# Estimation of Gold's Price


Sadaf M. Anis
ID: 20101537
*Department of CSE*
*BRAC University*
Dhaka, Bangladesh
sadaf.m.anis@g.bracu.ac.bd

Lamia Khan Shoily
ID: 20301085
*Department of CSE*
*BRAC University*
Dhaka, Bangladesh
lamia.khan.shoily@g.bracu.ac.bd

Kanak Roy Shanda
ID: 21301743
*Department of CSE*
*BRAC University*
Dhaka, Bangladesh
kanak.roy.shanda@g.bracu.ac.bd

Nur-E-Jannat
ID: 21301744
*Department of CSE*
*BRAC University*
Dhaka, Bangladesh
nur.e.jannat@g.bracu.ac.bd



***Abstract*—This paper investigates machine-learning based Gold Price Prediction. In this project, we obtained a dataset of gold prices from Kaggle containing gold prices for several days over a ten-year time frame, which we will use to train our machine learning algorithm so that when we train this algorithm and give it a new value, it can predict the gold prices for that specific value.**


## I. Introduction

Gold is one of the most significant minerals in the planet. Gold has historically been associated with wealth and is frequently utilized in coinage, jewelry, and other sectors. The contemporary gold market is characterized by the coexistence of high yield and high risk. Throughout history, gold had been utilized as money in many nations, including the USA. Gold serves as a reserve in any nation despite producing lucrative commodities. A gold reserve is a quantity of gold held by a nation's central bank for the aim of providing a guarantee to be used for payments or trade on the international market, hence boosting the nation's economy. The most preferred material for investment purposes worldwide is gold. Numerous factors influence the price of gold, making price fluctuations unpredictable. As a result, the price of gold fluctuates and is unpredictable. Investors can choose whether to buy or sell gold by forecasting the daily price's rise and fall.

## II. Methodology

We will emphasize the procedure we used for our machine in this section.

### A. Data Description

A list of data is contained in the file goldprice.csv that makes up our dataset. Debdatta Chatterjee, an Associate Analyst-Data Science, posted it on the data publication platform Kaggle under the Gold Price Data Classification heading. With 2290 rows and 7 columns, this data file is in the Comma Separated Values (CSV) format. It has one column with a date format and five columns with numerical data. It is evident from the data that the variables SPX, GLD, USO, SLV, and EUR/USD were valued in relation to the dates in the date column.

| | Date | SPX | GLD | USO | SLV | EUR/USD |
|---|---|---|---|---|---|---|
| 0 | 1/2/2008 | 1447.160034 | 84.860001 | 78.470001 | 15.180 | 1.471692 |
| 1 | 1/3/2008 | 1447.160034 | 85.570000 | 78.370003 | 15.285 | 1.474491 |
| 2 | 1/4/2008 | 1411.630005 | 85.129997 | 77.309998 | 15.167 | 1.475492 |
| 3 | 1/7/2008 | 1416.180054 | 84.769997 | 75.500000 | 15.053 | 1.468299 |
| 4 | 1/8/2008 | 1390.189941 | 86.779999 | 76.059998 | 15.590 | 1.557099 |

Fig. 1. Dataset

## III. Pre-Processing Techniques

### A. Libraries Tools

NumPy and Pandas, a library for data analysis that uses two-dimensional data structures called data frames, which are identical to spreadsheets like excel, import and read the train.csv file. Another program, Scikit-Learn, supports learning algorithms and other associated features for predictive data analysis.StandardScaler(). Our project will utilise a few of its modules. Our data is presented graphically thanks to MatPlotLib, a two-dimensional charting package.

### B. Checking for Null Values

We implemented it in our code.null().sum() that indicates the absence of null values by returning a count of 0 for each characteristic. As a result, the dataset has no missing values.

```
Date        0
SPX         0
GLD         0
USO         0
SLV         0
EUR/USD     0
dtype: int64
```

Fig. 2. Check null values

### C. Checking Data Types

On the dataset, we used .info() function, which gives the data type and domain of each attribute. Every type of data is either an int64 or a float64. Further categorization is not necessary because they are all numbered.

```
ds.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
```

Fig. 3. Data types

### D. Standardization

A scaling method known as standardization centers the data with a unit standard deviation around the mean. As a result, the distribution's standard deviation is equal to one and the attribute's mean is set to zero. Additionally, this can be helpful if the data has a Gaussian distribution. This isn't always the case, though. Unlike normalization, standardization does not have a boundary range. Therefore, even if there are outliers in the data, standardization will have no effect on them.

## IV. Models

### A. Decision Tree Regressor

Decision tree regression is one of the most often used regression algorithms. The group of supervised learning algorithms includes decision tree methods. A decision tree is a tool for making decisions that has a tree structure similar to a flowchart or is a model of decisions and all of the potential outcomes. It is advised to use a decision tree because it is clear and uncomplicated. Regression using a decision tree assesses an object's properties and trains a model to resemble the structure of a tree to forecast data in the future and produce valuable continuous output. "Continuous output" refers to a non-discrete output or result. We import DecisionTreeRegressor from sklearn.tree in order to train the model using the decision tree learning approach. An accuracy score of 0.97 is obtained, which is better than that of the Linear Regression model but below that of the Random Forest Regressor model.

```
"""# **Decision Tree**"""

from sklearn.tree import DecisionTreeRegressor
dtr = DecisionTreeRegressor()
dtr.fit(x_train,y_train)

▸ DecisionTreeRegressor

dt_acc1=print("The Training accuracy of the model is {:.2f}".format(dtr.score(x_train, y_train)))
dt_acc2=print("The Testing accuracy of the model is {:.2f}".format(dtr.score(x_test, y_test)))

The Training accuracy of the model is 1.00
The Testing accuracy of the model is 0.97
```

Fig. 4. Checking accuracy of our Model using Decision Tree

### B. Linear Regressor

A machine learning algorithm based on supervised learning is linear regression. Statistical analysis is carried out utilizing this method for prediction. Predictions are produced using linear regression for real, continuous, or numerical variables. In order to attempt to anticipate the relationship between two variables, linear regression fits a linear equation to the observed data. The explanatory variable is viewed as such, and the dependent variable as such. We import the LinearRegression model to train our model from sklearn.linear model. When this is done, the accuracy on the dataset is 0.88, which is significantly lower than the accuracy of all the other models we have used in our model.

```
"""# **Linear Regressor**"""

from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(x_train,y_train)
lr.predict(x_test)

lr_acc1=print("The Training accuracy of the model is {:.2f}".format(lr.score(x_train, y_train)))
lr_acc2=print("The Testing accuracy of the model is {:.2f}".format(lr.score(x_test, y_test)))

The Training accuracy of the model is 0.88
The Testing accuracy of the model is 0.88
```

Fig. 5. Checking accuracy of our Model using Linear Regression

### C. Random Forest

The supervised machine learning method known as random forest is a well-liked approach for classification and regression issues. It uses a variety of samples to generate decision trees, using the majority of them for categorization and the average of them for regression. The RandomForestRegressor is imported from sklearn.ensemble. The accuracy of this model, which is 0.98, is marginally lower than that of the KNeighborsRegressor model.

```
"""# **Random Forest**"""

#we will train our model using random forest regressor
from sklearn.ensemble import RandomForestRegressor
reg = RandomForestRegressor(n_estimators=100)
reg.fit(x_train, y_train)

#let our model predict on test data
test_data_prediction = reg.predict(x_test)
#print(test_data_prediction)

y_Pred = reg.predict(x_test)
y_Pred

rf_acc1=print("The Training accuracy of the model is {:.2f}".format(reg.score(x_train, y_train)))
rf_acc2=print("The Testing accuracy of the model is {:.2f}".format(reg.score(x_test, y_test)))
```

```
The Training accuracy of the model is 1.00
The Testing accuracy of the model is 0.98
```

Fig. 6. Checking accuracy of our Model using Random Forest

### D. KNeighborsRegressor

We import KNeighborsRegressor from the Scikit Learn module neighbors into our application, then we model train with the default value of nneighbors=3. In comparison to the other models on the dataset, this one gives the greatest accuracy of 0.99.

```
"""# **KNN model**"""

from sklearn.neighbors import KNeighborsRegressor
knn = KNeighborsRegressor(n_neighbors=3).fit(X_train_std, y_train)
y_knn = knn.predict(X_test_std)

y_knn

knn_acc1=print("Training accuracy is {:.2f}".format(knn.score(X_train_std, y_train)) )
knn_acc2=print("Testing accuracy is {:.2f} ".format(knn.score(X_test_std,y_test)) )
```

```
Training accuracy is 1.00
Testing accuracy is 0.99
```

Fig. 7. Checking accuracy of our Model using KNeighborsRegressor

## V. Find Error

In this step, we actually compared the actual value to our anticipated value and determined the error between the two. The computed error was around 0.9849880301288905.

```
"""# **Error Finding**"""

#now we will compare predicted values with our actual values
from sklearn import metrics
error = metrics.r2_score(y_test, test_data_prediction)
#error=error*100
print('error ',error)

#visulaization by plotting a graph
y_test = list(y_test)
print(y_test)
plt.plot(y_test, color='red', label = 'actual values')
plt.plot(test_data_prediction, color='green', label='predicted values')
plt.title('actual values vs predicted values')
plt.xlabel('Number of values')
plt.ylabel('GLD Price')
plt.legend()
plt.show()
```

```
error  0.9852103429280301
[110.82, 151.050003, 137.660004, 113.07, 114.629997, 96.18, 110.459999, 95.
```

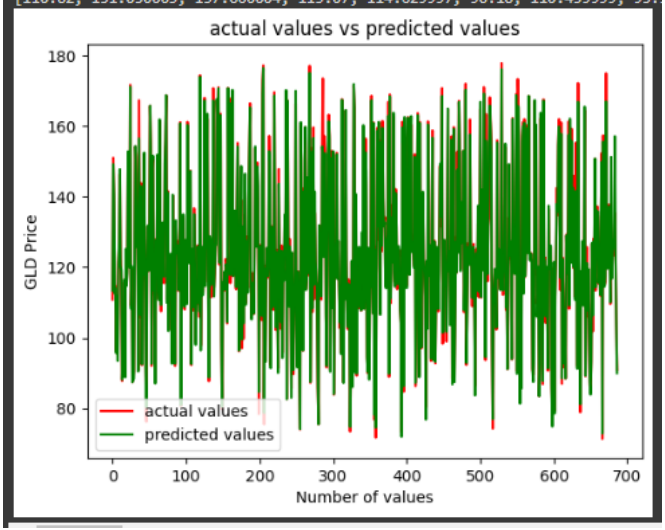


Fig. 8. comparing accuracy between the actual value and predicted value

## VI. Conclusion

The KNeighborsRegressor model clearly produces the highest accuracy score (0.99), while the Linear Regressor model generates the lowest accuracy score (0.88), as shown by our explanations of the four models. The random forest model yields a score of 0.98, which is fair. The Decision Tree model produces an accuracy of 0.97, and that is the end result. In light of the available data, the KNeighborsRegressor model can most accurately forecast the price of gold.