Used-Car Selling Price Prediction using Machine Learning

1st Mohammad Sayeem Sadat Hossain
Department of Computer Science and Engineering
Brac University
Dhaka, Bangladesh
mohammad.sayeem.sadat.hossain@g.bracu.ac.bd

2nd Shoumya Shuprabho Rasheed
Department of Computer Science and Engineering
Brac University
Dhaka, Bangladesh
shoumya.shuprabho.rasheed@g.bracu.ac.bd

3rd Monwar Labib
Department of Computer Science and Engineering
Brac University
Dhaka, Bangladesh
monwar.labib@g.bracu.ac.bd

4th Rakibul Hasan Remon
Department of Computer Science and Engineering
Brac University
Dhaka, Bangladesh
rakibul.hasan1@g.bracu.ac.bd

***Abstract*—Cars are the automobiles which are present almost in all of our lives now. Nowadays we see that almost each family owns a car and why would not they do. Transportation has been one of the key issues which helps you move from one point of a city to another, or even to a different country. As the number of works keep increasing, the need for a stable transportation also increases, and that is when cars come into handy. However, in Bangladesh, due to the extensive tax, the selling price of a brand-new car tends to be more than double the price with which it was bought while being brought here in Bangladesh. As a result, buying a brand-new car is quite near to impossible for most families here. Hence, in order to help in arranging that huge amount of money, already bought and used-cars are ought to be sold in local car markets. An upgrade to perhaps a newer model would seem quite easy if the car is sold in the right price. Now, people feel a lot more confused as to what would the selling price be. That is when our model comes into handy as it learns from the data of already sold second-hand cars in any local car market, and hence tells what will the adequate selling price of the car in question will be. In the process, three versatile and sophisticated models were used, and finally the results were compared to jump to a final remark.**

***Index Terms*—used-car, selling price prediction, supervised learning, machine learning, regression**

I. Introduction

Cars have been the daily go-to transportation in our lives. As the number of cars being imported to Bangladesh increases, so does the number of Used cars in the market. The local car markets tend to be filled up with used cars now more often than being filled up with brand new cars. As per [1], among the total car market, 50% are reconditioned among which almost 45% are used cars, while the rest 5% are brand-new. This accounts for the fact that used cars have been growing exponentially in terms of sales in local car markets. As a result, it gets important for a personnel to know the selling price of his/her used car beforehand, so that he/she gets the maximum benefit of selling his/her car without a need to wander around the local car markets. That is when, the need for an automated predictor, based on used car data already sold, needs to be implemented. However, this has to be kept in mind that the best possible predictor is not still available as no such predictors will give a straight 100% accurate prediction. However, if a model with a minimum 95% accuracy is implemented, that would be more than enough to predict the selling price of any used car. Now, the factors which may affect the selling price of any car in general, include the year the first time the car was bought, total kilometres driven, the current brand-new version price in showroom. the transmission type, fuel type, seller type and also the previous number of owners. As a result, we used a data-set which consists of these features.

II. Methodology

A. Dataset Description and Analysis

a) Features and Labels: : This collection includes statistics about used automobiles. Many different uses of this data are possible, including price prediction to demonstrate the application of linear regression in machine learning. The provided dataset's columns are listed as follows:

- **Car_Name:** This column includes the different name of the car that are being used widely by users.
- **Year:** This column indicates the year in which the car was bought.
- **Selling_Price:** The owner's desired selling price for the vehicle should be entered in this field.
- **Present_Price:** - This is the current ex-showroom price of the car.
- **Kms_Driven:** This is the distance completed by the car in km.
- **Fuel_Type:** Type of fuel used in the car.
- **Transmission:** Indicates whether the car is manual or automatic, that is, it tells the gear type of the car.
- **Owner:** Demonstrates the number of owners the car has previously had.

The shape of the data set. using the method *df.shape()*, that we have used is (301,9), which indicates that the there are 301 rows and 9 columns. And out of these 9 columns, the label for our problem is the "Selling_Price" column.

b) Initial Data Visualization: : Using the python built-in library ***sns***, we have tried to show the counts of every categorical features of our data set. In our data set, we had three categorical features, which are *Fuel_Type*, *Seller_Type* and *Transmission*. The corresponding counts of class of each feature is shown below.

```
[6]  1 ax = sns.countplot(x="Fuel_Type", data=car)
```

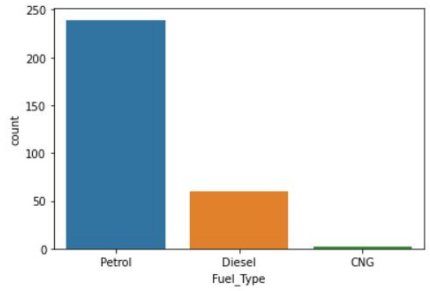

Fig. 1. Counts of Fuel Types in the data set

```
1 ax = sns.countplot(x="Seller_Type", data=car)
```

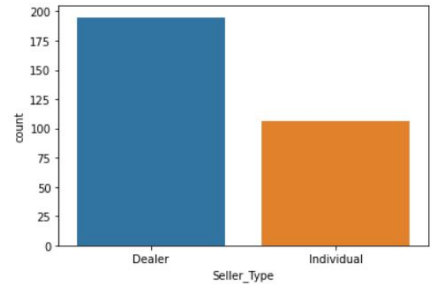

Fig. 2. Counts of Seller Types in the data set

```
1 ax = sns.countplot(x="Transmission", data=car)
```

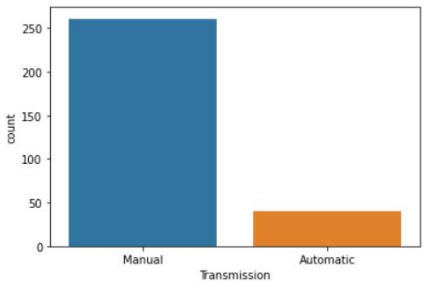

Fig. 3. Counts of Transmission Types in the data set

B. Data Pre-Processing

Data preprocessing is a crucial phase in the machine learning process since the quality of the data and the information that can be extracted from it directly influence how well our model can learn. For this reason, it is crucial that we preprocess the data before introducing it to the model.The data processing techniques that we have implemented are listed below:

- **Deleting duplicate and null values:** For better accuracy of results we need to check for any null values that are present in our data set. To do so, we use the *isnull()* function to check for any null values/missing values. In the snippet below we can see that the Selling_Price column has few null values using the *car.isnull().sum()* function.

a. Imputing Missing values

```
[ ]  #Check for missing values
     car.isnull().sum()

     Car_Name         0
     Year             0
     Selling_Price    5
     Present_Price    0
     Kms_Driven       0
     Fuel_Type        0
     Seller_Type      0
     Transmission     0
     Owner            0
     dtype: int64
```

Fig. 4. Finding the counts of null entries in our data set

- **Imputation of missing values:** To fill up the rows with null values we have used the *SimpleImputer()* function importing impute from **sklearn** library to insert/impute the mean of values from the selected column, *Selling_price* and fill up the null spaces. By doing so, we can see that there are no more missing values in the car data set.

```
#imputing missing values
from sklearn.impute import SimpleImputer

impute = SimpleImputer(missing_values=np.nan, strategy='mean')

impute.fit(car[['Selling_Price']])

car['Selling_Price'] = impute.transform(car[['Selling_Price']])
car.isnull().sum()

Car_Name         0
Year             0
Selling_Price    0
Present_Price    0
Kms_Driven       0
Fuel_Type        0
Seller_Type      0
Transmission     0
Owner            0
```

Fig. 5. Imputation and filling the null values with the mean

- **Feature Engineering:** In our data set, we had three categorical features, which include "Fuel_Type", "Seller_Type" and "Transmission." We have used the Python's built-in *map()* function to replace the corresponding classes of each feature mentioned, to numerical values. The whole code snippet is shown in Figure 6:
- **Feature Normalization/Scaling:** - We know that the features we use for predicting our label has to be scaled, since each feature may have values that range differently in comparison to each other. As a result, taking the scaled

```
i. Handling Categorical Features by Encoding

[ ]  # encoding "Fuel_Type" Column
     car['Fuel_Type'] = car['Fuel_Type'].map({'Petrol':0,'Diesel':1,'CNG':2})
     # encoding "Seller_Type" Column
     car['Seller_Type'] = car['Seller_Type'].map({'Dealer':0,'Individual':1})
     # encoding "Transmission" Column
     car['Transmission'] = car['Transmission'].map({'Automatic':0,'Manual':1})
```

Fig. 6. Handling categorical features by Encoding

X values help us to generate better accuracy. For this, we had to first split the data set into "X_train", "X_test", "y_train" and "y_test" using the *train_test_split()* function. Then, we have used the *MinMaxScaler* package under sklarn's preprocessing library, and used the *MinMaxScaler()* function to fit on our X_train data set and then to finally transform to X_train_scaled and X_test_scaled. The overall code snippet is shown below:

```
c. Feature Normalization/Scaling

+ Code  + Text

i. Train Test Splitting Data for creating scaled independent features

[ ]  X = car.drop(['Car_Name','Selling_Price'],axis=1)
     Y = car['Selling_Price']
     from sklearn.model_selection import train_test_split
     #random data splitting
     X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size = 0.25, random_state=42)

[ ]  from sklearn.preprocessing import MinMaxScaler
     scaler = MinMaxScaler()
     scaler.fit(X_train)

     # transform data
     X_train_scaled = scaler.transform(X_train)

     # transform test data
     X_test_scaled = scaler.transform(X_test)
```

Fig. 7. Feature Normalization/Scaling

- **Feature Selection:** Feature Selection is the method of reducing the input variable to any model by using only relevant data and getting rid of noise in data. It is the process of automatically choosing relevant features for our machine learning model based on the type of problem we are trying to solve. We do this by including or excluding important features without changing them. It helps in cutting down the noise in our data and reducing the size of our input data. To select such feature in our data set we have used a correlation method to see if any column has a correlation value greater than 0.75 at least twice, however we don't find any columns as such as shown in Figures 8 and 9:

C. Machine Learning Models Applied

Since in our data set we are dealing with Selling price of car which is a continuous data that falls under the category of Regression data type hence we have decided to implement the following Models suitable for Regression type of predictions. Furthermore, in each model, the steps are the same. At first we fit the model using Training Data, and then finally predict using our Testing data. The models that we used are shortly described below:

- **Linear Regression:** Linear regression analysis is used to predict the value of a variable that is based upon the value of another variable. The value to be predicted is known

```
d. Feature Selection

[ ]  car_corr = car.corr()
     car_corr
```

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
Year	1.000000	0.233810	-0.047584	-0.524342	0.053643	-0.039896	0.000394	-0.182104
Selling_Price	0.233810	1.000000	0.878933	0.032315	0.509261	-0.544225	-0.370469	-0.088033
Present_Price	-0.047584	0.878933	1.000000	0.203647	0.440415	-0.512030	-0.348715	0.008057
Kms_Driven	-0.524342	0.032315	0.203647	1.000000	0.166801	-0.101419	-0.162510	0.089216
Fuel_Type	0.053643	0.509261	0.440415	0.166801	1.000000	-0.352415	-0.080466	-0.055705
Seller_Type	-0.039896	-0.544225	-0.512030	-0.101419	-0.352415	1.000000	0.063240	0.124269
Transmission	0.000394	-0.370469	-0.348715	-0.162510	-0.080466	0.063240	1.000000	-0.050316
Owner	-0.182104	-0.088033	0.008057	0.089216	-0.055705	0.124269	-0.050316	1.000000

Fig. 8. Correlation Values

```
sns.heatmap(car_corr, cmap = 'YlGnBu')
<matplotlib.axes._subplots.AxesSubplot at 0x7f1d5b88b7d0>
```

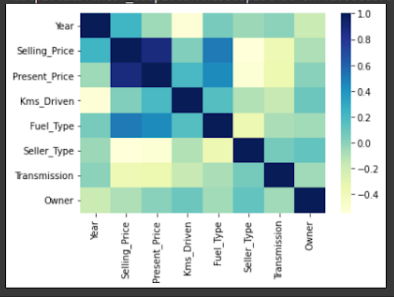

Fig. 9. Correlation Plot using Heatmap

as the dependent variable and the value used to calculate predicted value is known as independent variable [2]. Linear regression estimates the coefficient of the linear regression involving more than one independent variable that is used to determine the best predicted value of the dependent variable. This model implements a straight line of best fit that is used to minimize the difference between the predicted and actual values using any error function such as least squares [2]. Training the model and then fitting it on our test data is shown below:

```
a. Training the Model and Predicting

[38]  1 from sklearn.linear_model import LinearRegression
      2 from sklearn import metrics

[39]  1 # loading the linear regression model
      2 lin_reg_model = LinearRegression()

[40]  1 #Fitting the model on training data
      2 lin_reg_model.fit(X_train_scaled,y_train)
      3
      4 # prediction on Testing data
      5 test_data_prediction = lin_reg_model.predict(X_test_scaled)
```

Fig. 10. Linear Regression Fit and Predict

- **Decision Tree Regression:** Decision tree regression observes the current features of an object and trains it in the form of tree structure. It breaks down the data set into smaller and smaller subsets.The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. Decision tree regression is able to handle multi-output problems [3]. Decision tree is also able to deal with both categorical and numerical data. The deeper the tree goes on, the more complex the decision rules and the model

becomes more accurate. The root node, which is the best feature, is selected by different techniques, including finding the entropy or Gini index etc. Since our model also consists of many features, this model is well suited for our data set. Training the model and then fitting it on our test data is shown below:

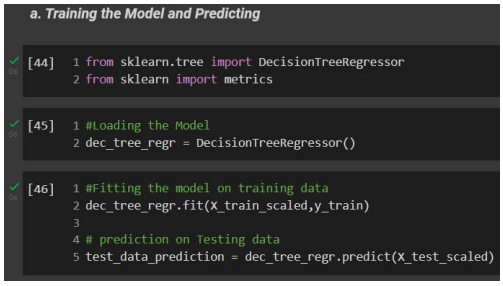

Fig. 11. Decision-Tree Regression Fit and Predict

- **Random Forest Regression:** Random forest Regressor is a meta estimator that uses a number of decision trees each dealing with a different kind of algorithm and then averages the predicted outcomes to improve the accuracy. So, the final output is the mean of all the outputs. The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees. This, as a result, is an enhanced and improved version of Decision Tree model. Training the model and then fitting it on our test data is shown below:

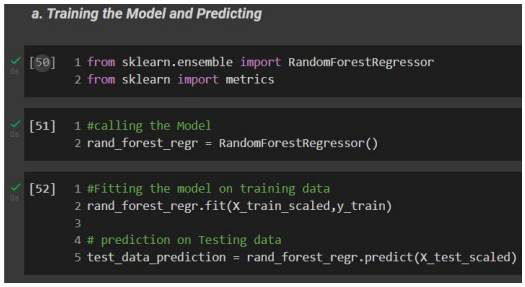

Fig. 12. Random-Forest Regression Fit and Predict

- **Gradient Boosting Regression:** Gradient boosting Algorithm helps to calculate a predictive model from weak predictive models that are typically decision tree models. Gradient boosting is a unique method that features a number of weak models and combines them together to get a better performance as a whole [5]. Gradient boosting is used for both classification and regression models. Gradient boosting model even outperforms random forest Regression [6]. Since our model is well suited for Random Forest regression, and consists of features that may not even have any connection with the label, this Machine Learning model is intended to perform even better as it takes the weak features into account. Training the model and then fitting it on our test data is shown below:

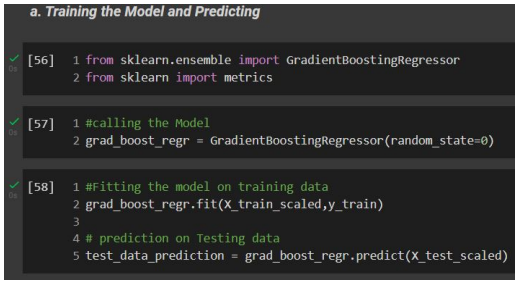

Fig. 13. Gradient-Boosting Regression Fit and Predict

III. Result and Analysis

In order to predict the price of the car, we have used different machine learning models including Linear regression, Decision tree regression, Random Forest regression and Gradient Boosting regression. To test the accuracy, we have used two metrics, suitable highly for regression type of machine learning problems. Detailed explanation with the corresponding values are shown below.

A. Detailed Metric Explanation

- **R2 Score:** R2 indicates the proportion of data point which lies within the line created by the regression equation. It is defined as "the proportion of the variance in the dependent variable that is predictable from the independent variable(s)." A higher value of R2 is desirable as it indicates better correlation between the predicted value or the label in relation to the predictors or independent variables. Formula for R2 Score is:

$$R^2 = 1 - \frac{SSR}{TSS} \tag{1}$$

where **SSR** means the "Sum of Squared Residuals" and **TSS** means the "Total Sum of Squares" respectively.

- **Mean Absolute Error, *MAE*:** MAE is simply, as the name suggests, the mean of the absolute errors. The absolute error is the absolute value of the difference between the predicted value and the actual value. So, the lesser the value of the MAE, the better the model.The formula for MAE is:

$$\sum_{i=1}^{D} |y_{pred} - y_i| \tag{2}$$

where y_{pred} values are the "Predicted Values" and y_i values are the "True Values" respectively.

B. Accuracy Scores and Visualizations

The R2 error value, MAE values and the corresponding scatter plot with the trend fitted line for our used Machine Learning models in the scatter plot are shown below:

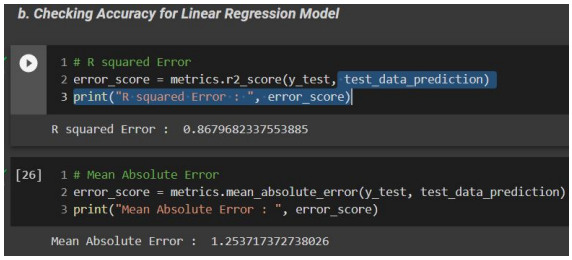

Fig. 14. Linear Regression Accuracy scores

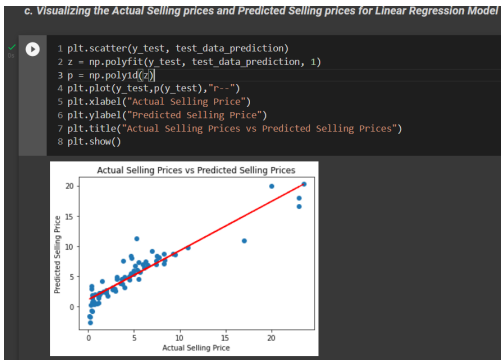

Fig. 15. Linear Regression Scatter Plot

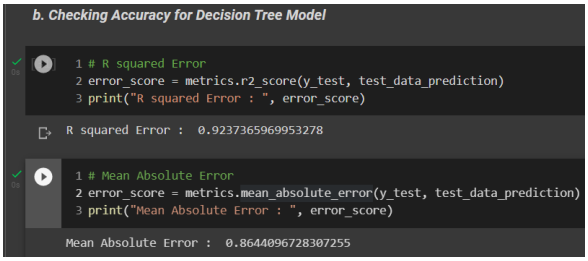

Fig. 16. Decision Tree Regression Accuracy scores

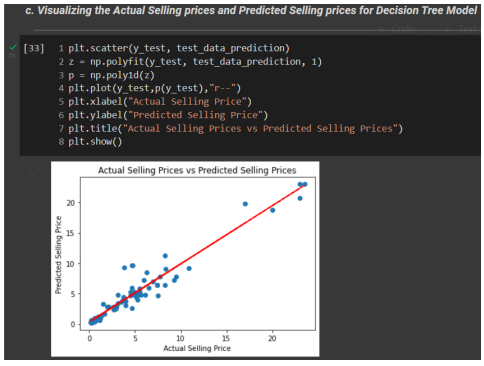

Fig. 17. Decision Tree Regression Scatter Plot

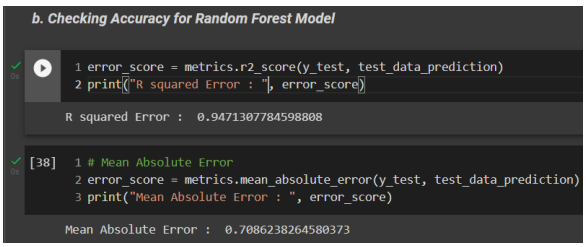

Fig. 18. Random Forest Regression Accuracy scores

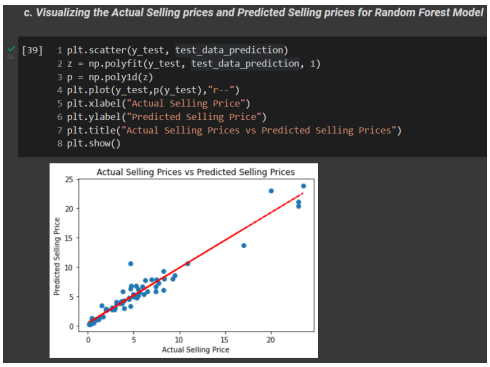

Fig. 19. Random Regression Scatter Plot

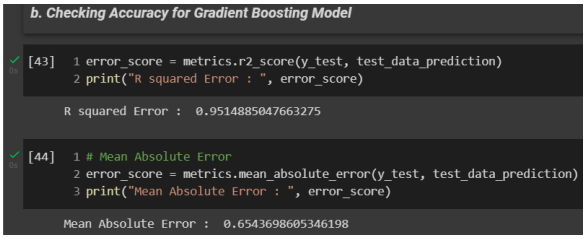

Fig. 20. Gradient Boosting Regression Accuracy scores

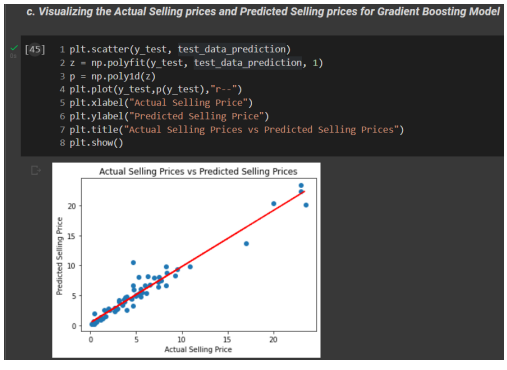

Fig. 21. Gradient Boosting Regression Scatter Plot

IV. Conclusion

From the analysis results of the accuracy scores and visualization pattern from the above section, we can surely come to conclusion that ensemble machine learning models are well suited for the data set that we have gathered from [6]. And from that analysis, we can see that an R2 score of 0.95 was achieved after using Gradient Boosting Regression model. As a result, we jump into conclusion that for predicting the sale price of any used car, the user of this model can use Gradient Boosting to fit and finally predict the desired price.

References

[1] Bikroy.com. (2018). Car Market in Bangladesh based on 2018 statistics. Bikroy. Retrieved August 25, 2018, from https://blog.bikroy.com/car-market-in-bangladesh-based-on-2018-statistics-

[2] "About linear regression," Ibm.com. [Online]. Available: https://www.ibm.com/sg-en/topics/linear-regression. [Accessed: 28-Aug-2022]

[3] "1.10. Decision Trees," scikit-learn. [Online]. Available: https://scikit-learn.org/stable/modules/tree.html. [Accessed: 28-Aug-2022].

[4] T. Masui, "All you need to know about gradient boosting algorithmpart 1. Regression," Towards Data Science, 20-Jan-2022. [Online]. Available:

[5] Wikipedia contributors, "Gradient boosting," Wikipedia, The Free Encyclopedia, 21-Jul-2022. [Online]. Available: https://en.wikipedia.org/w/index.php?

[6] N. Birla, "Vehicle dataset." 24-Oct-2020.