Course/Programme	BSc (Hons) Computing
Module name and code:	Emerging Technologies (SWE6206-22-UOB-A)
Student ID	2011184
Tutor:	Pradeep Hewage
Assessment Number:	2 of 2
Assessment Type:	Coursework
Assessment Title:	Portfolio – Component 4
Weighting	40% of overall module grade
Issue Date:	01/10/2022
Submission Deadline:	13/01/2023 at 23:59
Submission Date:	15/01/2023

Learning Outcomes assessed:

LO1: Construct a prototype or simulation demonstrating the potential operation of an emerging technology to solve a given problem.

Contents

Table of Figures

1. Introduction

The fourth version of the industrial revolution, is transforming the way companies produce, enhance, and distribute their products. Companies are incorporating new technologies such as the Internet of Things (IoT), cloud computing, analytics, augmented and virtual reality, and AI and machine learning into their operations and production facilities in order to improve their efficiency and profit. These digital technologies lead to increased automation, predictive maintenance, self-optimisation of processes, and a new level of efficiency and responsiveness to customers. Replacing manual inspection models with AI-powered visual insights reduces manufacturing errors, saves time and money. Applying machine learning algorithms can detect errors quickly, rather than at later stages when repairs are more costly. These Industry 4.0 concepts and technologies can be applied across all industrial sectors and are being developed in a fast pace. The scope of this work focused on the application of Machine Learning (ML) and the algorithms it incorporates to accurately predict the price of gold, so the user can have a clear picture of how it can fluctuate in future.

2. Aims and Objectives

2.1. Aims

The aim of this project is to use machine learning techniques to project predictions for the price of gold.

2.2. Objectives

The objectives of this project are:

- To analyse and select the appropriate machine learning algorithm that is most suitable for the task
- Asses the data related to the specific problem
- Clean and prepare the data for the task
- Build the selected model
- Test and evaluate the proposed model and report its performance
- Compare with other selected models

3. Literature Review

3.1. What is Machine Learning?

Machine learning is a subfield of AI that allows computers to learn from data without human intervention. It enables systems to improve their performance automatically. Machine learning algorithms are used to identify patterns in data, make predictions and even take decisions. There are several tasks that the different models are able to support humans with, namely:

- Descriptive function – data is utilised to clarify what events occurred
- Predictive – gives an explanation and predictions as to what will happen

- Prescriptive – provides the user with proposals for further actions to be taken (Brown, 2021; Bell, 2014; Cambridge University Press, 2023)

There are different approaches in machine learning such as:

- Supervised learning - the system is provided with labelled data to be trained on and improve the accuracy of the model's predictions
- Unsupervised learning - no labelled data is provided and it is used to push the program to identify patterns in the data that might not be obvious for a human
- Reinforcement learning - a method of teaching computers through trial and error via the implementation of a reward system. Hence, when the model is taking an action, this can be evaluated and scored through it for future reference, so the model would re-adjust the actions taken (Bell, 2014; Brown, 2021)

Machine learning has a wide range of applications such as image and speech recognition, natural language processing and autonomous vehicles(Bell, 2014; Brown, 2021).

It goes through several steps in general:

- Data collection – it is assembled and processed in order to be utilised as a base for the algorithm's training. The efficiency of the model is improved, if there are larger amounts of data to be trained on.
- Choice of appropriate model – the developer needs to do that, so the gathered data can be fed to it. Then the model would train itself to predict values or find patterns in the data. This stage is also characterised by the fact that the programmer can re-adjust the model to improve its accuracy.
- Testing the model's accuracy – slices of the dataset are purposely left out of the training phase, so the model can be evaluated and its predictions can be compared to the actual data. This would improve the model's accuracy when working with new data (Brown, 2021; Bell, 2014).

3.2. Machine Learning Algorithms

Machine Learning utilises various algorithms to address data issues. Data scientists often emphasise that there is not a single algorithm that is universally superior for resolving all problems. The algorithm employed is based on the type of problem to be addressed, the number of variables, the type of model that would be most appropriate, etc. Here's a brief overview of some of the algorithmic models often applied in machine learning.

3.2.1. Gradient Descent Algorithm

It's an iterative algorithm, whose goal is to minimise a cost function. It is necessary to be able to calculate the partial derivative, also known as the slope or gradient, of the function. The coefficients are calculated at each iteration by finding the negative derivative and reducing the coefficients by a specified learning rate, which is multiplied by the derivative. This process continues until the local minimum of the cost function is reached, and the iterations are stopped when the minimum value is achieved and there is no further reduction in the cost function (Ray, 2019; Mahesh, 2020).

Model variations:

- Batch Gradient Descent – update is performed only after all training sets are assessed
- Stochastic Gradient Descent – updates are issued after each training set
- Mini Batch Gradient Descent – is a mixture of the above (Ray, 2019; Mahesh, 2020)

Disadvantages:

- Speed of learning affects performance(Ray, 2019; Mahesh, 2020)

3.2.2. Linear Regression

Part of the supervised learning algorithms. Utilised to work with continuous variables and produce predictions. It works with labelled data. The output prediction is affected by the input variable. The name of this regression comes from the hyperplane line that is drawn through the data(Ray, 2019; Mahesh, 2020).

Advantages:

- Ease of use
- Shorter learning curve
- Overfitting of data is a lesser issue with this model(Ray, 2019; Mahesh, 2020)

Disadvantages:

- Does not perform well when non-linear relationships are involved
- Not very efficient with complex patterns(Ray, 2019; Mahesh, 2020)

3.2.3. Multivariate Regression

Here, multiple independent variables are affecting a single dependant one. However, the inclusion of more independent variables does not guarantee better performance of the algorithm. With this type of regression, it is really important that multicollinearity between the independent variables is not present. They should be only correlated to the dependent one for good performance of the model(Ray, 2019; Mahesh, 2020).

Advantages:

- Provides with good information about the connection between independent-dependant variables
- Also, it describes the connection between the dependant ones(Ray, 2019; Mahesh, 2020)

Disadvantages:

- Requires a certain level of statistical experience
- Harder learning curve(Ray, 2019; Mahesh, 2020)

3.2.4. Decision Tree

It is a part of the supervised learning group of algorithms. Decision trees aim to resolve issues of categorisation or prediction by constantly slicing the data into fragments according to a provided parameter. The way the algorithm works is by separating the data in its nodes, while the choices are being made within its leaves.

Advantages:

- A good fit for both categorisation and variable prediction
- Ease of use
- Can populate values that are not present with the one that is most likely
- Good level of performance(Ray, 2019; Mahesh, 2020)

Disadvantages:

- The dimensions of the tree are hard to manage
- Can be unstable (Ray, 2019; Mahesh, 2020)

3.2.5. Support Vector Machine

- Tackles classification and regression issues. Here, it is really important that a hyperplane is defined, because it outlines the margin of the decision, because objects that go under different are hard to be split otherwise. These objects can be split by kernels. The algorithm has as a goal to classify them by using the training data as a foundation(Ray, 2019; Mahesh, 2020).

Advantages:

- Performs well with both semi-structured and structured data
- The level of overfitting of data is not so high
- Data that utilises multiple dimensions can be efficiently processed by the algorithm's scaling ability(Ray, 2019; Mahesh, 2020)

Disadvantages:

- Performance is affected by the size of the data set

- Kernels that are needed for each data set need experience to locate
- Noisy data affects the functioning of the model
- Harder learning curve
- Not able to fill in missing values(Ray, 2019; Mahesh, 2020)

3.2.6. Naïve Bayes

This model works with conditional probability. The model resembles a table that is filled in by the utilisation of the training data. The "probability table" is based on the feature values of an observation, where one must look up the class probabilities in order to predict a new observation. The basic assumption is of conditional independence, which is why it is referred to as "naive". However, in real-world contexts, it is unlikely that all input features are truly independent of one another(Ray, 2019; Mahesh, 2020).

Advantages:

- Easy to use
- Performing well
- Can use lower amounts of data to train
- Scalable
- Not sensitive to feature that have no connection(Ray, 2019; Mahesh, 2020)

Disadvantages:

- Too simple
- Difficult to work with when having continuous variables
- If the number of variables is increased the model becomes very cost ineffective from an operational point of view(Ray, 2019; Mahesh, 2020)

4. Methodology

After reviewing some of the available machine learning algorithms that are used in industry, the choice was made to continue with Multivariate Linear Regression to predict the future price of gold. It falls under the supervised learning group of algorithms. This choice was made due to the fact that continuous variables are involved, hence the choice of algorithm(Ray, 2019; Mahesh, 2020).

5. Data Collection, Cleaning and Preparation

5.1. Data Collection

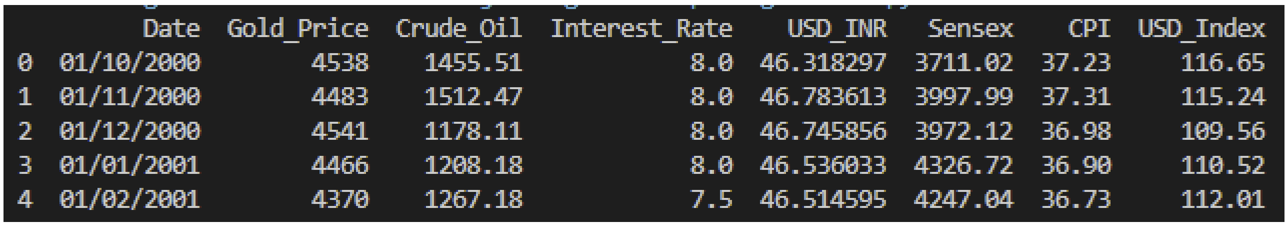
The dataset that was used for the delivery of this work can be accessed and downloaded here:

https://www.kaggle.com/datasets/somyaagarwal69/gold-forecasting

The observations in the dataset span in time from 01/10/2000 to 01/08/2020. It is important to note that the prices are in Indian Rupees, hence the existence of columns like the USD exchange rate, CPI (Customer Price Index), Sensex, USD_Index.

5.2. Data Analysis

The analysis of the dataset was started via using the **head()** function in order to view a slice of the data (Figure 1).

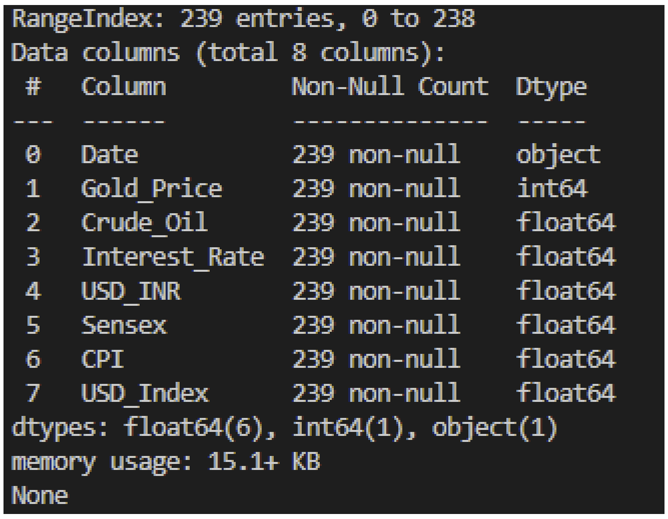


```
         Date  Gold_Price  Crude_Oil  Interest_Rate    USD_INR   Sensex    CPI  USD_Index
0  01/10/2000        4538    1455.51            8.0  46.318297  3711.02  37.23     116.65
1  01/11/2000        4483    1512.47            8.0  46.783613  3997.99  37.31     115.24
2  01/12/2000        4541    1178.11            8.0  46.745856  3972.12  36.98     109.56
3  01/01/2001        4466    1208.18            8.0  46.536033  4326.72  36.90     110.52
4  01/02/2001        4370    1267.18            7.5  46.514595  4247.04  36.73     112.01
```

Figure 1 - head() function applied to the dataset

This function allows the to see the first five tuples of the dataset and what can be concluded from that is that it consists of 8 series, none of which is a categorical one.

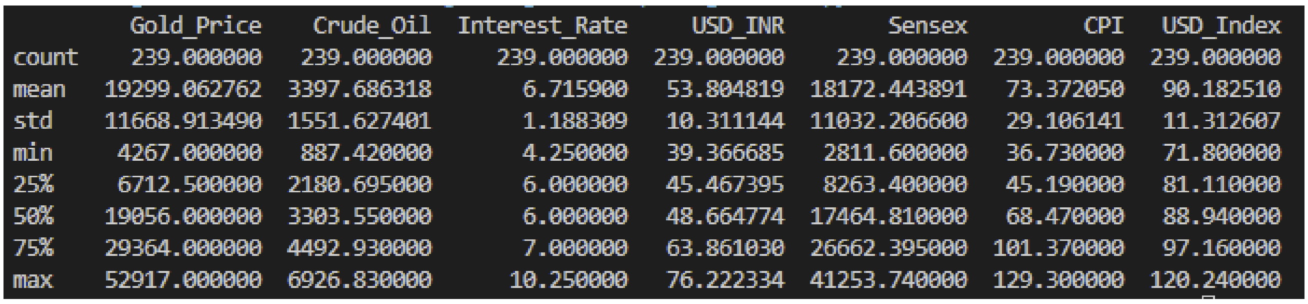
Next, the **info()** function was utilised. This also showed that there are 8 columns in the dataset with their respective headers and 239 observations. The dataset also contains float, integer and object values (dates). The function also shows that the memory usage of the dataset is 15.1 KB (Figure 2).



```
RangeIndex: 239 entries, 0 to 238
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Date           239 non-null    object
 1   Gold_Price     239 non-null    int64
 2   Crude_Oil      239 non-null    float64
 3   Interest_Rate  239 non-null    float64
 4   USD_INR        239 non-null    float64
 5   Sensex         239 non-null    float64
 6   CPI            239 non-null    float64
 7   USD_Index      239 non-null    float64
dtypes: float64(6), int64(1), object(1)
memory usage: 15.1+ KB
None
```

Figure 2 - info() function applied to the dataset

Afterwards the **describe()** command was used to provide further intel (Figure 3).



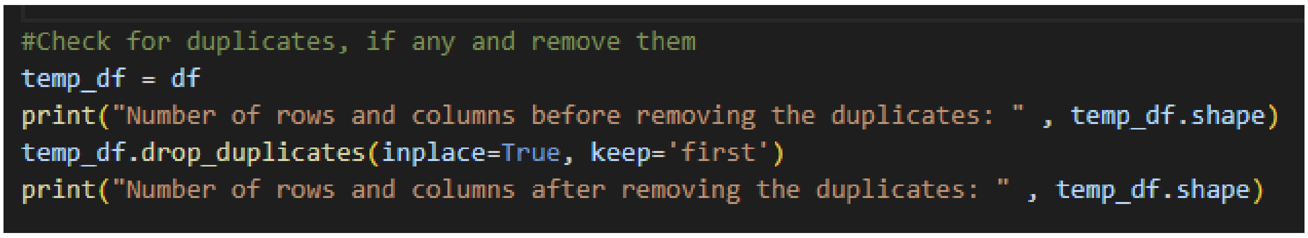
```
         Gold_Price    Crude_Oil  Interest_Rate     USD_INR        Sensex         CPI   USD_Index
count    239.000000   239.000000     239.000000  239.000000    239.000000  239.000000  239.000000
mean   19299.062762  3397.686318       6.715900   53.804819  18172.443891   73.372050   90.182510
std    11668.913490  1551.627401       1.188309   10.311144  11032.206600   29.106141   11.312607
min     4267.000000   887.420000       4.250000   39.366685   2811.600000   36.730000   71.800000
25%     6712.500000  2180.695000       6.000000   45.467395   8263.400000   45.190000   81.110000
50%    19056.000000  3303.550000       6.000000   48.664774  17464.810000   68.470000   88.940000
75%    29364.000000  4492.930000       7.000000   63.861030  26662.395000  101.370000   97.160000
max    52917.000000  6926.830000      10.250000   76.222334  41253.740000  129.300000  120.240000
```

Figure 3 - describe() used

Here, we can see that each column has 239 non-empty rows as the “count” section states. The mean value for each column, the standard deviation, minimum and maximum value, percentiles.

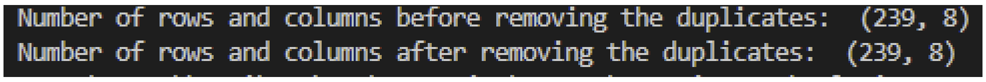
5.3. Data Cleaning

First, the dataset was checked for duplicate values through the creation of a **temp_df**, which is a temporary data frame copy of the existing dataset, so the original one would not be affected. This check proved that there were no missing values (Figures 4 & 5).



```
#Check for duplicates, if any and remove them
temp_df = df
print("Number of rows and columns before removing the duplicates: " , temp_df.shape)
temp_df.drop_duplicates(inplace=True, keep='first')
print("Number of rows and columns after removing the duplicates: " , temp_df.shape)
```

Figure 4 - Code used to check for duplicates

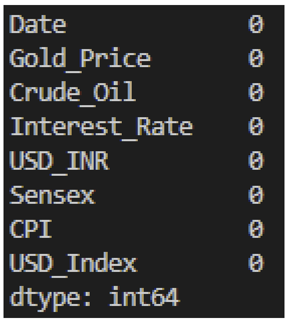


```
Number of rows and columns before removing the duplicates:  (239, 8)
Number of rows and columns after removing the duplicates:  (239, 8)
```

Figure 5 - Result of duplicates check

The next task to be completed was to check for missing values through the code in Figure 6.

The result showed that there were no missing values in our data (Figure 7).

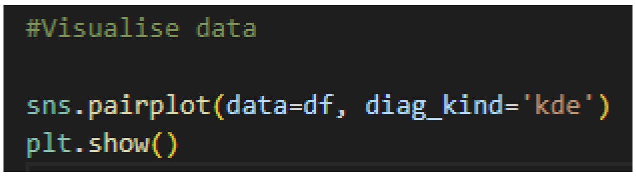


```
Date             0
Gold_Price       0
Crude_Oil        0
Interest_Rate    0
USD_INR          0
Sensex           0
CPI              0
USD_Index        0
dtype: int64
```

Figure 6 - Missing value check

5.4. Data Visualisation

After the data was cleaned it had to be visualised to gain a bit more insight of how the values were spread (Figures 7 & 8).



```
#Visualise data

sns.pairplot(data=df, diag_kind='kde')
plt.show()
```

Figure 7 - Data visualisation code

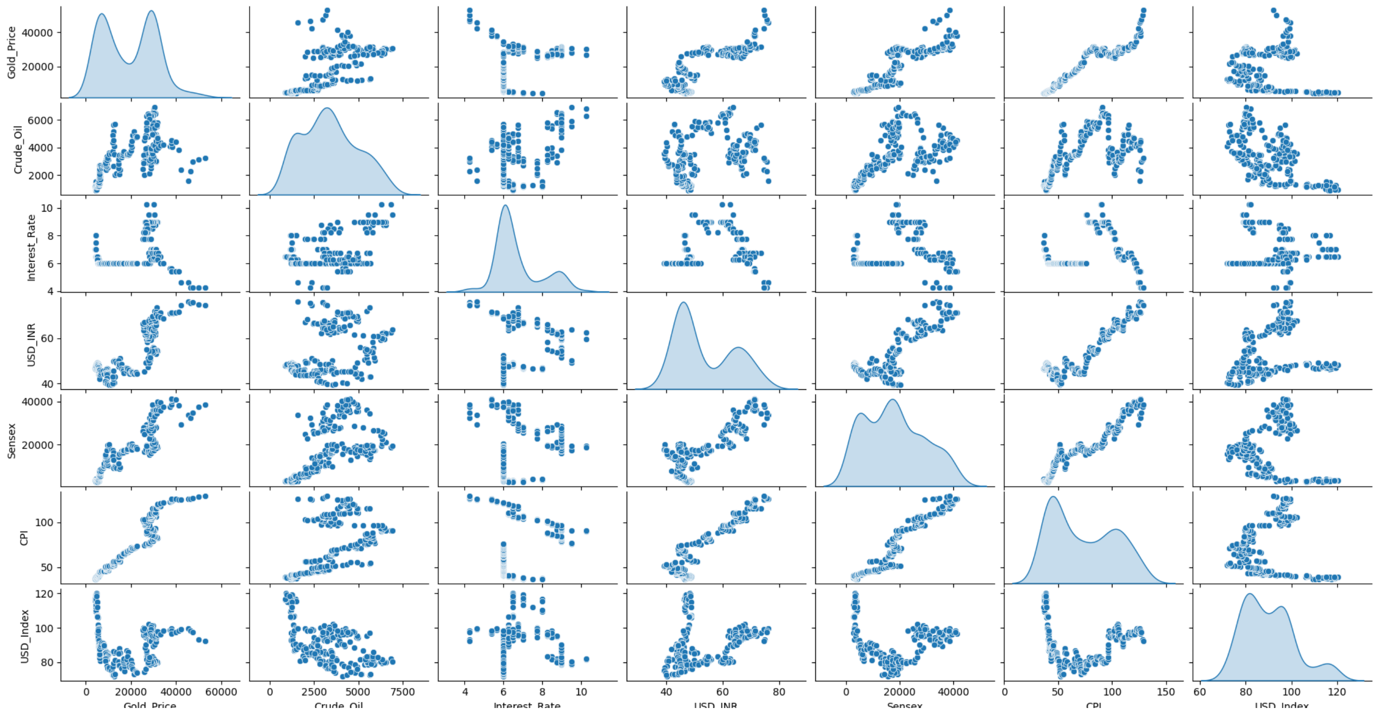


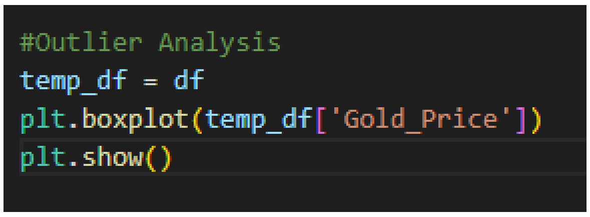
Figure 8 - Plots visualising the data

Observations:

- The data is skewed
- No normal distribution of data is observed in any of the columns

5.5. Data Preparation

The next step that was undertaken was to **check the for outliers** in the "Gold_Price" column (Figure 9). The result (Figure 10) was that there are no outliers present.



```
#Outlier Analysis
temp_df = df
plt.boxplot(temp_df['Gold_Price'])
plt.show()
```

Figure 9 - Outlier Analysis Code

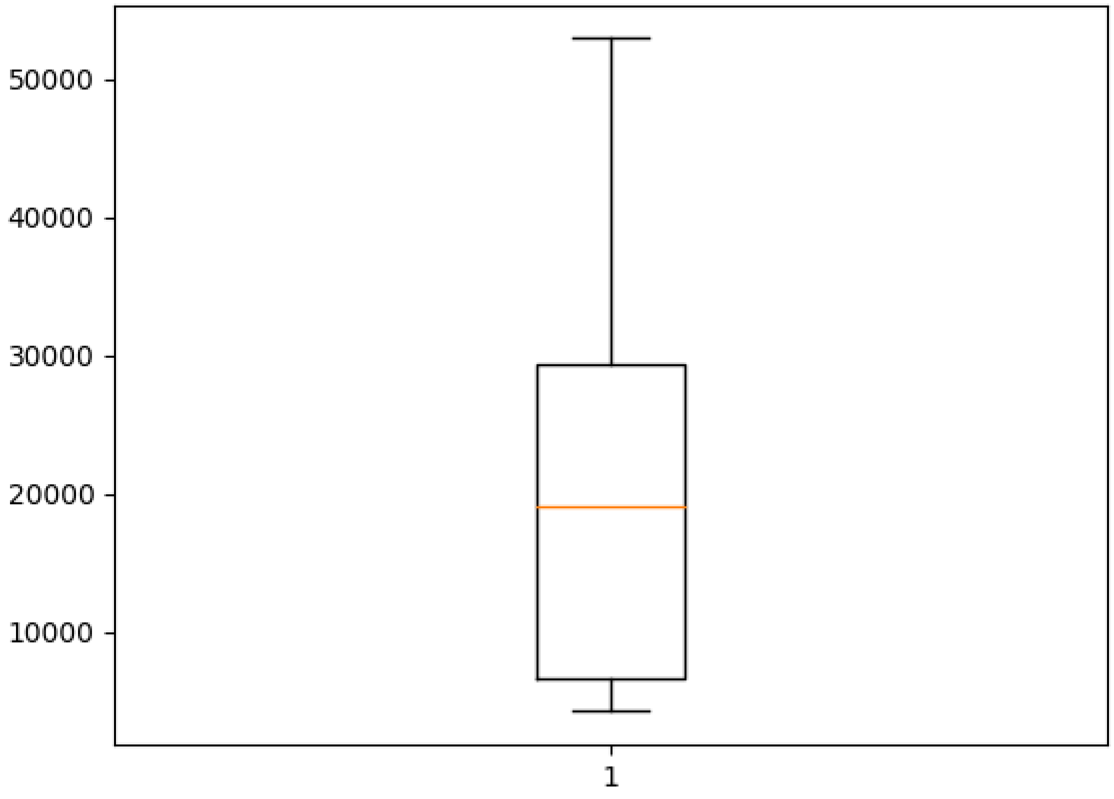


Figure 10 - Outliers result

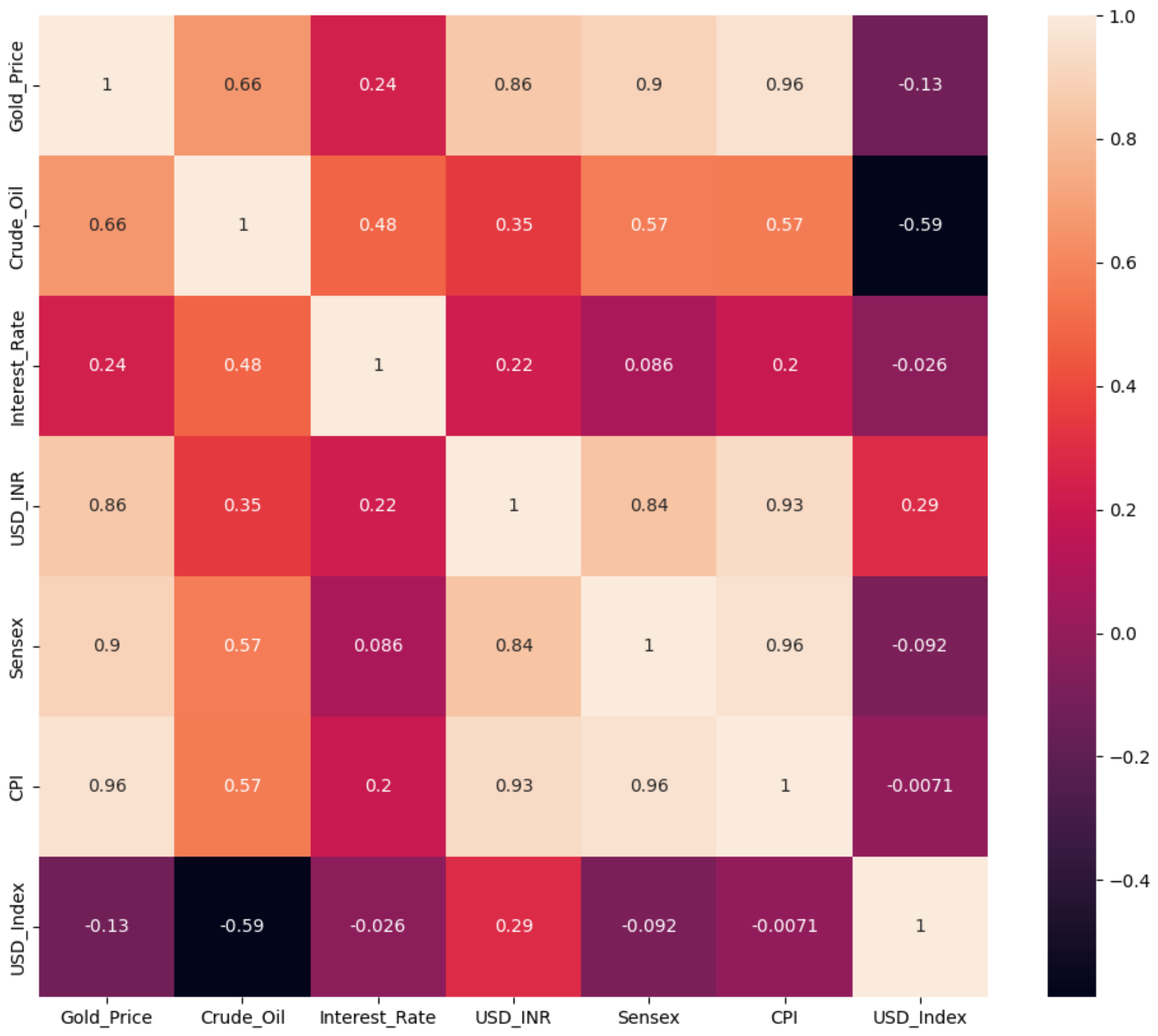
Following this step, there was the need to check for collinearity. As it is clear from Figures 11 & 12, the features that are most relevant to the price of gold are:

- CPI
- Sensex
- USD_INR

It is important to note that some multicollinearity exists between the other columns, but these would not be handled for now.

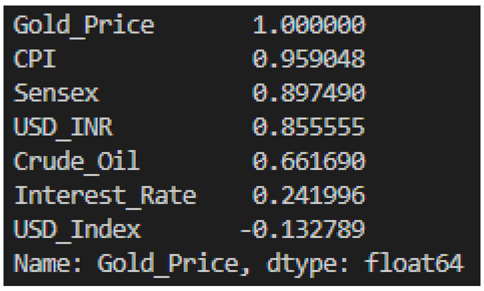
Multicollinearity is a phenomenon where two or more predictor variables in a multiple regression model are highly correlated with one another. This can be problematic as it can make it difficult to determine the unique effect of each predictor variable on the response variable. It can also lead to unstable or unreliable estimates of the regression coefficients. In simple terms, it is when two or more variables used to predict an outcome are too similar to each other(Kutner *et al.*, 2005).

Also, the names of the columns could have been changed and simplified, including bringing them all to small letters, in order to prevent from mis-types, however that does not affect the performance of the model, hence the original names were left.



	Gold_Price	Crude_Oil	Interest_Rate	USD_INR	Sensex	CPI	USD_Index
Gold_Price	1	0.66	0.24	0.86	0.9	0.96	-0.13
Crude_Oil	0.66	1	0.48	0.35	0.57	0.57	-0.59
Interest_Rate	0.24	0.48	1	0.22	0.086	0.2	-0.026
USD_INR	0.86	0.35	0.22	1	0.84	0.93	0.29
Sensex	0.9	0.57	0.086	0.84	1	0.96	-0.092
CPI	0.96	0.57	0.2	0.93	0.96	1	-0.0071
USD_Index	-0.13	-0.59	-0.026	0.29	-0.092	-0.0071	1

Figure 11 - Collinearity heatmap



```
Gold_Price        1.000000
CPI               0.959048
Sensex            0.897490
USD_INR           0.855555
Crude_Oil         0.661690
Interest_Rate     0.241996
USD_Index        -0.132789
Name: Gold_Price, dtype: float64
```

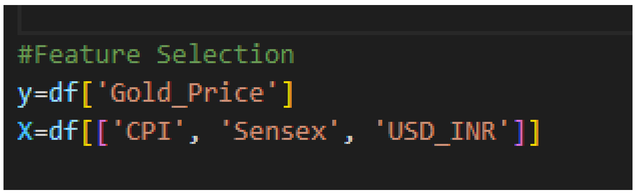
Figure 12 - Collinearity to Gold_Price

5.6. Feature Selection

After the above analysis and preparation, the features that were selected were the ones that had the highest impact on the price of gold, namely:

- CPI
- Sensex
- USD_INR

These were then selected and assigned to our independent axis of “X” and the “Gold_Price” was assigned to “y” (Figure 13).



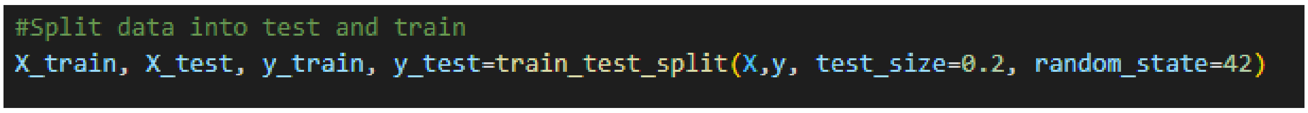
```
#Feature Selection
y=df['Gold_Price']
X=df[['CPI', 'Sensex', 'USD_INR']]
```

Figure 13 - Feature selection

6. Modelling and Results

6.1. Modelling

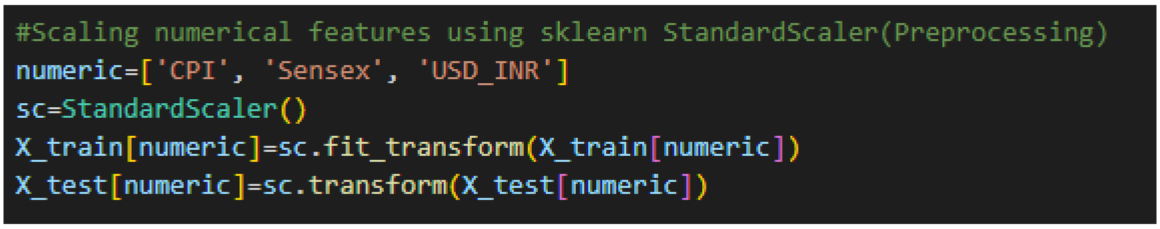
After the features were selected it was time to split it into training and testing sets (Figure 14). Each set is going to contain 20% of the whole dataset.



```
#Split data into test and train
X_train, X_test, y_train, y_test=train_test_split(X,y, test_size=0.2, random_state=42)
```

Figure 14 - Splitting the dataset

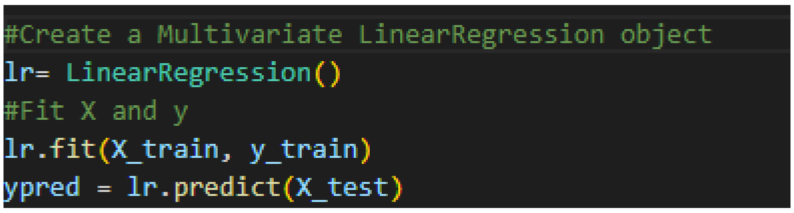
As it was mentioned earlier the data was skewed, hence the data was scaled using **standardisation of numerical values**. Scaling numerical values is important in machine learning because many algorithms, such as linear and logistic regression and neural networks, are sensitive to the scale of the input features. If the features are not scaled, certain features that have a larger scale can dominate the model and skew the results. Scaling the features to a common range, such as 0 to 1 or -1 to 1, can help to ensure that all features are given equal consideration and can improve the performance and accuracy of the model(Scikit Learn Developers, 2022b; Richert and Coelho, 2015).



```
#Scaling numerical features using sklearn StandardScaler(Preprocessing)
numeric=['CPI', 'Sensex', 'USD_INR']
sc=StandardScaler()
X_train[numeric]=sc.fit_transform(X_train[numeric])
X_test[numeric]=sc.transform(X_test[numeric])
```

Figure 15 - Scaling of the numerical values

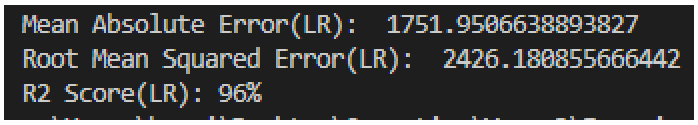
The model was then **created and the training** sets were fed to it (Figure 16).



```
#Create a Multivariate LinearRegression object
lr= LinearRegression()
#Fit X and y
lr.fit(X_train, y_train)
ypred = lr.predict(X_test)
```

Figure 16 - Creating the model and feeding the training sets.

Results proved to be quite good with the model having an r2 score of 96% and well overlapping data between the predictions and actual values (Figures 17 & 18).



```
Mean Absolute Error(LR):  1751.9506638893827
Root Mean Squared Error(LR):  2426.180855666442
R2 Score(LR): 96%
```

Figure 17 - Report of Results

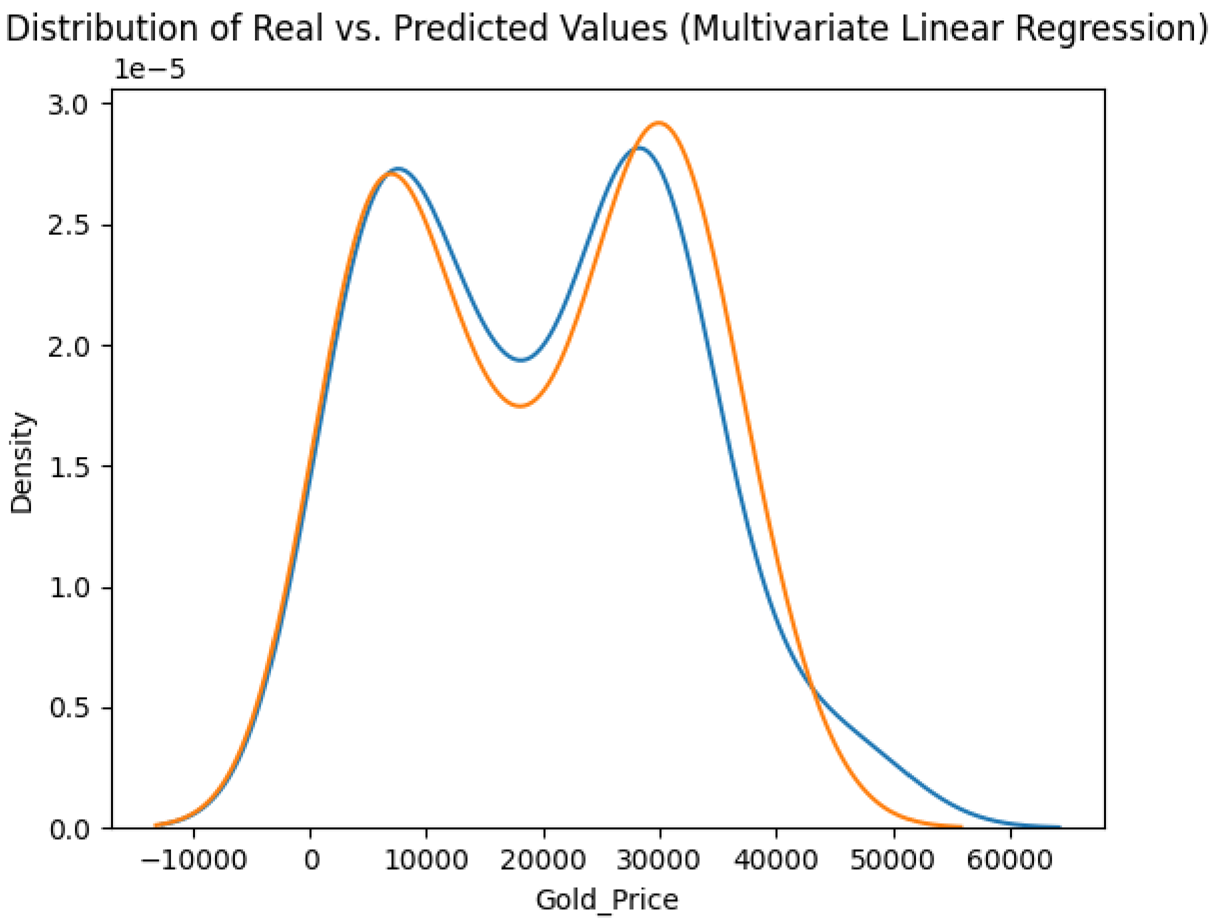
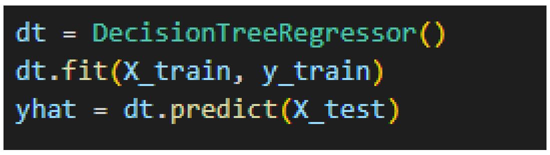


Figure 18 - Distribution of Real vs. Predicted Values (Multivariate Linear Regression)

7. Modelling and Results of a Decision Tree Model for Comparing Purposes

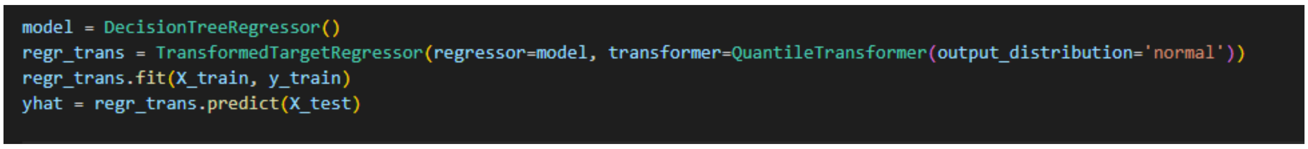
Although, the first model scored quite well, a decision was taken for a decision tree to be created for comparison purposes. This was done to see which model would prove more accurate.

7.1. Decision Tree Modelling and Training



```
dt = DecisionTreeRegressor()
dt.fit(X_train, y_train)
yhat = dt.predict(X_test)
```

Figure 19 - Creating the Decision Tree Model and fitting the data



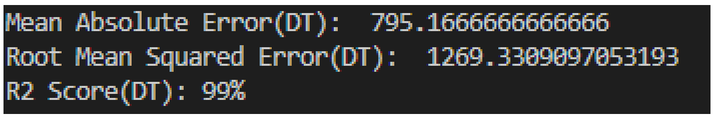
```
model = DecisionTreeRegressor()
regr_trans = TransformedTargetRegressor(regressor=model, transformer=QuantileTransformer(output_distribution='normal'))
regr_trans.fit(X_train, y_train)
yhat = regr_trans.predict(X_test)
```

Figure 20 - Engineering of the DT model

In Figure 20, it is visible that a transformer was used to improve the distribution. This method changes the features to conform to a uniform or normal distribution. As a result, for a specific feature, it tends to spread out the most frequently occurring values. Additionally, it reduces the influence of marginal outliers, making it a robust method of pre-processing. The transformation is applied to each feature independently. First, an estimate of the cumulative distribution function of a feature is used to map the original values to a uniform distribution. Then, the obtained values are mapped to the desired output distribution using the corresponding quantile function. Values of new/unseen data that fall above or below the fitted range will be mapped to the bounds of the output distribution. Note that this transformation is non-linear. It may affect linear correlations between variables measured at the same scale but makes variables measured at different scales more directly comparable(Scikit Learn Developers, 2022c, 2022a).

7.2. Results from Decision Tree Model

In Figure 21 is exhibited the performance of the Decision Tree model. It scored slightly better from our previous model with an r2 score of 99% and better overlapping of predicted vs. actual values (Figure 22).



```
Mean Absolute Error(DT):  795.1666666666666
Root Mean Squared Error(DT):  1269.3309097053193
R2 Score(DT): 99%
```

Figure 21 - Results from Decision Tree Model

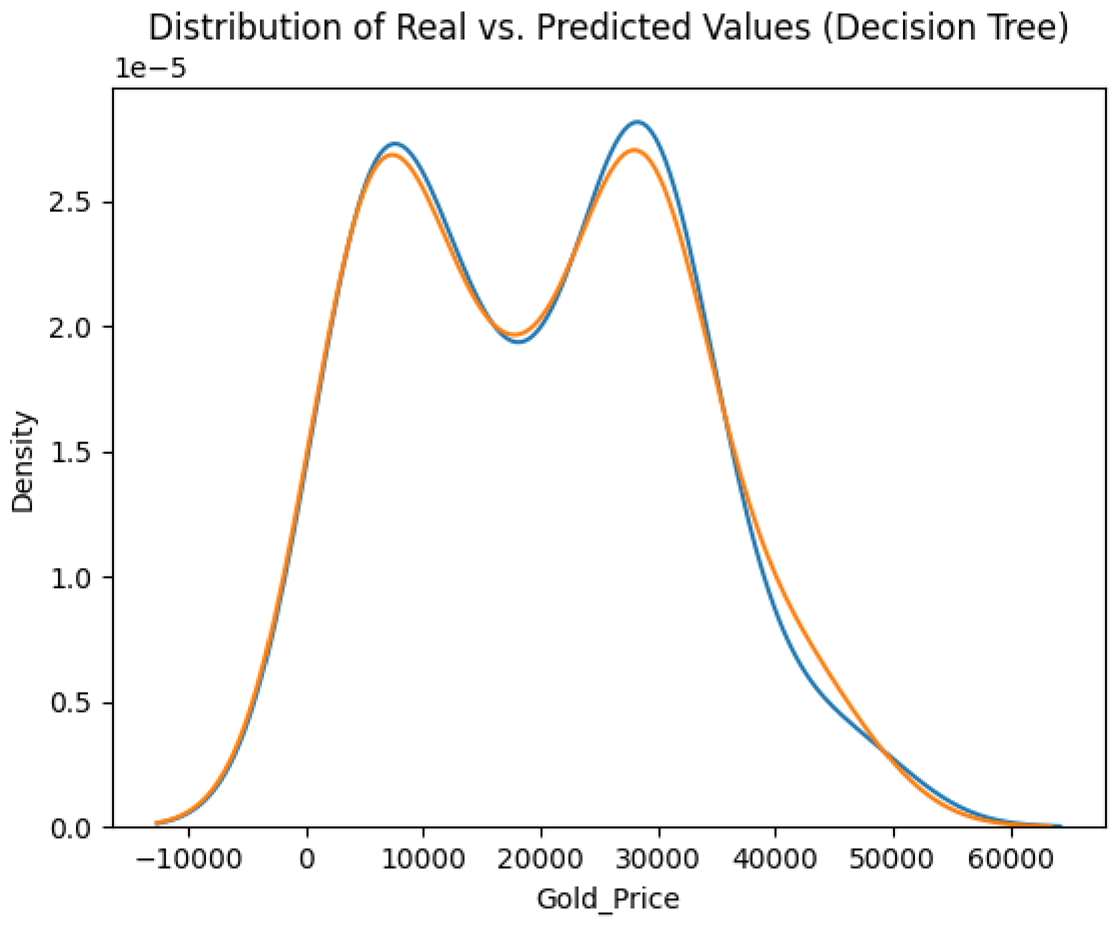


Figure 22 - Distribution of Actual vs. Predicted Values (Decision Tree)

Conclusion

This work has achieved the purpose of creating an accurate model to predict gold prices in Indian Rupees and can be implemented for production use. The comparison between the two models showed a bit better performance of the Decision Tree model over the Multivariate Linear Regression. However, the regression model can be further improved via different techniques as further data normalisation, reduce the multicollinearity further by removing some of the correlated columns, cross-validation, ensemble methods, algorithm tuning and more. The dataset in this work was not vast and problematic, but all checks were completed to ensure the data could be trained efficiently in the model.

Bibliography

Bell, J. (2014) Machine Learning: Hands-On for Developers and Technical Professionals. In: *WIley*.

Brown, S. (2021) Machine learning, explained _ MIT Sloan. *MIT Sloan*. [Accessed: 12 January 2023].

Cambridge University Press (2023) *MACHINE LEARNING | English meaning - Cambridge Dictionary*. Available at: https://dictionary.cambridge.org/dictionary/english/machine-learning [Accessed: 12 January 2023].

Kutner, M. et al. (2005) *Applied Linear Statistical Models*. 5th ed., I. [Online]. Boston, Mass.; London: McGraw-Hill. Available at: doi:10.1080/00401706.1997.10485141 [Accessed: 15 January 2023].

Mahesh, B. (2020) Machine Learning Algorithms-A Review. *International Journal of Science and Research (IJSR)*, 9(1), 381–386. [Accessed: 13 January 2023].

Ray, S. (2019) *A Quick Review of Machine Learning Algorithms*. In: Institute of Electrical and Electronics Engineers Inc., Feb 1, 2019. Institute of Electrical and Electronics Engineers Inc. Available at: doi:10.1109/COMITCon.2019.8862451

Richert, W. & Coelho, L.P. (2015) Building Machine Learning Systems with Python, Second Edition. In: *Book*. [Online]. Available at: doi:10.1007/s13398-014-0173-7.2

Scikit Learn Developers (2022a) *sklearn.preprocessing.QuantileTransformer — scikit-learn 1.2.0 documentation*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.QuantileTransformer.html [Accessed: 15 January 2023].

Scikit Learn Developers (2022b) *sklearn.preprocessing.StandardScaler — scikit-learn 1.2.0 documentation*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html [Accessed: 14 January 2023].

Scikit Learn Developers (2022c) *sklearn.tree.DecisionTreeRegressor — scikit-learn 1.2.0 documentation*. Available at: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html [Accessed: 15 January 2023].

Word Count

3004 words