NATIONAL INSTITUTE OF TECHNOLOGY SIKKIM

Machine Learning
(EC16105)
Project Report
Academic Year 2023-2024

Submitted by:
Kanak (B210049EC)
Avantash Ranjan (B210076EC)
Sonu Kumar Bhagat (B210066EC)
Semester VI

Submitted to:
Dr. Hemant Kumar Kathania
(Course instructor)
Department of ECE, NIT Sikkim

Contents:

Abstract:

The project, titled "GOLD PRICE PREDICTIONS", predicts gold prices based on last year's gold data. The main purpose of the program is to help investors decide when to buy or sell gold by predicting the rise and fall of gold prices every day. Inventory forecasting plays an important role in the financial success of a business. The gold price is calculated by analysing the dataset containing the price of gold from the previous year. The rise in gold prices, along with fluctuations and price declines in other markets such as the capital market and real estate market, has led more and more investors to look at gold as a copy investment tool. There is concern that these high prices will continue and prices will reverse. There are many studies examining the relationship between gold prices and some financial variables. We use machine learning techniques to predict financial fluctuations and focus on predicting the price of gold.

Objective:

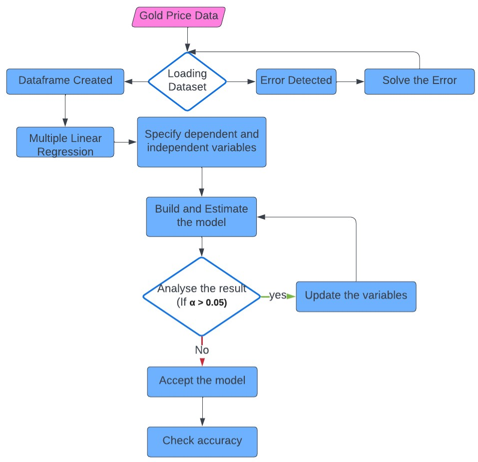
The main objectives of the project are:
1. This project is based on the applicability of the proposed machine learning algorithms that had demonstrated their efficiency to predict gold prices with a better predictive rate.
2. To apply the best appropriate Machine Learning procedure.
3. We proposed the development of a prediction model for predicting future gold prices using Multiple Linear Regression, Random Forest, KNN.

Dataset Variations:

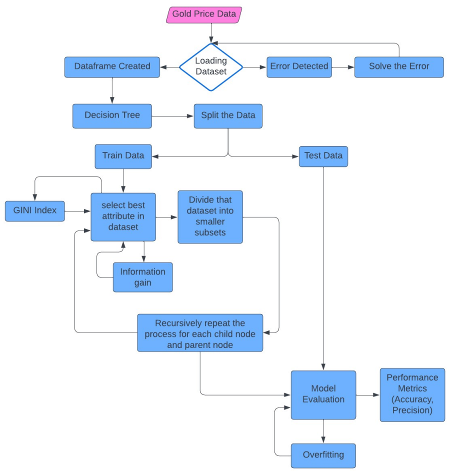
	Date	SPX	GLD	USO	SLV	EUR/USD
1	1/2/2008	1447.160034	84.860001	78.470001	15.18	1.471692
2	1/3/2008	1447.160034	85.57	78.370003	15.285	1.474491
3	1/4/2008	1411.630005	85.129997	77.309998	15.167	1.475492
4	1/7/2008	1416.180054	84.769997	75.5	15.053	1.468299
5	1/8/2008	1390.189941	86.779999	76.059998	15.59	1.557099
6	1/9/2008	1409.130005	86.550003	75.25	15.52	1.466405
7	1/10/2008	1420.329956	88.25	74.019997	16.061001	1.4801
8	1/11/2008	1401.02002	88.580002	73.089996	16.077	1.479006
9	1/14/2008	1416.25	89.540001	74.25	16.280001	1.4869
10	1/15/2008	1380.949951	87.989998	72.779999	15.834	1.48021
11	1/16/2008	1373.199951	86.699997	71.849998	15.654	1.466405
12	1/17/2008	1333.25	86.5	71.029999	15.717	1.464
13	1/18/2008	1325.189941	87.419998	71.540001	16.030001	1.461796
14	1/22/2008	1310.5	88.169998	70.550003	15.902	1.464794
15	1/23/2008	1338.599976	87.889999	69.5	15.9	1.463208
16	1/24/2008	1352.069946	90.080002	70.93	16.299999	1.47741
17	1/25/2008	1330.609985	90.300003	71.910004	16.298	1.467502
18	1/28/2008	1353.959961	91.75	72.349998	16.549999	1.478809

Flowcharts:

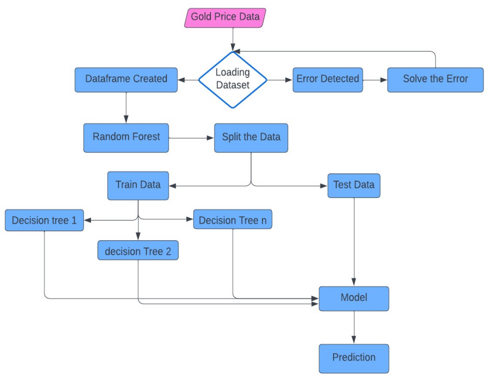
Multiple Linear Regression:



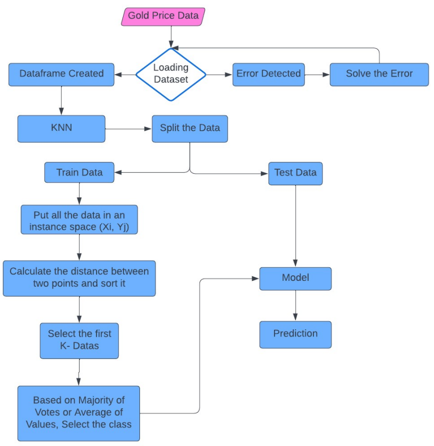
Decision Tree:



Random Forest:



KNN:



Codes:

Multiple Linear Regression:

```
class MultipleLinearRegression:

    def __init__(self, learning_rate=0.01, iterations = 2000):
        self.learning_rate=learning_rate
        self.iterations=iterations
        self.cofficients = None
        self.intercept = None

    def predict(self, X):
        return self.intercept + self.cofficients * X

    def fit(self,X,y):
        self.cofficients = 0
        self.intercept = 0
        n=len(X)

        for i in range(self.iterations):
            y_predicted = self.predict(X)

            d_cofficient = (-2/n)*sum(X*(y-y_predicted))
            d_intercept =  (-2/n) *sum((y-y_predicted))

            self.cofficients -= self.learning_rate * d_cofficient
            self.intercept -= self.learning_rate * d_intercept
```

Decision Tree:

```
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score
import pandas as pd

data = pd.read_csv('gld_price_data.csv')

X = data.drop(['Date','GLD'], axis = 1)
y = data['GLD']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42

regressor = DecisionTreeRegressor()

# Train the model
regressor.fit(X_train, y_train)

# Make predictions
predictions = regressor.predict(X_test)
```

Random Forest:

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

gold_data = pd.read_csv('gld_price_data.csv')
gold_data.head()

#to check any missing values
gold_data.isnull().sum()

 #splitting the fearures and target
X = gold_data.drop(['Date','GLD'], axis = 1)
y = gold_data['GLD']

X_train , X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 2)

regressor.fit(X_train, y_train)
regressor = RandomForestRegressor(n_estimators = 100)

#prediction on test data
test_data_prediction = regressor.predict(X_test)
```

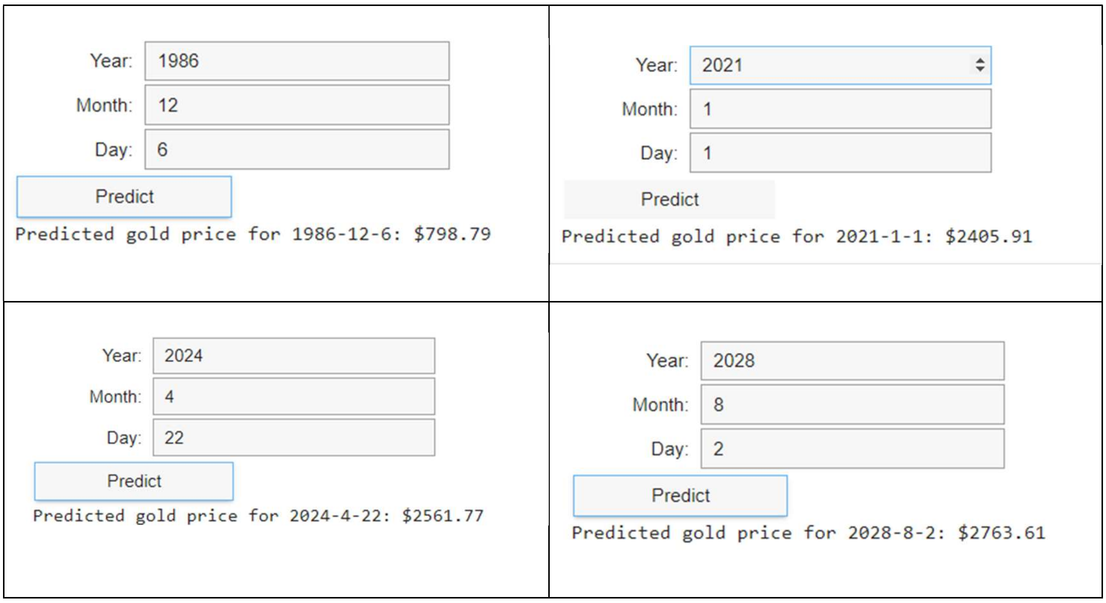
KNN:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error

gold_data = pd.read_csv('gld_price_data.csv')

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

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)


k = 3
regressor = KNeighborsRegressor(n_neighbors=k)

regressor.fit(X_train_scaled, y_train)

predictions = regressor.predict(X_test_scaled)
```

Accuracy:

Algorithms	R2_Score	Mean Square Error
Multiple Linear Regression	0.8975640982991402	56.16559421500603
Decision Tree	0.9831109316901961	9.260274392156793
Random Forest	0.9892766469309671	5.655962881475242
KNN	0.9941851307615867	3.188292196731859

Outputs:



Conclusion:

This project aimed at understanding the relationship between gold prices and options that affect gold prices such as stocks, oil prices, rupee dollar exchange rate, silver price. This study uses monthly price data from January 2000 to December 2018. The data is divided into two periods. The first period was from January 2000 to October 2011, when gold prices increased; Four machine learning algorithms were used to analyse this data: linear regression, decision tree, random forest regression and KNN. The relationship between the variables is strong in stage I, and strong in stage II. It was found to be weak at this stage. While this model fit the first-level data well, the second-level fit was poor. KNN has a better prediction accuracy for the entire time- period, while multiple linear regression shows less accuracy as it best suits for the data having linear relationship between its features and target.

Kernel Functions:

1. **Polynomial Kernel:**
   It represents the similarity of vectors in the training set of data in a feature space over polynomials of the original variables used in the kernel.

$$f(x1, x2) = (x1^T . x2 + 1)^d$$

2. Sigmoid Kernel:

It is used for taking input, mapping them to a value of 0 and 1 so that they can be separated by a simple straight line.

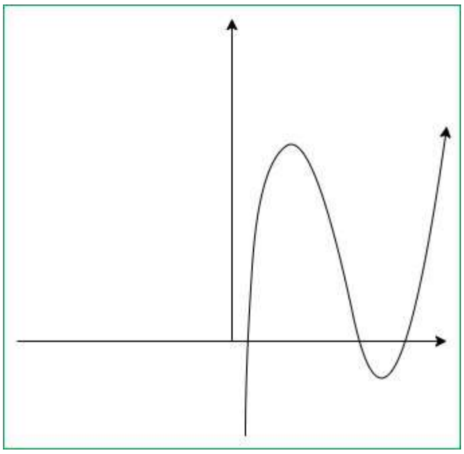
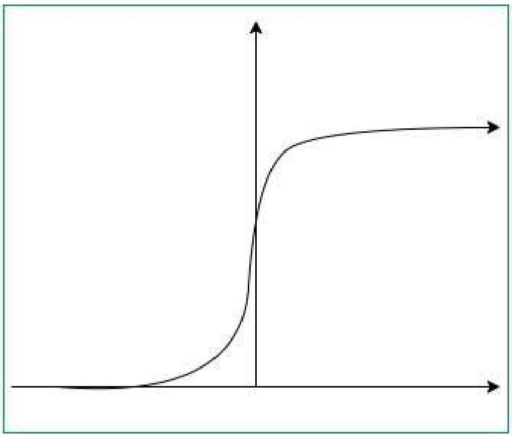
$$\boldsymbol{f(x1, x2) = tanh(\alpha x^T . y + x)}$$

3. RBF Kernel:

Radial **B**asis **F**unction (RBF) is used to create non-linear combinations of our features to lift our samples onto a higher-dimensional feature space where we can use a linear decision boundary to separate our classes. It is the most used kernel in SVM classifications.

Mathematically it can be represented as:

$$\boldsymbol{f(x1, x2) = e^{-||x1-x2||^2/2\sigma^2}}$$

Polynomial Kernel	***Sigmoid Kernel***
	

RBF Kernel