Summer Training Report/ Synopsis/ Minor Project

on

CAR PRICE PREDICTION USING MACHINE LEARNING

A Project Report/Synopsis submitted in partial fulfillment of the requirements for the award of

Bachelor of Engineering

IN

COMPUTER SCIENCE AND ENGINEERING

Submitted by

ABDUL RAHIM

Roll no: CO20301

AT

NATIONAL INSTITUTE OF TECHNICAL TEACHERS TRAINING AND RESEARCH (NITTTR), CHANDIGARH

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)

Government Institute under Chandigarh (UT) Administration, Affiliated to Panjab University , Chandigarh

Sector-26, Chandigarh. PIN-160019

July, 2022

Department of Computer Sc. & Engineering

CANDIDATE'S DECLARATION

I hereby declare that the work presented in this report entitled "CAR PRICE PREDICTION USING MACHINE LEARNING", in fulfillment of the requirement for the award of the degree Bachelor of Engineering in Computer Science & Engineering, submitted in CSE Department, Chandigarh College of Engineering & Technology(Degree wing) affiliated to Punjab University, Chandigarh, is an authentic record of my/our own work carried out during my degree under the guidance of Er. Amrendra Sharan The work reported in this has not been submitted by me for award of any other degree or diploma.

Date : 30 January 31, 2023 — ABDUL RAHIM

Place : CCET, Chandigarh — CO20301

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

CERTIFICATE

राष्ट्रीय तकनीकी शिक्षक प्रशिक्षण एवं अनुसंधान संस्थान

National Institute of Technical Teachers Training and Research

Ministry of Education, Government of India / शिक्षा मंत्रालय, भारत सरकार

Sector-26, Chandigarh-160019 (India) | ISO 9001:2015 Certified

तकनीकी क्षमता विकास केंद्र

Centre for Development of Technical Competencies (CDTC)

NITTTR/CDTC/2022-23/CSE/48

12/08/2022

Certificate

Certified that ***ABDUL RAHIM*** student of *B.E (CSE)*, CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY, *CHANDIGARH* has attended 5 Weeks Industrial Training on ***"Data Science (a Project based learning)"*** conducted by Computer Science & Engineering Department, NITTTR Chandigarh from **4/7/2022** to **5/8/2022**. He/She has successfully undergone the training programme.

Coordinator
(Er. Amrendra Sharan)

Chairperson, CDTC
(Dr. Meenakshi Sood)

Head of the Department
(Dr. C. Rama Krishna)

CHANDIGARH COLLEGE OF ENGINEERING AND TECHNOLOGY (DEGREE WING)
Government Institute under Chandigarh (UT) Administration | Affiliated to Panjab University , Chandigarh
Sector-26, Chandigarh. PIN-160019 | Tel. No. 0172-2750947, 2750943
Website: www.ccet.ac.in | Email: principal@ccet.ac.in | Fax. No. :0172-2750872

Department of Computer Sc. & Engineering

ACKNOWLEDGEMENT

It is my proud privilege and duty to acknowledge the help and guidance received from several people in preparing this report. It would not have been possible to prepare this report in this form without their valuable help, cooperation, and guidance.

I feel deeply honoured in expressing my sincere thanks to “NITTTR, Chandigarh” for giving me this opportunity and guiding me throughout my internship to get a good understanding of data science and valuable insights leading to matured clarity in the industry.

I wish to record my sincere gratitude to Faculty members, Mentors, and Academic cells for their constant support and encouragement in the preparation of this report.

The guidance on internship opportunities were very helpful us in giving the necessary background information and inspiration. Their contributions and technical support in preparing this report are greatly acknowledged.

Last but not the least, we wish to thank our parents for financing our studies in this college as well as for constantly encouraging us to learn to engineer. Their personal sacrifice in providing this opportunity to learn engineering is gratefully acknowledged.

Department of Computer Sc. & Engineering

ABSTRACT

Predicting used car prices is important for various industries and individuals for making informed decisions about buying, selling, and owning cars. "Car Price Prediction using Machine Learning" is a project aimed at accurately predicting the price of a car using machine learning algorithms. The project uses four commonly used algorithms: Linear Regression, Polynomial Regression, Decision Tree Regression, and Random Forest Regression. These algorithms are applied to a dataset and the performance of each algorithm is compared to determine the most accurate model.

The project also implements a user interface using the Pygame module in Python, allowing users to interact with the model and obtain price predictions. The project aims to provide a reliable solution for predicting car prices using machine learning and to demonstrate the effectiveness of different regression algorithms.

This project has the potential to positively impact various industries and individuals by providing reliable and accurate used car price predictions.

TABLE OF CONTENTS

INTRODUCTION

- Estimating price of used car’s is a big problem. Even experts may give varying estimates for a given vehicle. This sometimes leads to exploitation of uninformed consumers.
- Usually people are unaware of market price of their used vehicles and they end up selling it at a very low rate.
- On the contrary an unaware buyer may buy a used vehicle at a ridiculously high price.
- Predicting used vehicle price is also beneficial for companies which sell new vehicles at exchange offers (exchanging old vehicles for new), buyers also benefit, since they get fair price for their used vehicle.
- Car price prediction algorithms attempt to predict price of used cars, such algorithms can be implemented in online shopping websites, 2nd hand online markets, and can also be used by dealers to maximise their profit.

The project aims to develop a predictive model to estimate the selling price of used cars based on various factors such as new price, year, kilometres driven and other attributes.

In today's rapidly growing automotive market, used cars play a crucial role in providing affordable transportation options to many consumers. However, determining the fair market value of a used car can be a challenging task, especially for inexperienced buyers and sellers.

This project aims to provide an accurate and reliable solution for estimating used car prices. By using machine learning algorithms, the project will analyse large amounts of data on used cars and develop a model that can predict the selling price of a used car based on its attributes.

The results of this project will provide valuable insights for buyers and sellers in the used car market and help them make informed decisions about the buying and selling of used cars. The project will also contribute to the advancement of machine learning techniques for price prediction in the automotive industry.

PROBLEM DEFINATION

The problem is to predict price of used car from its attributes such as Kms driven, Year, Fuel Type etc. So the flowchart for the basic functioning of the program is as follows:

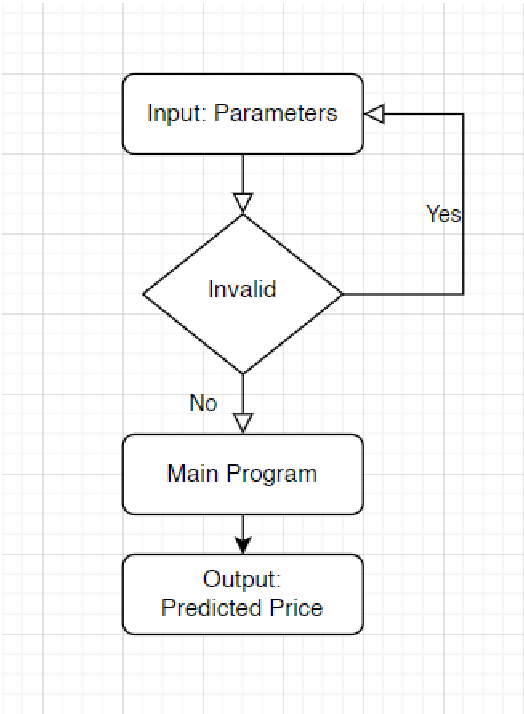

Fig. 1 Flowchart of the basic working of the program

Now that we have defined the problem, lets think of how we can implement it. Since, we are to predict price of a car, which is a continuous float value, hence it is a regression problem.

Regression is a common problem in machine learning where the goal is to predict a continuous dependent variable based on one or more independent variables. It is used to model the relationship between the dependent variable and the independent variables, with the goal of making predictions about the dependent variable based on new or unseen data. To solve regression problems various algorithms are used such as linear regression, decision tree regression, polynomial regression etc.

The workflow for typical Regression Problem looks as follows:

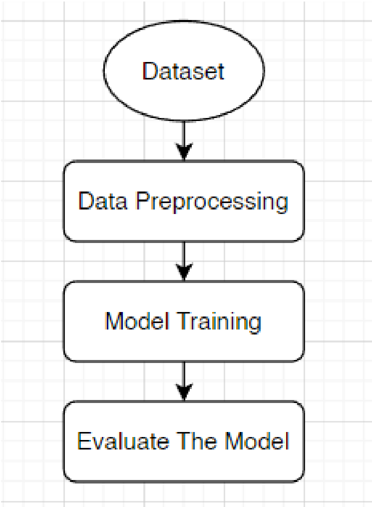

Fig. 2 Workflow of Regression problem

DATA PREPROCESSING

The data preprocessing is the most important step in training any machine learning model. Data preprocessing refers to the steps taken to prepare and transform raw data into a format that is suitable for analysis and modelling. The main goal of data preprocessing is to make the data suitable for use in machine learning algorithms.

The steps involved in data preprocessing typically include:

- Data Collection: Gathering data from various sources such as databases, files, and web scraping.
- Data Cleaning: Identifying and removing errors, inconsistencies, and inaccuracies in the data.
- Data Transformation: For example, converting categorical variables into numerical values through one-hot encoding.
- Data Normalization: Scaling the data to a standard scale to ensure that the magnitude of the variables does not affect the model's results.
- Data Reduction: Reducing the dimensionality of the data by removing irrelevant or redundant features to improve the model's performance.
- Data Partitioning: Dividing the data into training and testing sets to evaluate the model's performance.

MODEL TRAINING

Model training typically includes:

- Select the model: Select the appropriate regression model based on the data and the problem. This could include linear regression, polynomial regression, decision tree regression, or random forest regression, among others.
- Train the model: Train the selected model on the training data.

EVALUATE THE MODEL

Evaluate the performance of the model on the test data using metrics such as mean squared error, mean absolute error, or R-squared.

After we have trained our model we need some mechanism by which we can input values/attributes in our model and it outputs the prediction. This functionality is provided by a python module called Pickle. We also need to make a user interface by which we can have a user input attribute values, and then the result is displayed on the interface, for this we are going to use a python module called Pygame.

DATA PREPROCESSING

DATASET

To implement a machine learning algorithm, we first need a dataset, The model is then trained on the dataset, and then the trained model can be used for prediction.

For this problem we have used our dataset from Kaggle called 'car price.csv'.

	A	B	C	D	E	F	G	H	I	J
1	Car_Name	Year	Selling_Price	New_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	
2	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	
3	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	
4	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	
5	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	
6	swift	2014	4.6	6.87	42450	Diesel	Dealer	Manual	0	
7	vitara brezza	2018	9.25	9.83	2071	Diesel	Dealer	Manual	0	
8	ciaz	2015	6.75	8.12	18796	Petrol	Dealer	Manual	0	
9	s cross	2015	6.5	8.61	33429	Diesel	Dealer	Manual	0	
10	ciaz	2016	8.75	8.89	20273	Diesel	Dealer	Manual	0	
11	ciaz	2015	7.45	8.92	42367	Diesel	Dealer	Manual	0	
12	alto 800	2017	2.85	3.6	2135	Petrol	Dealer	Manual	0	
13	ciaz	2015	6.85	10.38	51000	Diesel	Dealer	Manual	0	
14	ciaz	2015	7.5	9.94	15000	Petrol	Dealer	Automatic	0	
15	ertiga	2015	6.1	7.71	26000	Petrol	Dealer	Manual	0	
16	dzire	2009	2.25	7.21	77427	Petrol	Dealer	Manual	0	

Fig. 2: Dataset

DATA CLEANING

Data Cleaning is the process of identifying and correcting or removing errors, inaccuracies, and inconsistencies in a dataset. For this we need to check the datatype of attributes in our data and what range of values the variables can take.

The function car_data.info() returns information about datatype of attributes in our system. From this we can infer that the all attributes are in the required format.

```
In [4]: car_data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 301 entries, 0 to 300
Data columns (total 9 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   Car_Name       301 non-null    object
 1   Year           301 non-null    int64
 2   Selling_Price  301 non-null    float64
 3   New_Price      301 non-null    float64
 4   Kms_Driven     301 non-null    int64
 5   Fuel_Type      301 non-null    object
 6   Seller_Type    301 non-null    object
 7   Transmission   301 non-null    object
 8   Owner          301 non-null    int64
dtypes: float64(2), int64(3), object(4)
memory usage: 21.3+ KB
```

Fig. 3: Data type of attributes

HANDLING MISSING VALUES

Now we check for missing values in our data. The function car_data.isnull().sum() returns the sum of null values in each attribute.

```
In [5]: # this gives the sum of all null values in our d
        car_data.isnull().sum()

Out[5]: Car_Name         0
        Year             0
        Selling_Price    0
        New_Price        0
        Kms_Driven       0
        Fuel_Type        0
        Seller_Type      0
        Transmission     0
        Owner            0
        dtype: int64
```

Fig. 3: Sum of null values in each attribute

It is clear that there are no missing values in our data, hence we can safely proceed further.

OUTLIER DETECTION

To eliminate outliers in our data, we need statistical information about the data. The function car_data_describe() returns statistical information about our data.

```
In [6]: # this stastical information about our data
        car_data.describe()

Out[6]:
```

	Year	Selling_Price	New_Price	Kms_Driven	Owner
count	301.000000	301.000000	301.000000	301.000000	301.000000
mean	2013.627907	4.661296	7.628472	36947.205980	0.043189
std	2.891554	5.082812	8.644115	38886.883882	0.247915
min	2003.000000	0.100000	0.320000	500.000000	0.000000
25%	2012.000000	0.900000	1.200000	15000.000000	0.000000
50%	2014.000000	3.600000	6.400000	32000.000000	0.000000
75%	2016.000000	6.000000	9.900000	48767.000000	0.000000
max	2018.000000	35.000000	92.600000	500000.000000	3.000000

Fig.4 Statistical information about the data

From this table we can infer that all maximum and minimum values seem to be reasonable, hence we can conclude that there are no outliers in our data.

HANDLING INCONSISTENT DATA

To check for inconsistent data, we find what range of values can be present in categorical variables like Car_Name, , Fuel_Type, Seller_Type, Transmission.

```
In [8]: # # there are 98 type of cars in our data.
        print(car_data['Car_Name'].value_counts())

city                         26
corolla altis                16
verna                        14
fortuner                     11
brio                         10
                             ..
Honda CB Trigger              1
Yamaha FZ S                   1
Bajaj Pulsar 135 LS           1
Activa 4g                     1
Bajaj Avenger Street 220      1
Name: Car_Name, Length: 98, dtype: int64
```

Fig. Range of values Car_Name can take

```
In [10]: # there are 3 possible values in Fuel_type variable
         print(car_data['Fuel_Type'].value_counts())

Petrol    239
Diesel     60
CNG         2
Name: Fuel_Type, dtype: int64

In [11]: # there are 2 possible values in Seller_type variable
         print(car_data['Seller_Type'].value_counts())

Dealer        195
Individual    106
Name: Seller_Type, dtype: int64

In [12]: # there are 2 possible values in Transmission variable
         print(car_data['Transmission'].value_counts())

Manual       261
Automatic     40
Name: Transmission, dtype: int64
```

Fig. Range of values tuples can take for these attributes

From this we can conclude that data is in consistent format in our dataset.

DATA EXPLORATION

In this section we will try to understand the relationship between variables, in our data. First lets visualise how Fuel Type, Seller Type, Transmission are related to the selling price of the car.

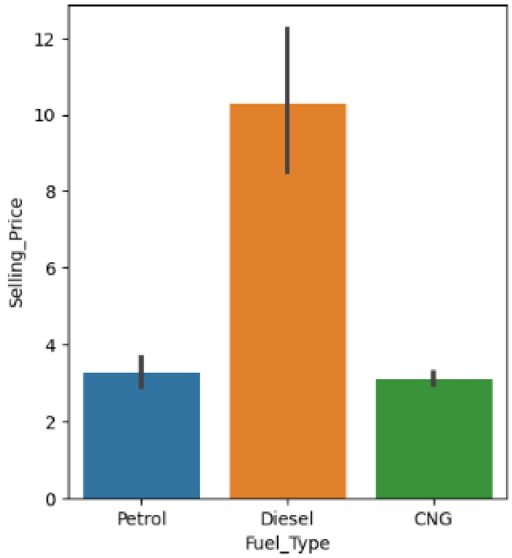

Fig. Visualising relationship between Fuel Type and selling price

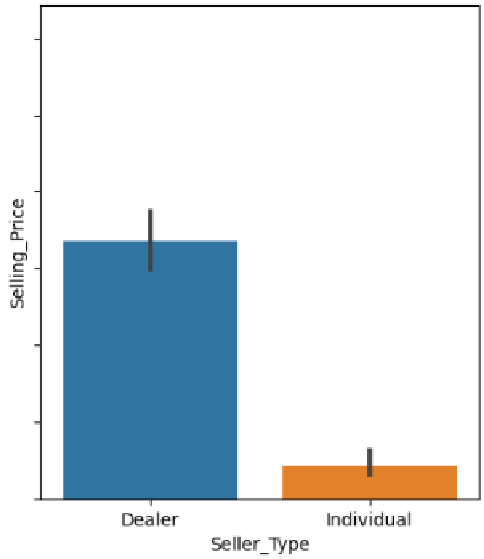

Fig. Visualising relationship between Seller Type and selling price

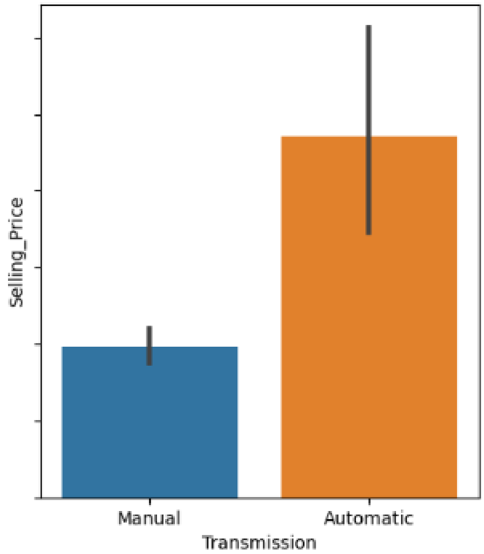

Fig. Visualising relationship between Transmission and selling price

From the above plots we can infer that there is some corelation between attributes fuel type, seller type, transmission. The price of Diesel vehicles seem to be higher than petrol and CNG, The price of vehicles sold by dealer seem to be higher that vehicles sold by individuals, and automatic transmission vehicles seem to be higher priced than manual transmission vehicles. Hence we can say that these attributes are valuable to our analysis.

DATA TRANSFORMATION

Converting categorical variables into numerical values is called data transformation. We have 3 categorical variables in our dataset, that is: fuel type, seller type, transmission.

encoding categorical data

```
In [ ]:

In [15]: # manual encoding
car_data.replace({'Fuel_Type':{'Petrol':0, 'Diesel':1, 'CNG':2}}, inplace=True)

#one hot encoding
car_data = pd.get_dummies(car_data, columns=['Seller_Type', 'Transmission'], drop_first=True)

In [16]: car_data.head()
Out[16]:
```

	Car_Name	Year	Selling_Price	New_Price	Kms_Driven	Fuel_Type	Owner	Seller_Type_Individual	Transmission_Manual
0	ritz	2014	3.35	5.59	27000	0	0	0	1
1	sx4	2013	4.75	9.54	43000	1	0	0	1
2	ciaz	2017	7.25	9.85	6900	0	0	0	1
3	wagon r	2011	2.85	4.15	5200	0	0	0	1
4	swift	2014	4.60	6.87	42450	1	0	0	1

Fig. Encoding categorical variables

We have used one hot encoding for variables seller type, transmission. And manual encoding for variable fuel type.

DIMENTIONALITY REDUCTION

Reducing the dimensionality of the data by removing irrelevant or redundant features to improve the model's performance. This can be done by finding correlation between attributes by plotting a heatmap.

It was mentioned previously that the attribute car name has 98 type of values,

```
In [8]: # # there are 98 type of cars in our data.
        print(car_data['Car_Name'].value_counts())

city                         26
corolla altis                16
verna                        14
fortuner                     11
brio                         10
                             ..
Honda CB Trigger              1
Yamaha FZ S                   1
Bajaj Pulsar 135 LS           1
Activa 4g                     1
Bajaj Avenger Street 220      1
Name: Car_Name, Length: 98, dtype: int64
```

Fig. range of values for car name

Now let's assume that it would have been fewer, then it is reasonable to think that the model would have found a pattern and conclude that the resale value of certain cars is more than others. Let's say Maruti Suzuki has more resail value than Ford fiat. But the problem is that there are so many type of cars in the dataset that such a pattern cannot be generated, hence we conclude that the attribute car name is a redundant feature.

HEATMAP

Now we will plot a heat map to find out the correlation between variables. Generally we want the correlation between variables to be minimum. If two variables are highly correlated(say more than 90%) then one of them needs to be removed for the model to be accurate.

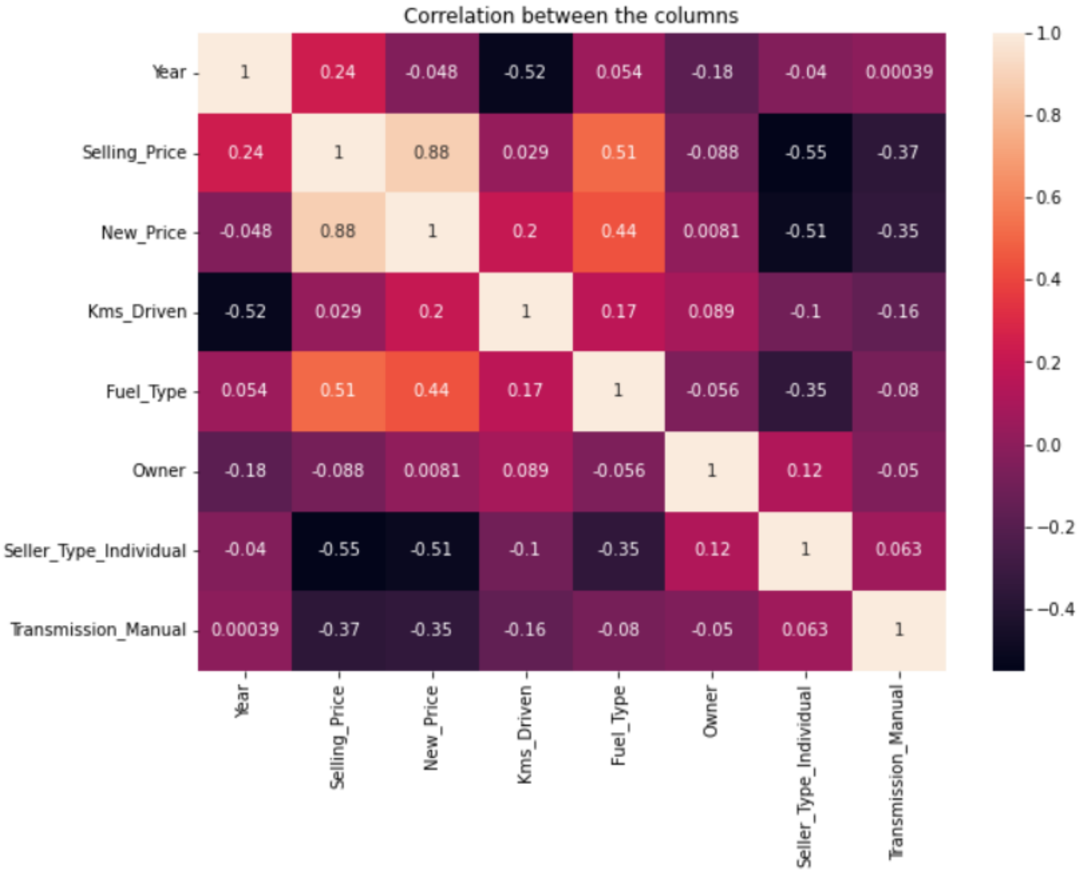

Fig. Heatmap

From this heatmap we can conclude that there are no highly correlated values in our dataset hence we can proceed further.

TRAIN TEST SPLIT

One part of the data goes to train the model and the other part goes to test the model, so we need to split our data set into training and testing.

```
In [20]: # splitting the data into 70% training and 30% testing
         X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.3, random_state=42)

         print("X_test shape:", X_test.shape)
         print("X_train shape:", X_train.shape)
         print("y_test shape: ", y_test.shape)
         print("y_train shape:", y_train.shape)

         X_test shape: (91, 7)
         X_train shape: (210, 7)
         y_test shape:  (91,)
         y_train shape: (210,)
```

Fig. Splitting data into training and testing set

DATA NORMALISATION

Scaling the data to a standard scale to ensure that the magnitude of the variables does not affect the model's results is called data normalisation.

```
In [22]: # now we normalise our data using standard scaler
         scaler = StandardScaler()

         X_train = scaler.fit_transform(X_train)
         X_test = scaler.transform(X_test)
```

Fig. Normalising the data using standard scaler

We normalise out data using standard scaler.

MODEL TRAINING AND EVALUATION

Model training and evaluation is the process of training the model on training set, and testing it on test set using various metrics such as Mean Absolute Error(MAE), Mean Squared Error(MSE), R2 score.

Training the model is a simple and straightforward process in python.

LINEAR REGRESSION

```
In [20]: # implementing Linear Regression Model
         model = LinearRegression()

         model.fit(X_train, y_train)

         pred = model.predict(X_test)
```

```
In [21]: print('Linear Regression:')
         print("Mean Absolute Error: ", (metrics.mean_absolute_error(pred, y_test)))
         print("Mean Squared Error: ", (metrics.mean_squared_error(pred, y_test)))
         print("R2 score: ", (metrics.r2_score(pred, y_test)))
```

```
Linear Regression:
Mean Absolute Error:  1.2581404706473374
Mean Squared Error:  3.4932860262251473
R2 score:  0.8294933369778816
```

Fig. Training Linear Regression Model

Regression plots as the name suggests creates a regression line between 2 parameters and helps to visualize their linear relationships.

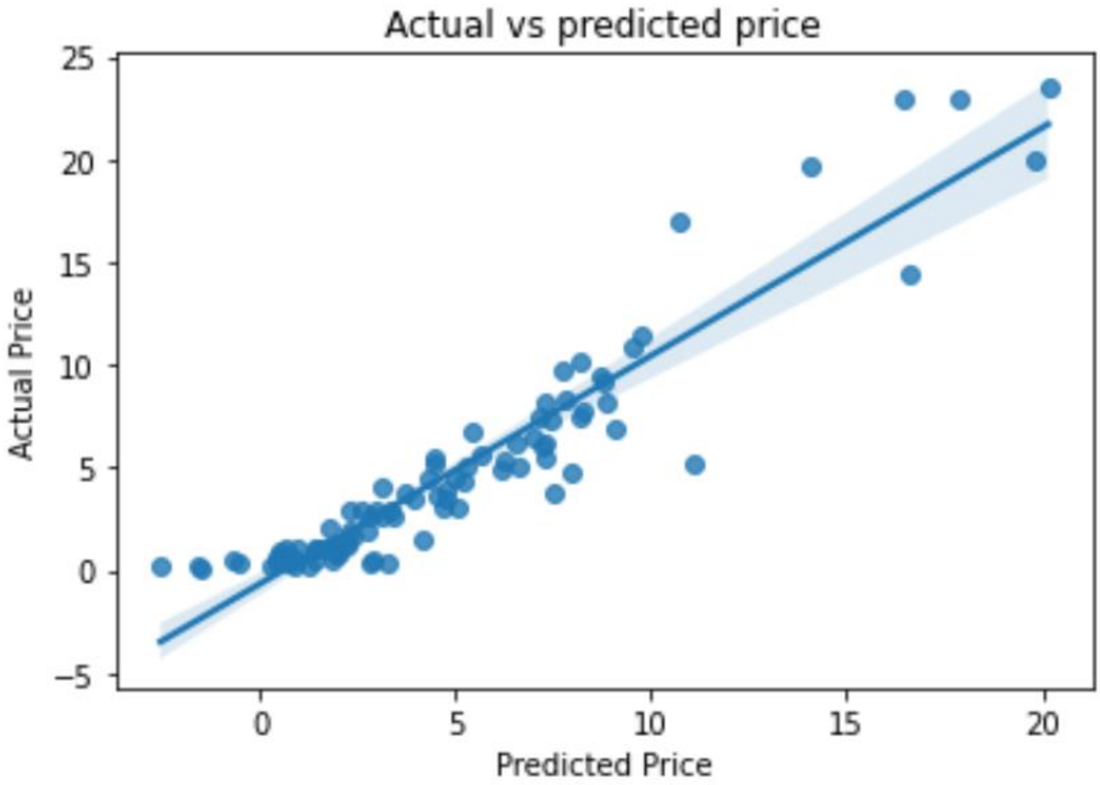

Fig. Regression Plot

POLYNOMIAL REGRESSION

Now lets implement polynomial regression

In [26]:
```python
# Fitting Polynomial Regression to the dataset
from sklearn.preprocessing import PolynomialFeatures

poly_reg = PolynomialFeatures(degree = 2)
X_poly = poly_reg.fit_transform(X_train)
X_test_poly = poly_reg.fit_transform(X_test)
polynomialRegression = LinearRegression()
polynomialRegression.fit(X_poly , y_train)
```

Out[26]:
```
LinearRegression()
```

In [27]:
```python
y_pred_poly = polynomialRegression.predict(X_test_poly)
```

In [28]:
```python
print('Polynomial Regression:')
print("Mean Absolute Error: ", (metrics.mean_absolute_error(y_pred_poly, y_test)))
print("Mean Squared Error: ", (metrics.mean_squared_error(y_pred_poly, y_test)))
print("R2 score: ", (metrics.r2_score(y_pred_poly, y_test)))
```

```
Polynomial Regression:
Mean Absolute Error:  0.8757172905219781
Mean Squared Error:  1.8301131081885034
R2 score:  0.9386259739027897
```

Fig. Polynomial Regression

DECISION TREE REGRESSION

```
In [ ]: # now lets implement Decision Tree Regression
```

```
In [24]: DecisionTree = DecisionTreeRegressor(random_state = 0)
         DecisionTree.fit(X_train, y_train)
         y_dtpred = DecisionTree.predict(X_test)
```

```
In [25]: # results
         print('Decision Tree Regression:')
         print("Mean Absolute Error: ", (metrics.mean_absolute_error(y_dtpred, y_test)))
         print("Mean Squared Error: ", (metrics.mean_squared_error(y_dtpred, y_test)))
         print("R2 score: ", (metrics.r2_score(y_dtpred, y_test)))

         Decision Tree Regression:
         Mean Absolute Error:  0.7323076923076924
         Mean Squared Error:  1.5089692307692306
         R2 score:  0.9404393781436885
```

Fig. Decision Tree Regression

RANDOM FOREST REGRESSION

```
In [ ]: # Now lets implement Random Forest Regression
```

```
In [26]: randomForest = RandomForestRegressor(n_estimators = 100)
         randomForest.fit(X_train, y_train)
         y_pred = randomForest.predict(X_test)
```

```
In [27]: # results
         print('Random Forest Regression:')
         print("Mean Absolute Error: ", (metrics.mean_absolute_error(y_pred, y_test)))
         print("Mean Squared Error: ", (metrics.mean_squared_error(y_pred, y_test)))
         print("R2 score: ", (metrics.r2_score(y_pred, y_test)))

         Random Forest Regression:
         Mean Absolute Error:  0.6191890109890111
         Mean Squared Error:  1.0457069668131869
         R2 score:  0.9593753321547437
```

```
In [ ]:
```

Fig. Random Forest Regression

COMPARING THE ALGORITHMS

Now we will plot the MSE, MAE, and R2 score of each algorithm to find out which algorithm performed the best. It is clear that **Random Forest performs the best in all metrics** inferring form the plots below.

- **R2 score:** The proportion of the variance in the dependent variable that is predictable from the independent variable(s). R2 score varies between 0 and 1, algorithm with **higher R2 score is better**.

- **Mean Absolute Error (MAE):** The magnitude of difference between the prediction of an observation and the true value of that observation. Algorithm with **lower MAE is better**.

- **Mean Squared Error (MSE):** Mean square error (MSE) is the average of the square of the errors. The larger the number the larger the error. Algorithm with **lower MSE is better**.

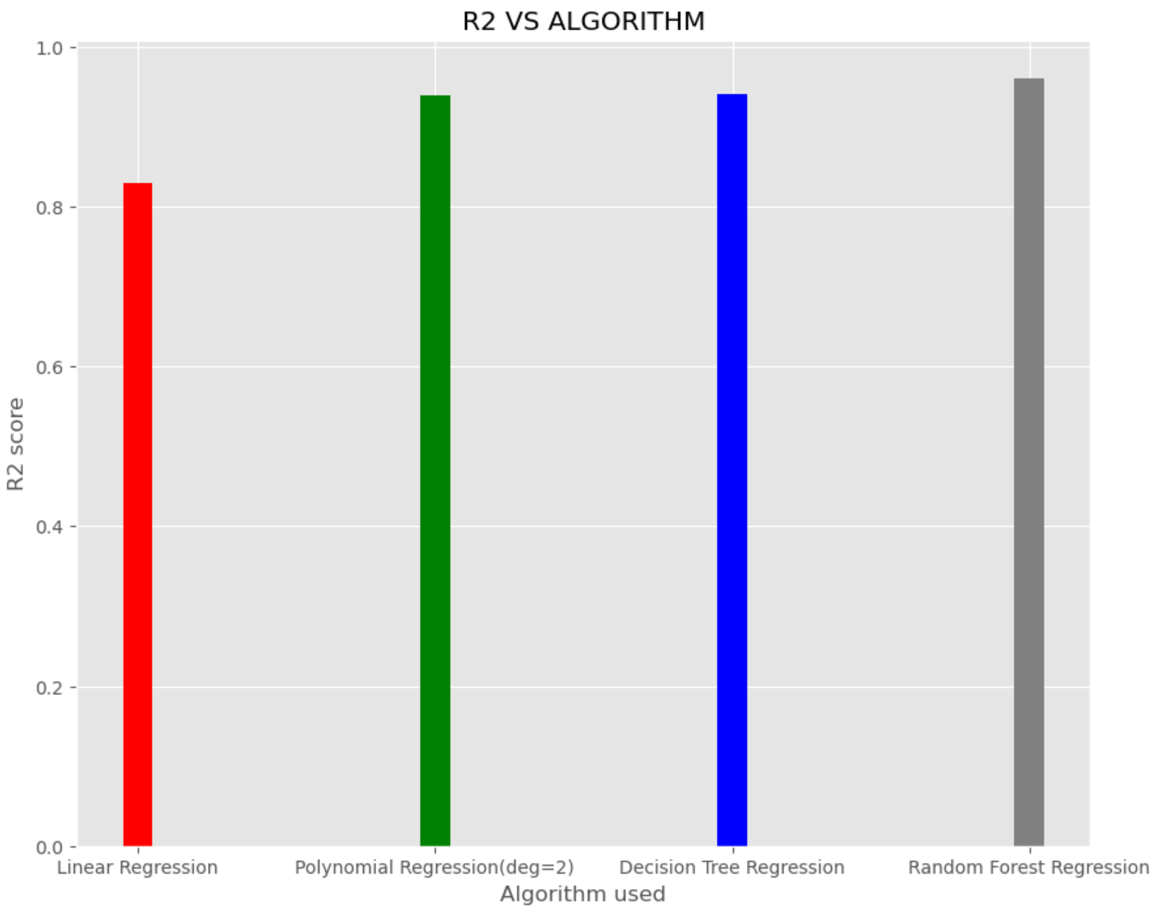

Fig. Plot R2 score vs algorithm used

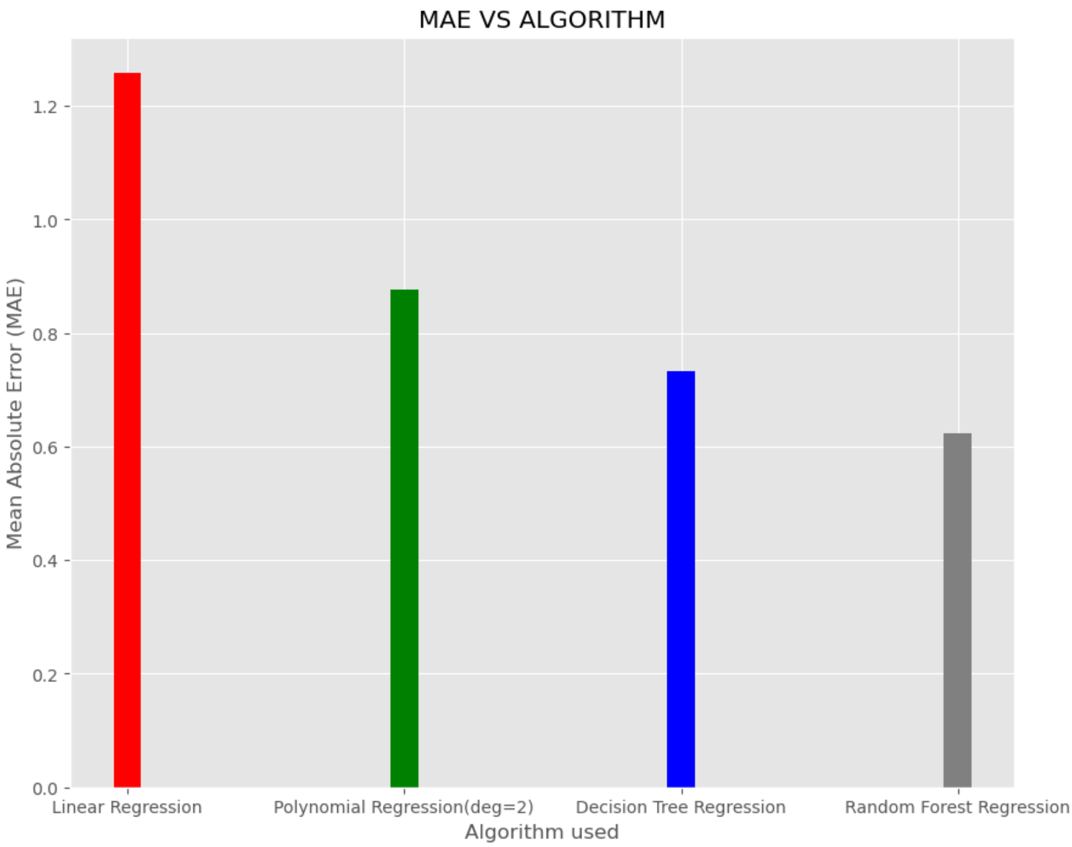

Fig. Plot MAE vs algorithm used

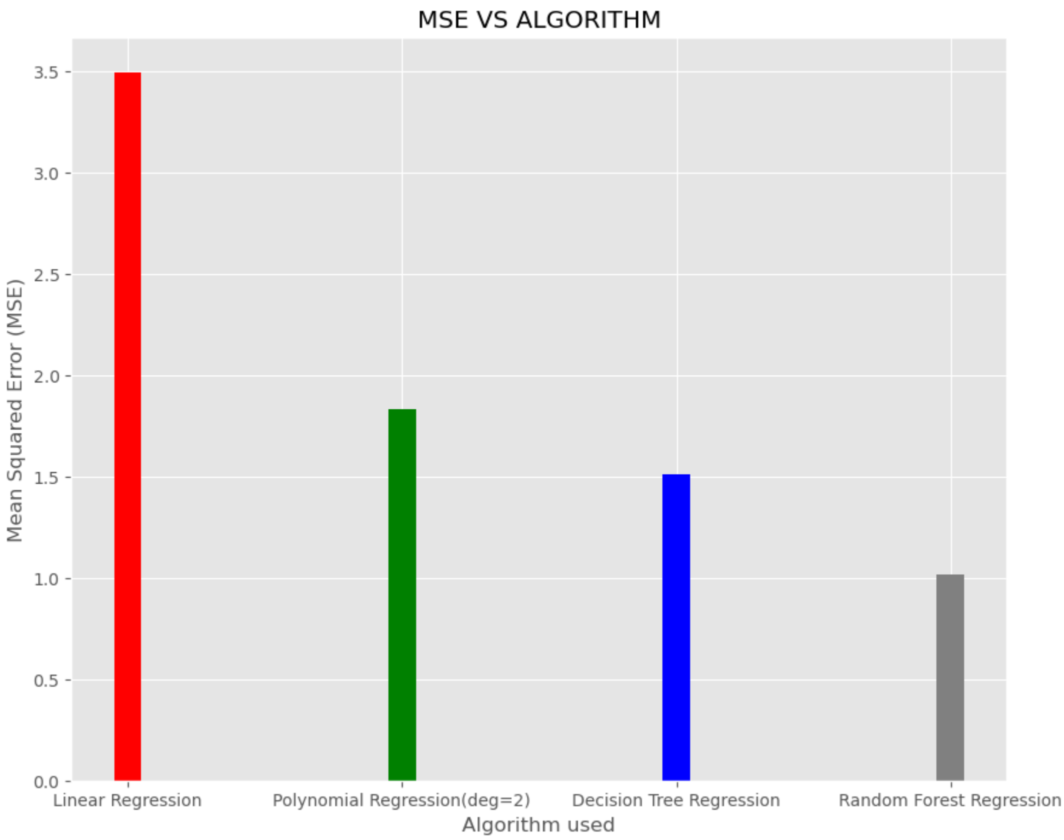

Fig. Plot MSE vs algorithm used

USER INTERFACE

Once we Train the model it can be used to predict new values, but first we need some mechanism to save this model, for this we use pickle module in python.

PICKLE

The pickle module in Python is used for serializing and deserializing Python objects to and from binary data. This means you can use pickle to save a Python object to disk and later load it back into memory, and the object will be in the exact same state as it was when you saved it. pickle is especially useful when you want to preserve the state of an object between sessions, or when you need to transfer an object over a network connection.

We can save the models as follows:

Saving Models

```
In [42]: filename = 'linearRegressionModel'
         outfile = open(filename,'wb')

         pickle.dump(model,outfile)
         outfile.close()
```

Fig. Saving models

PYGAME

The user interface is implemented in python using python module called pygame. Pygame is a python module that is basically used to develop computer games, it provides functionality of graphics, input and sound.

Fig. Pygame

SCREENSHOTS

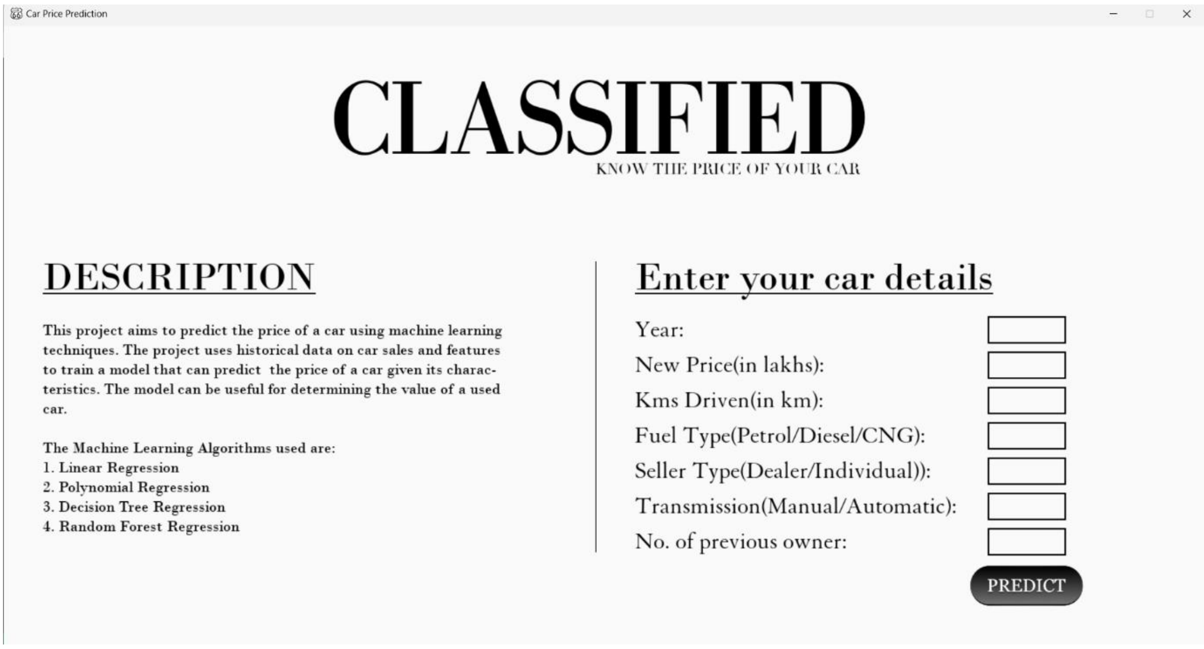

Fig. Screenshot of user interface

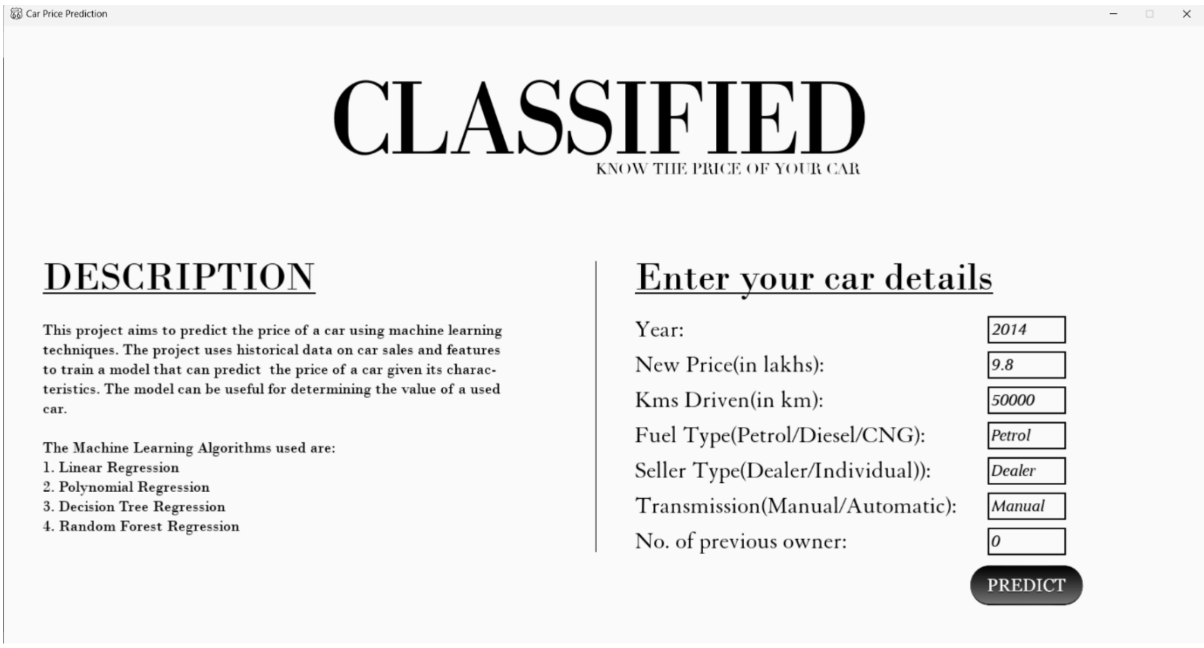

Fig. Screenshot of user interface

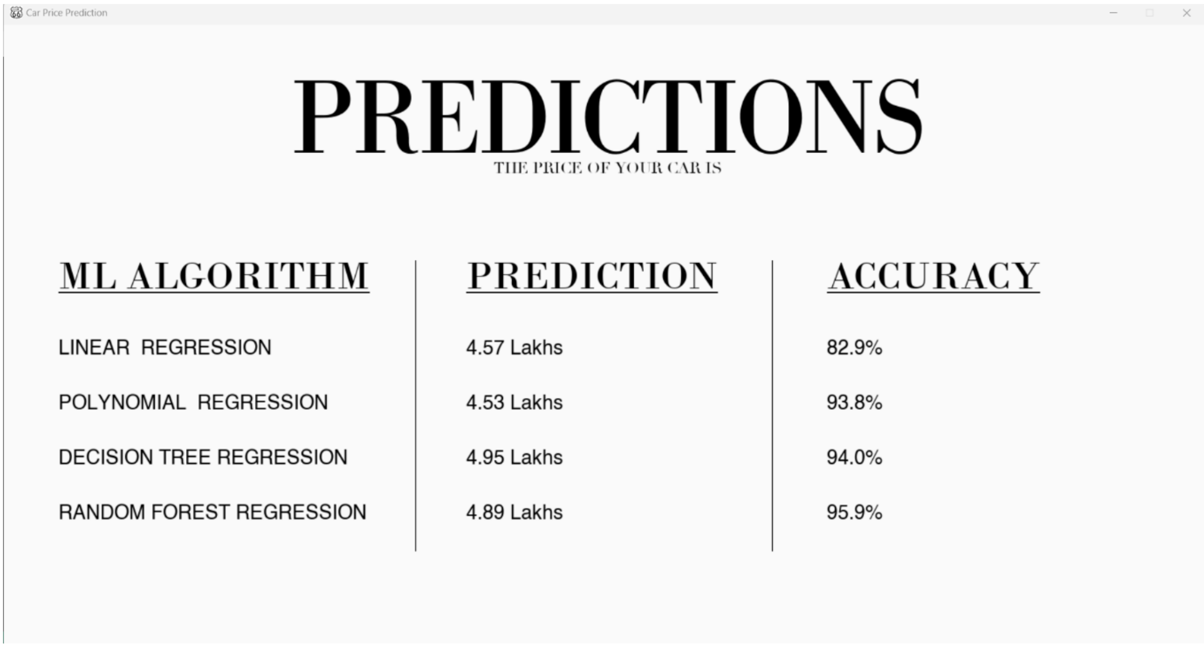

Fig. Screenshot of user interface

CONCLUSIONS

- We conclude that random forest regression is the best technique for determining used car prices.
- In future, datasets with more features which take into account the repair history, condition of the car, etc, will provide better accuracy.
- These algorithms will standardize the values of used vehicles, which will be very beneficial for the customers and buyers.
- These algorithms can be used in online platforms(such as OLX, Car Dekho etc,) dedicated to reselling of used vehicles.

.