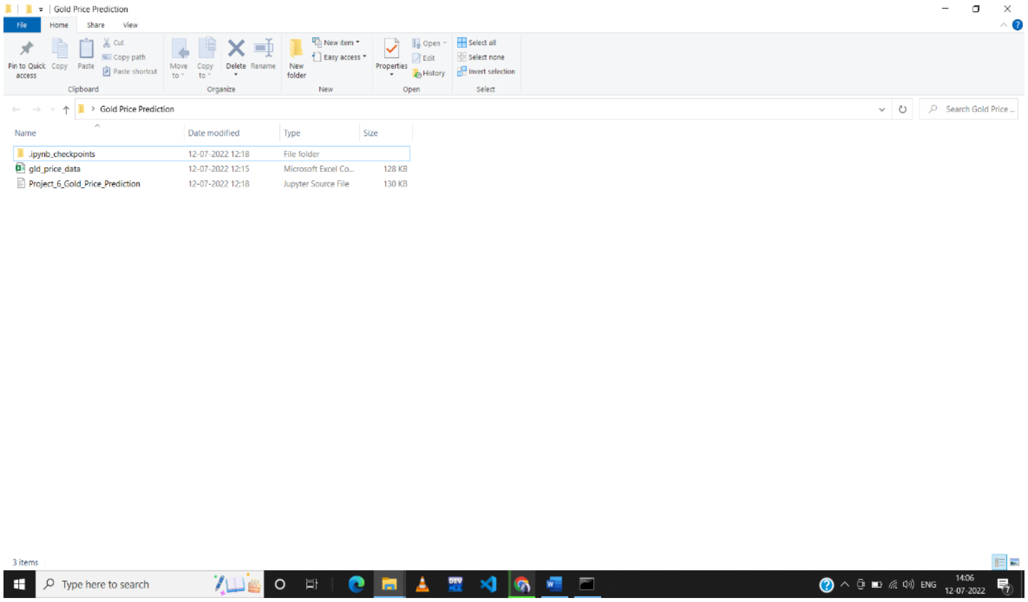
Fig. Details of Project Developed

Data flow Diagram / Algorithm


Selection of model variables for gold price forecasting
Data preprocessing
Normalization
Extract dataset
Training set
Testing set
Genetic algorithm for optimization of input layer weights and hidden layer neuron thresholds
Train the predicted model
Model
Predicted output of gold price

Input Output Datasets / screenshots

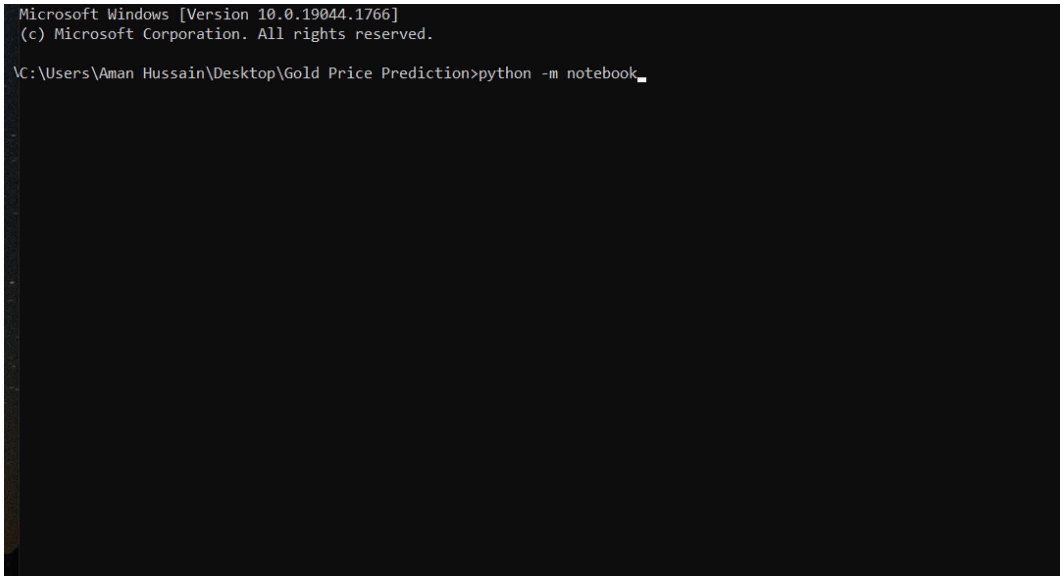
1. Gold Price Prediction File Location?



Name	Date modified	Type	Size
.ipynb_checkpoints	12-07-2022 12:18	File folder	
gld_price_data	12-07-2022 12:15	Microsoft Excel Co...	128 KB
Project_6_Gold_Price_Prediction	12-07-2022 12:18	Jupyter Source File	130 KB

Fig:1

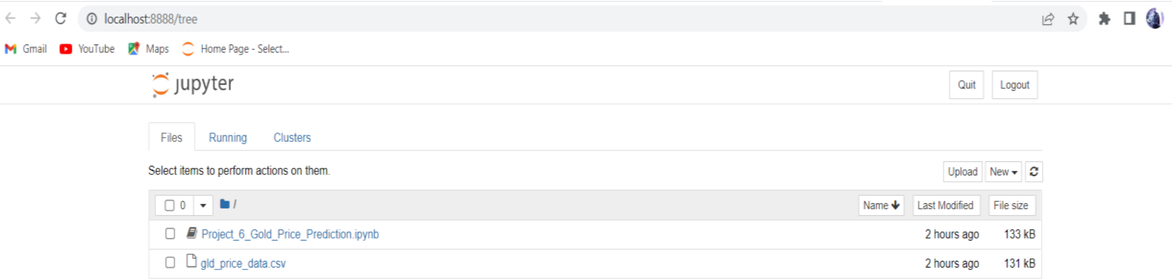
2. How to Open Gold Price Detection System



```
Microsoft Windows [Version 10.0.19044.1766]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Aman Hussain\Desktop\Gold Price Prediction>python -m notebook
```

Fig:2

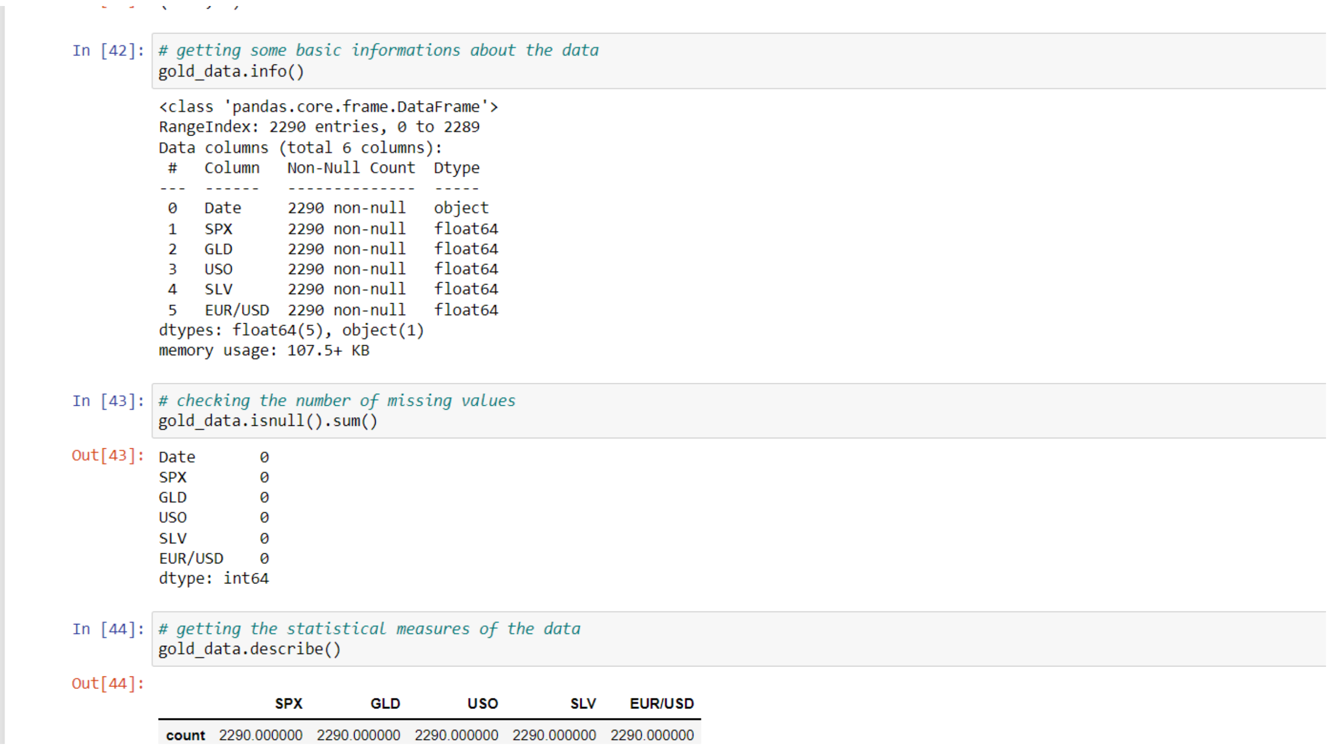
2. Gold Price Predictor Python File Location?



Name	Last Modified	File size
Project_6_Gold_Price_Prediction.ipynb	2 hours ago	133 kB
gld_price_data.csv	2 hours ago	131 kB

Fig:3

3. Getting Basic Information about data:



In [42]:
```
# getting some basic informations about the data
gold_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2290 entries, 0 to 2289
Data columns (total 6 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   Date     2290 non-null   object
 1   SPX      2290 non-null   float64
 2   GLD      2290 non-null   float64
 3   USO      2290 non-null   float64
 4   SLV      2290 non-null   float64
 5   EUR/USD  2290 non-null   float64
dtypes: float64(5), object(1)
memory usage: 107.5+ KB
```

In [43]:
```
# checking the number of missing values
gold_data.isnull().sum()
```

Out[43]:
```
Date       0
SPX        0
GLD        0
USO        0
SLV        0
EUR/USD    0
dtype: int64
```

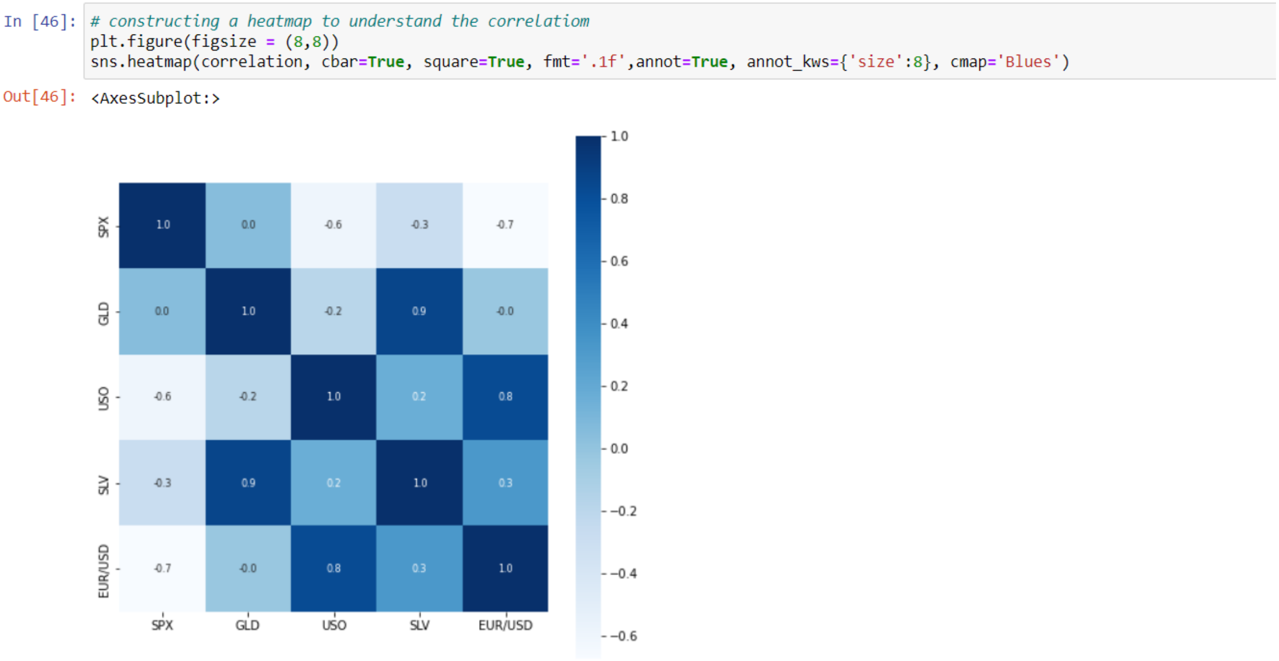
In [44]:
```
# getting the statistical measures of the data
gold_data.describe()
```

Out[44]:

	SPX	GLD	USO	SLV	EUR/USD
count	2290.000000	2290.000000	2290.000000	2290.000000	2290.000000

Fig:4

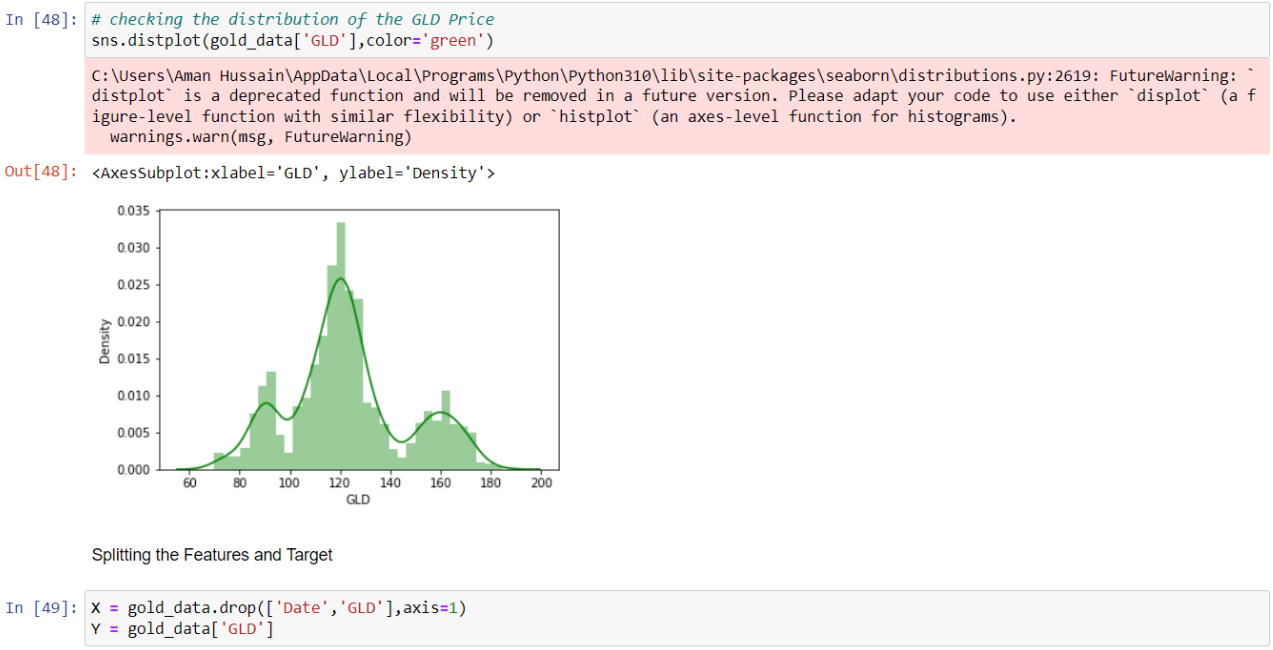
4. Constructing Heatmap To Understand Correlation:



```
In [46]: # constructing a heatmap to understand the correlatiom
         plt.figure(figsize = (8,8))
         sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')

Out[46]: <AxesSubplot:>
```

Fig:5

5. Distribution of Gold Price:



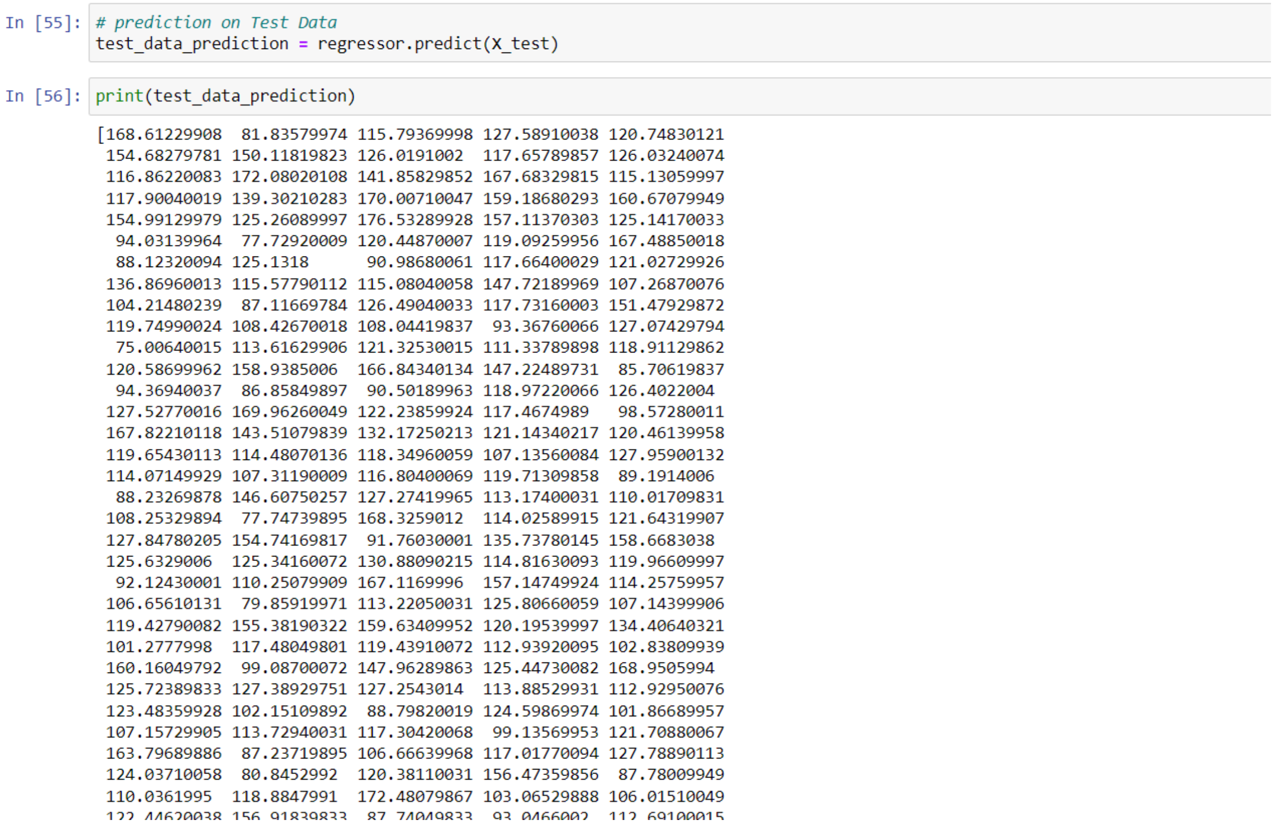
```
In [48]: # checking the distribution of the GLD Price
         sns.distplot(gold_data['GLD'],color='green')

         C:\Users\Aman Hussain\AppData\Local\Programs\Python\Python310\lib\site-packages\seaborn\distributions.py:2619: FutureWarning: `
         distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a f
         igure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
           warnings.warn(msg, FutureWarning)

Out[48]: <AxesSubplot:xlabel='GLD', ylabel='Density'>
```

Splitting the Features and Target

```
In [49]: X = gold_data.drop(['Date','GLD'],axis=1)
         Y = gold_data['GLD']
```

Fig:6

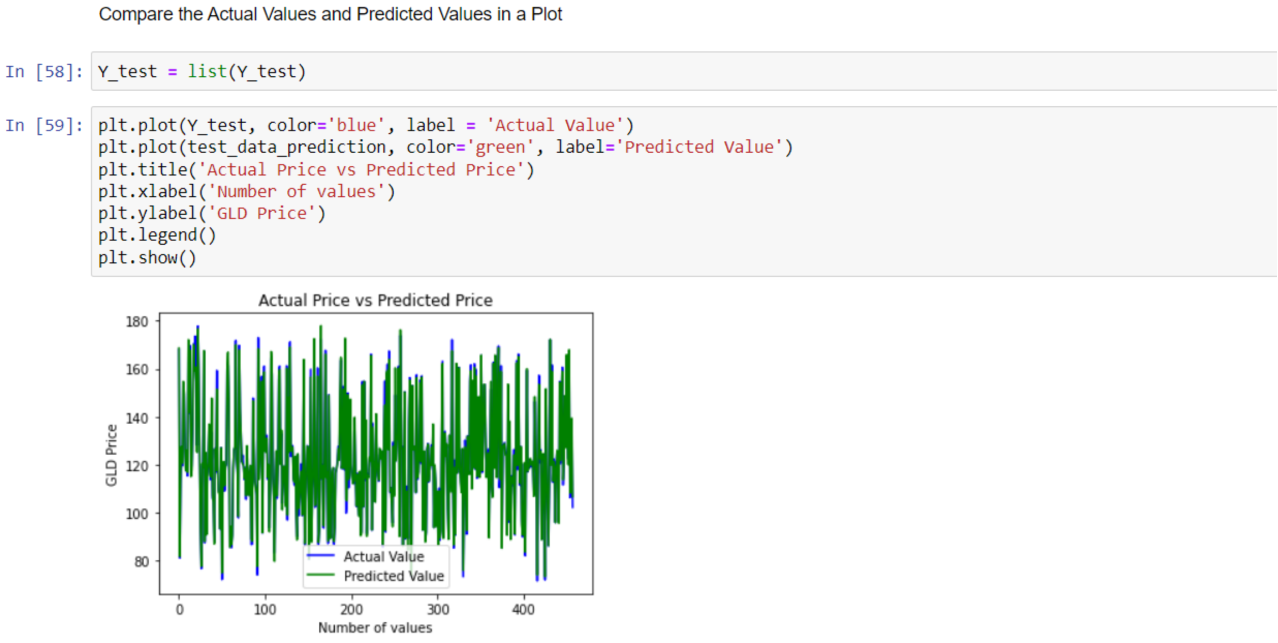
6. Prediction on Test Data:



```
In [55]: # prediction on Test Data
         test_data_prediction = regressor.predict(X_test)

In [56]: print(test_data_prediction)

[168.61229908  81.83579974 115.79369998 127.58910038 120.74830121
 154.68279781 150.11819823 126.0191002  117.65789857 126.03240074
 116.86220083 172.08020108 141.85829852 167.68329815 115.13059997
 117.90040019 139.30210283 170.00710047 159.18680293 160.67079949
 154.99129979 125.26089997 176.53289928 157.11370303 125.14170033
  94.03139964  77.72920009 120.44870007 119.09259956 167.48850018
  88.12320094 125.1318      90.98680061 117.66400029 121.02729926
 136.86960013 115.57790112 115.08040058 147.72189969 107.26870076
 104.21480239  87.11669784 126.49040033 117.73160003 151.47929872
 119.74990024 108.42670018 108.04419837  93.36760066 127.07429794
  75.00640015 113.61629906 121.32530015 111.33789898 118.91129862
 120.58699962 158.9385006  166.84340134 147.22489731  85.70619837
  94.36940037  86.85849897  90.50189963 118.97220066 126.4022004
 127.52770016 169.96260049 122.23859924 117.4674989   98.57280011
 167.82210118 143.51079839 132.17250213 121.14340217 120.46139958
 119.65430113 114.48070136 118.34960059 107.13560084 127.95900132
 114.07149929 107.31190009 116.80400069 119.71309858  89.1914006
  88.23269878 146.60750257 127.27419965 113.17400031 110.01709831
 108.25329894  77.74739895 168.3259012  114.02589915 121.64319907
 127.84780205 154.74169817  91.76030001 135.73780145 158.6683038
 125.6329006  125.34160072 130.88090215 114.81630093 119.96609997
  92.12430001 110.25079909 167.1169996  157.14749924 114.25759957
 106.65610131  79.85919971 113.22050031 125.80660059 107.14399906
 119.42790082 155.38190322 159.63409952 120.19539997 134.40640321
 101.2777998  117.48049801 119.43910072 112.93920095 102.83809939
 160.16049792  99.08700072 147.96289863 125.44730082 168.9505994
 125.72389833 127.38929751 127.2543014  113.88529931 112.92950076
 123.48359928 102.15109892  88.79820019 124.59869974 101.86689957
 107.15729905 113.72940031 117.30420068  99.13569953 121.70880067
 163.79689886  87.23719895 106.66639968 117.01770094 127.78890113
 124.03710058  80.8452992  120.38110031 156.47359856  87.78009949
 110.0361995  118.8847991  172.48079867 103.06529888 106.01510049
 122.44620038 156.91839833  87.74049833  93.0466002  112.69100015
```

Fig:7

7. Actual Value Vs Predicted Value:

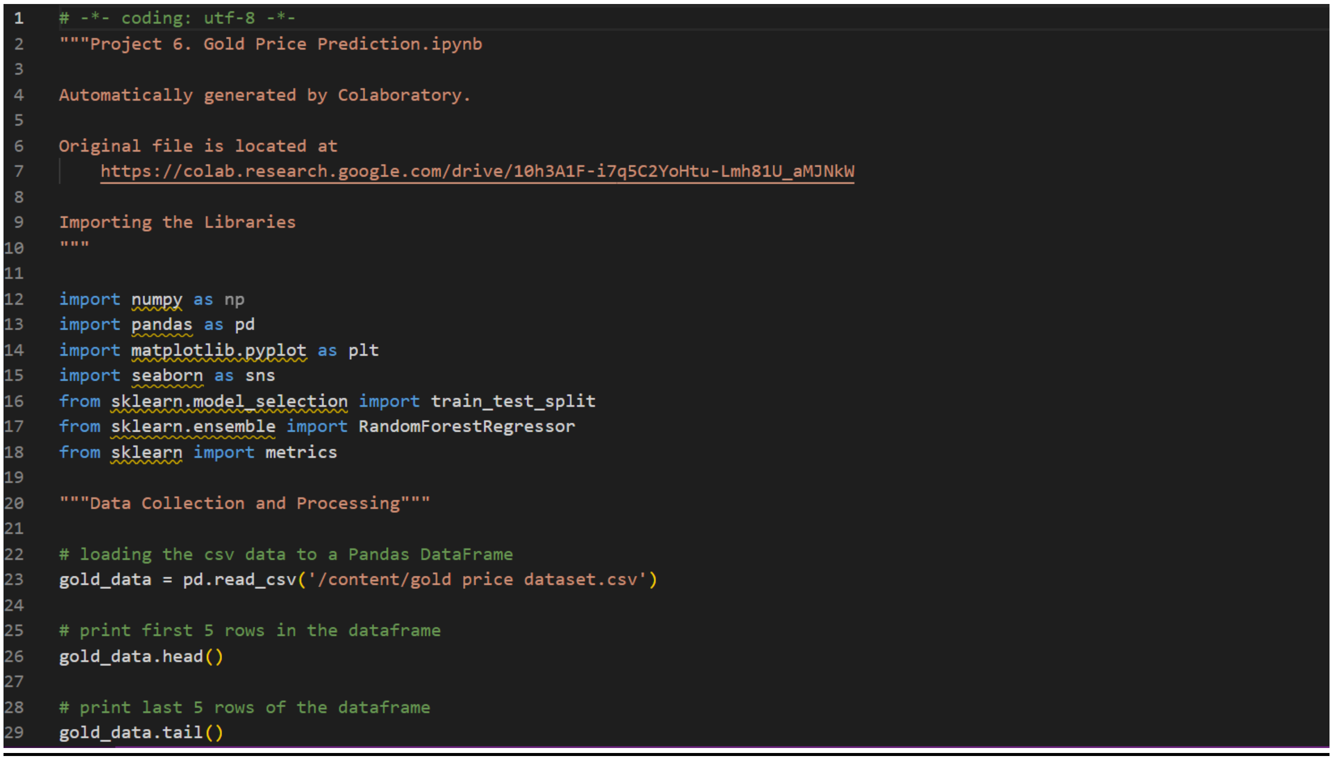


Compare the Actual Values and Predicted Values in a Plot

```
In [58]: Y_test = list(Y_test)

In [59]: plt.plot(Y_test, color='blue', label = 'Actual Value')
         plt.plot(test_data_prediction, color='green', label='Predicted Value')
         plt.title('Actual Price vs Predicted Price')
         plt.xlabel('Number of values')
         plt.ylabel('GLD Price')
         plt.legend()
         plt.show()
```

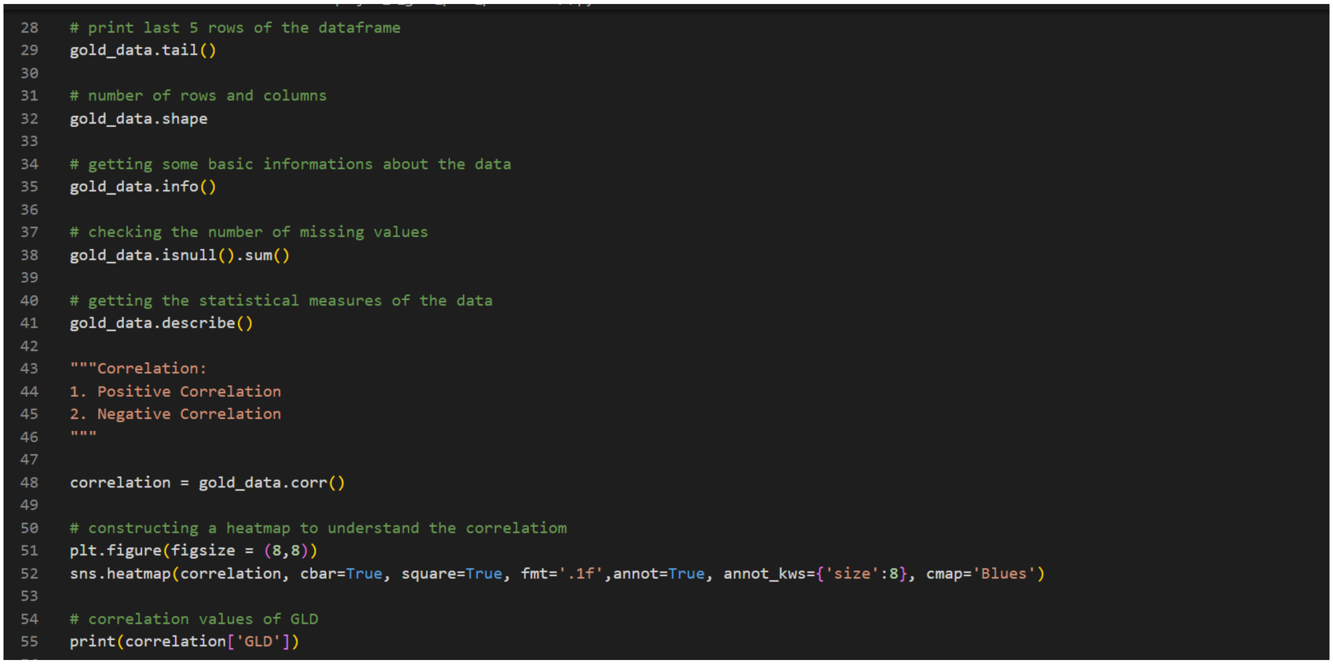
Fig :8

Text code



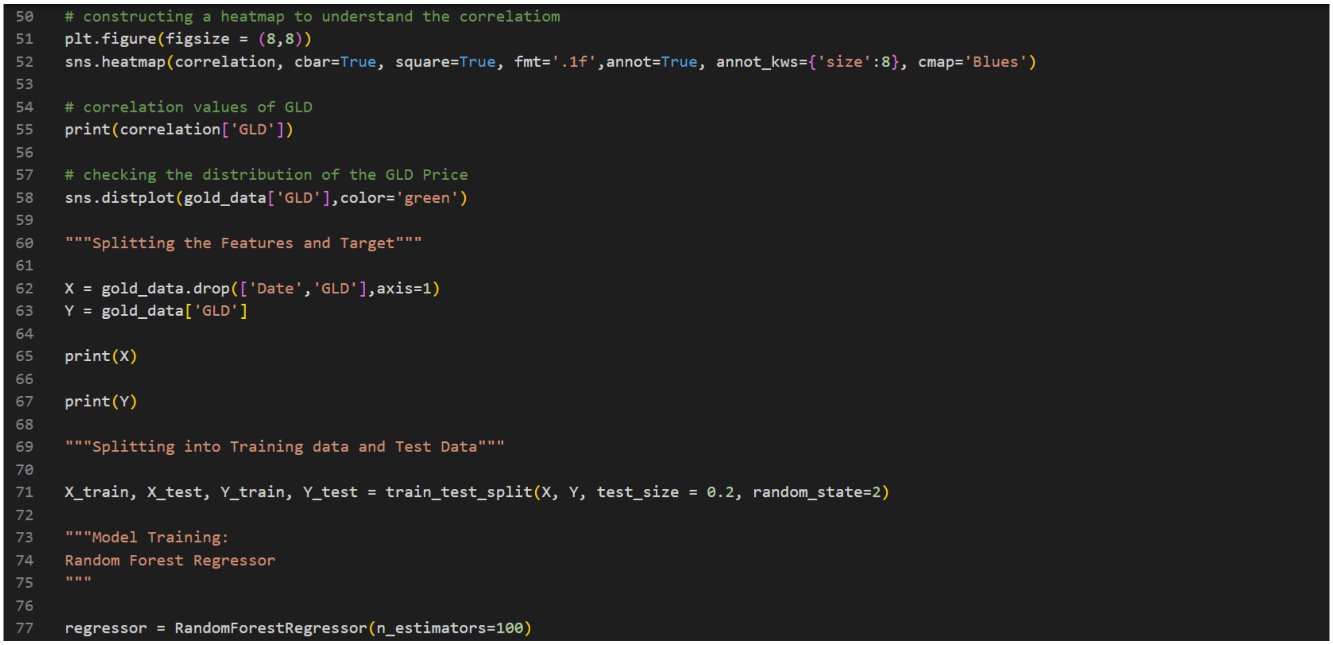
```
# -*- coding: utf-8 -*-
"""Project 6. Gold Price Prediction.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/10h3A1F-i7q5C2YoHtu-Lmh81U_aMJNkW

Importing the Libraries
"""

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn import metrics

"""Data Collection and Processing"""

# loading the csv data to a Pandas DataFrame
gold_data = pd.read_csv('/content/gold price dataset.csv')

# print first 5 rows in the dataframe
gold_data.head()

# print last 5 rows of the dataframe
gold_data.tail()
```

Fig:9



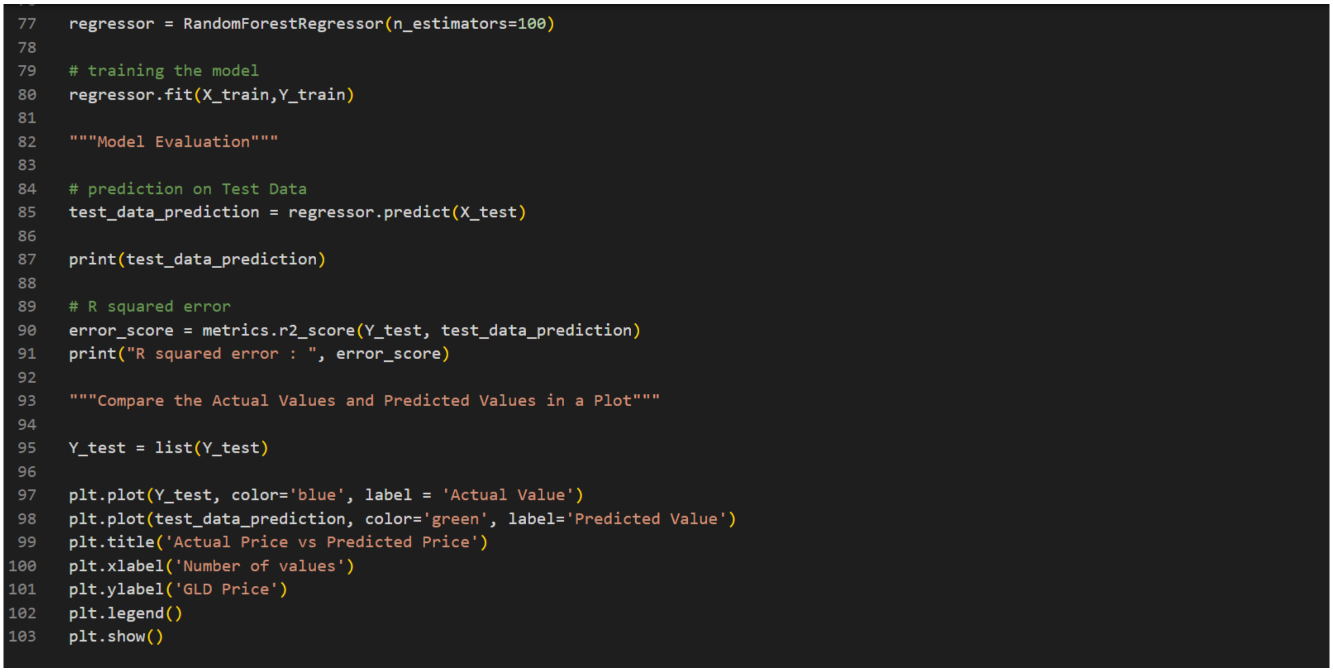
```
# print last 5 rows of the dataframe
gold_data.tail()

# number of rows and columns
gold_data.shape

# getting some basic informations about the data
gold_data.info()

# checking the number of missing values
gold_data.isnull().sum()

# getting the statistical measures of the data
gold_data.describe()

"""Correlation:
1. Positive Correlation
2. Negative Correlation
"""

correlation = gold_data.corr()

# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')

# correlation values of GLD
print(correlation['GLD'])
```

Fig:10



```
# constructing a heatmap to understand the correlatiom
plt.figure(figsize = (8,8))
sns.heatmap(correlation, cbar=True, square=True, fmt='.1f',annot=True, annot_kws={'size':8}, cmap='Blues')

# correlation values of GLD
print(correlation['GLD'])

# checking the distribution of the GLD Price
sns.distplot(gold_data['GLD'],color='green')

"""Splitting the Features and Target"""

X = gold_data.drop(['Date','GLD'],axis=1)
Y = gold_data['GLD']

print(X)

print(Y)

"""Splitting into Training data and Test Data"""

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=2)

"""Model Training:
Random Forest Regressor
"""

regressor = RandomForestRegressor(n_estimators=100)
```

Fig:11



```
regressor = RandomForestRegressor(n_estimators=100)

# training the model
regressor.fit(X_train,Y_train)

"""Model Evaluation"""

# prediction on Test Data
test_data_prediction = regressor.predict(X_test)

print(test_data_prediction)

# R squared error
error_score = metrics.r2_score(Y_test, test_data_prediction)
print("R squared error : ", error_score)

"""Compare the Actual Values and Predicted Values in a Plot"""

Y_test = list(Y_test)

plt.plot(Y_test, color='blue', label = 'Actual Value')
plt.plot(test_data_prediction, color='green', label='Predicted Value')
plt.title('Actual Price vs Predicted Price')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```

Fig:12

CONCLUSION

As you saw in this project, we first train a machine learning model, then use the trained model for prediction. Similarly, any model can be made much more precise, by feeding a very large dataset, to get a very accurate score (but it will be pretty time-consuming). For a beginner, I feel the dataset that I had used was pretty decent.

Random forest regression is found to have better prediction accuracy for the entire period and gradient boosting regression is found to give better accuracy for the two period taken separately.

References

- www.geeksforgeeks.com
- www.youtube.com
- www.wikipedia.com
- www.pycharm.com
- www.chrome.com