INDEX

ABSTRACT

The price of a new car in the industry is fixed by the manufacturer with some additional costs incurred by the Government in the form of taxes. So, customers buying a new car can be assured of the money they invest to be worthy. But, due to the increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features.

Determining whether the listed price of a used car is a challenging task, due to the many factors that drive a used vehicle's price on the market. The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases. We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models across cities in the United States.

Our results show that Random Forest model yield the best results, but are compute heavy. Conventional linear regression also yielded satisfactory results, with the advantage of a significantly lower training time in comparison to the aforementioned methods

INTRODUCTION

A car price prediction has been a high interest research area, as it requires noticeable effort and knowledge of the field expert. Considerable number of distinct attributes are examined for the reliable and accurate prediction. Vehicle price prediction especially when the vehicle is used and not coming direct from the factory, is both a critical and important task. With increase in demand for used cars more and more vehicle buyers are finding alternatives of buying new cars. There is a need of accurate price prediction mechanism for the used cars. Prediction techniques of machine learning can be helpful in this regard. It is common to lease a car in many countries rather than buying a new car. After the lease period is over, the buyer has the possibility to buy the car at its residual value, i.e., its expected resale value.

This project aims to solve the problem of predicting the price of a used car, using Sklearn's supervised machine learning techniques. It is clearly a regression problem and predictions are carried out on dataset of cars data from Kaggle. Several regression techniques have been studied, including Linear Regression, Decision Trees and Random forests of decision trees. Their performances were compared in order to determine which one works best without dataset.

LITERATURE SURVEY

Sl no	Title	Author / Journalname / Year	Technique	Result
1.	Prediction of Prices for Used Car by Using Regression Models	Nitis Monburinon. Ieeexplore.org 2018	multiple linear regression, random forest, gradient boosting,	With gradient boosted regression trees gave the highest performance with only MAE =0.28. Followed by random forest regression with 0.35 errors, and multiple linear regression with 0.55 errors. Thus, we concluded that gradient boosted regression trees are recommended to develop the price evaluation model.
2.	Car Price Prediction using Machine Learning Techniques	Enis Gegic TEM Journal 2019	support vector machines, classification, machine learning.	Applying single machine algorithm on the data set accuracy was less than 50%. Therefore, the ensemble of multiple machine learning algorithms has been proposed and this combination of ML methods gains accuracy of 92.38%. This is significant improvement compared to single machine learning method approach
3.	Used Cars Price Prediction using Supervised Learning Techniques	Pattabiraman Venkatasubbu, International Journal of Engineering and Advanced Technology (IJEAT) 2019	ANOVA, Lasso Regression, Regression Tree, Tukey's Test	The prediction error rate of all the models was under 5% of error. But, on further analysis, the mean error of regression tree model was more than the multiple regression and lasso regression models. Even though the regression tree has better accuracy, its error rates are higher. This has been confirmed by performing an ANOVA. Also, the post-hoc test revealed that error rates in multiple regression models and lasso regression models aren't significantly different from each other

4.	Used Car Price Prediction using K-Nearest Neighbor Based Model	Dr R.Ashok Kumar International Journal of Innovative Research in Applied Sciences and Engineering (IJIRASE) 2020	K Nearest Neighbor, Prediction, Machine Learning, Used Cars Accuracy, Preprocessing, Regression, Cross-validation, K-Fold.	we have trained our model with used cars data set to predict the price. Here we have used K nearest Neighbor algorithm and we got accuracy 85% where the accuracy of linear regression is 71%. The proposed model is also validated with 5 and 10 folds by using K Fold Method. The experimental analysis shows that the proposed model is fitted as the optimized model.
5.	old car price prediction with machine learning	Akshay Gondaliya International Research Journal of Modernization in Engineering Technology and Science (IRJMETS) 2021	Machine Learning, Random Forest, XG boost, Decision Tree, Linear Regression	We have used total five supervised machine learning models, and their root mean squared error are, KNN Regressor: 7771.09, Linear Regression: 6846.23, XG Boost: 3980.77, Random Forest: 3702.34 and Decision Tree Regressor: 5590.43. Out of all Random Forest has lowest RMSE, and performed well with highest Rsquared value: 0.93. The limitation of this research is less number of records of old car. in future if we get more data than we can retrain our models and might get more accurate and stable model
6.	Used Cars Price Prediction using Machine Learning with Optimal Features	Samina Yasin Pakistan Journal of Engineering and Technology 2021	Machine learning, Regression, Statistical test, IF, OLS	Proposed methodology shows the model helps the customer and dealer for customer they support them for selling and buying and as well for dealers how much price they buy the used vehicle and when they sell them the proposed system helps the dealer for good profitable deal they can make. The 90% correct prediction results show that the study is effective and efficient for both market end nodes.

7.	Predicting the Price of Used Cars using Machine Learning Techniques	Sameerchand Pudaruth International Journal of Information & Computation Technology 2014	Multiple Linear Regression Analysis, K-Nearest Neighbours (kNN), Decision Trees, Naïve Bayes	The mean error with linear regression was about Rs51, 000 while for kNN it was about Rs27, 000 for Nissan cars and about Rs45, 000 for Toyota cars. J48 and NaiveBayes accuracy dangled between 60-70% for different combinations of parameters. The main weakness of decision trees and naïve bayes is their inability to handle output classes with numeric values. Hence, the price attribute had to be classified into classes which contained a range of prices but this evidently introduced further grounds for inaccuracies. The main limitation of this study is the low number of records that have been used.
8.	Research on the Prediction Model of the Used Car Price in View of the PSO-GRA-BP Neural Network	Enci Liu, Anni Zheng MDPI 2022	BP Neural Network, Pearson correlation, PSO Optimization Algorithm	This paper analyzes the factors affecting the price of used cars from three aspects— used car parameters, vehicle condition factors, and transaction factors— and establishes a used car price evaluation system including 12 characteristic variables. Using web crawler technology to obtain used car transaction data, three prediction models of BPNN, GRABPNN, and PSO-GRA-BPNN were constructed to conduct comparative verification and result analysis. In a rough comparison, the BPNN model has lower accuracy, with an error range of about 19.979%, and it is unstable. In the case of feature variable screening, the prediction accuracy of the GRA-BPNN model is higher than that of the BPNN, and the error range is about 13.986%. However, the constructed PSO-GRA-BPNN used car price prediction model not only has high accuracy but

				also has an error range of about 7.978%, which has good scalability
9.	Fair Price Prediction System for Used Cars in Sri Lanka Using Machine Learning and Robotic Process Automation	T P Jayadeera ICOBI 2019	Multiple Linear Regression, Decision Tree Regression, Gradient Boosting Regression, Random Forest Regression, K-means	Multiple Linear Regression and Random Forrest Regression algorithms work best in predicting the prices while Decision Tree Regression and Gradient Boosting Regression algorithms shows moderate performance. But K-Nearest Neighbors algorithm does not work as expected with both online and Sri Lankan data sets. By considering the above algorithms, a web interface has been developed as a system to generate the selling price of a particular used car, when the user enters the features of his/her expected car such as Model year, Transmission, Body type, Fuel type, Transmission, Engine capacity and Mileage. Thus, it can be concluded that predicting the price of used cars is a very risky enterprise but which is feasible.

DATASET

- The Dataset here we used is cars_data.csv taken from cardekho data from Kaggle.
- The dataset consists of 301 rows which not enough to get correct accuracy we used the Data Augmentation method and increased the columns to 20 times in which the updated dataset consists of around 6000 rows which gives us good accuracy for the model used for prediction.
- We will read the dataset by connecting it to MongoDB using MongoDB compass.

Car_Name	company	Year	Selling_Pri...	Present_Pr...	Kms_Driven	Fuel_Type	Seller_Type	Transmissi...	Owner
ritz	maruti suzuki	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
sx4	maruti suzuki	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
ciaz	maruti suzuki	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
wagon r	maruti suzuki	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
swift	maruti suzuki	2014	4.6	6.87	42450	Diesel	Dealer	Manual	0
vitara brezza	maruti suzuki	2018	9.25	9.83	2071	Diesel	Dealer	Manual	0
ciaz	maruti suzuki	2015	6.75	8.12	18796	Petrol	Dealer	Manual	0
s cross	maruti suzuki	2015	6.5	8.61	33429	Diesel	Dealer	Manual	0
ciaz	maruti suzuki	2016	8.75	8.89	20273	Diesel	Dealer	Manual	0
ciaz	maruti suzuki	2015	7.45	8.92	42367	Diesel	Dealer	Manual	0
alto 800	maruti suzuki	2017	2.85	3.6	2135	Petrol	Dealer	Manual	0
ciaz	maruti suzuki	2015	6.85	10.38	51000	Diesel	Dealer	Manual	0
ciaz	maruti suzuki	2015	7.5	9.94	15000	Petrol	Dealer	Automatic	0
ertiga	maruti suzuki	2015	6.1	7.71	26000	Petrol	Dealer	Manual	0

EXISTING SYSTEM AND DRAWBACKS

- Existing System includes a process where a seller decides a price randomly and buyer has no idea about the car and its value in the present-day scenario. In fact, seller also has no idea about the car's existing value or the price he should be selling the car at the seller will demand some price which is nearer to the cost he buy.

- The main drawback for this system is if the seller demands some cost for the vehicles but without knowing the condition and present value of that car in the market the buyer will give the amount and buy which is worth for 1 or 2 years for the buyer.

- Another drawback is that the car price varies for different brand & different fuel type but the buyer & seller will not know the differences and the buyer is ready to buy the CNG fuel type car also which is similar to cost of Diesel car.

- It is a big Drawback from seller side also like if the seller sell the diesel car similar to cost of CNG type car he may got loss because diesel cars have high value compare to CNG type cars.

PROPOSED SYSTEM

- The focus of this project is developing machine learning models that can accurately predict the price of a used car based on its features, in order to make informed purchases.

- We will use the dataset that is loaded from the MongoDB compass which we imported in the MongoDB compass.

- We implement and evaluate various learning methods on a dataset consisting of the sale prices of different makes and models.

- We will compare the performance of various machine learning algorithms like Linear Regression, Ridge Regression, Lasso Regression, Elastic Net, Decision Tree Regressor and choose the best out of it. Depending on various parameters we will determine the price of the car.

- Regression Algorithms are used because they provide us with continuous value as an output and not a categorized value because of which it will be possible to predict the actual price a car rather than the price range of a car.

- User Interface has also been developed which acquires input from any user and displays the Price of a car according to user's inputs.

HARDWARE/SOFTWARE USED

MongoDB Compass:

MongoDB Compass is a powerful GUI for querying, aggregating, and analyzing your MongoDB data in a visual environment.

MongoDB Compass is a convenient tool for browsing through the data stored in a MongoDB database through a graphical interface. It removes the burden of having to remember the names of obscure databases or collections, and you can navigate to any database or collection on your MongoDB server with just a few clicks.

Compass can help you accomplish, such as importing and managing data from an easy-to-navigate interface.

For Importing the data first, Connect to a MongoDB deployment hosted on MongoDB Atlas, or a deployment hosted locally on your own machine.

Next, Import data from CSV or JSON files into your MongoDB database from your device.

Jupyter Notebook:

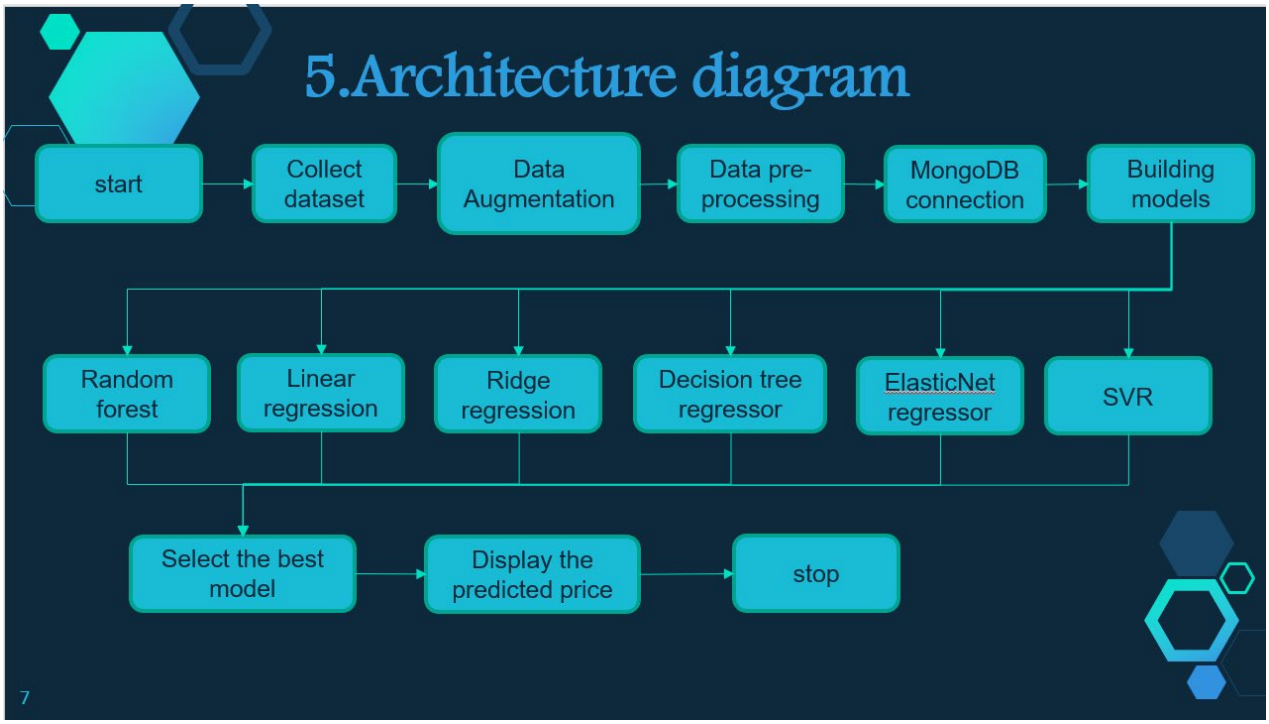
The Jupyter Notebook is the original web application for creating and sharing computational documents. It offers a simple, streamlined, document-centric experience.

Excel:

By organizing data using software like Excel, data analysts and other users can make information easier to view as data is added or changed. Excel contains a large number of boxes called cells that are ordered in rows and columns. Data is placed in these cells.

Analyze Data in Excel empowers you to understand your data through natural language queries that allow you to ask questions about your data without having to write complicated formulas.

SYSTEM ARCHITECTURE



MODULE WISE DESCRIPTION

Data collection & Augmentation:

We have taken our used car dataset from Kaggle
We will be using single dataset and further splits into two datasets:
Train.csv and test.csv
Train.csv is used to train the model.
Test.csv is used to predict the price of the cars.
The dataset consists of 300 rows and 9 columns
We will perform Data Augmentation for our dataset.

- Data Augmentation is a very useful tool to have in your ML arsenal. It is a technique used to increase the amount of data available for training a machine learning model

- Data augmentation can be used when there is a limited amount of training data available. By artificially generating new data points, data augmentation can help prevent overfitting and improve the generalizability.

- It's a powerful way to increase the amount of data you have without having to actually collect more data. Here are some key reasons why you should use data augmentation in machine learning:

    1) **More Data, More Better**: The more data you have, the better your machine learning models will be. Data augmentation can help you increase the amount of data you have without having to actually collect more data.
    2) **Generalization**: Data augmentation can help your machine learning models generalize better to new data. When your models see new data that is similar to the data they were trained on, they will be better able to make accurate predictions.
    3) **Overfitting:** Data augmentation can help reduce overfitting. Overfitting is a problem in machine learning where your models perform well on the training data but not so well on the test data. This is because your models have learned the training data too well and have not generalize well to new data. Data augmentation can help reduce overfitting by providing your models with more data to learn from.

4) **Computational Efficiency**: Data augmentation can also be used to increase the computational efficiency of your machine learning models. This is because you can train your models on smaller datasets and still get good results.

MongoDB Data Connection

- For Connection of our CSV dataset to MongoDB we will use MongoDB Compass software to connect.
- For Importing the data first, connect to a MongoDB deployment hosted on MongoDB Atlas, or a deployment hosted locally on your own machine.
- Next, Import data from CSV or JSON files into your MongoDB database from your device.
- After Importing we have to connect to the same host through Jupyter notebook in which MongoDB deployment hosted and we have to take the imported data from the database and the collection that we have defined.
- Now, we have to just read the data using the attributes of the dataset we have to perform this coding in jupyter notebook
- Then, the data which we imported will be ready to use for the use of machine learning models.

Machine Learning Models

1. Random Forest:

- Random Forest Regression is a supervised learning algorithm that uses ensemble learning method for regression. A Random Forest operates by constructing several decision trees during training time and outputting the mean of the classes as the prediction of all the trees.
- It takes less training time as compared to other algorithms.
- It predicts output with high accuracy, even for the large dataset it runs efficiently.
- It can also maintain accuracy when a large proportion of data is missing.
- We use Random forest Hyperparameters in which they are parameters whose values control the learning process and determine the values of model parameters that a learning algorithm ends up learning.

2. Decision Tree:

- Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.
- In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node.

3. Linear Regression:

- Linear regression is one of the easiest and most popular Machine Learning algorithms. It is a statistical method that is used for predictive analysis. Linear regression makes predictions for continuous/real or numeric variables such as sales, salary, age, product price, etc.
- Linear regression algorithm shows a linear relationship between a dependent (y) and one or more independent (y) variables, hence called as linear regression. Since linear regression shows the linear relationship, which means it finds how the value of the dependent variable is changing according to the value of the independent variable.

4. Ridge Regression:

- Ridge Regression is a popular type of regularized linear regression that includes an L2 penalty. This has the effect of shrinking the coefficients for those input variables that do not contribute much to the prediction task.
- Ridge Regression is an extension of linear regression that adds a regularization penalty to the loss function during training.

5. Elastic Net Regression:

- Elastic net is always preferred over lasso & ridge regression because it solves the limitations of both methods, while also including each as special cases. So, if the ridge or lasso solution is, indeed, the best, then any good model selection routine will identify that as part of the modelling process.

6. Support Vector Machine(SVM):

- Support Vector Machine or SVM is one of the most popular Supervised Learning algorithms, which is used for Classification as well as Regression problems. However, primarily, it is used for Classification problems in Machine Learning.
- The Support Vector Regression (SVR) uses the same principles as the SVM for classification, with only a few minor differences. First of all, because output is a real number it becomes very difficult to predict the information at hand, which has infinite possibilities.
- In the case of regression, a margin of tolerance (epsilon) is set in approximation to the SVM which would have already requested from the problem. But besides this fact, there is also a more complicated reason, the algorithm is more complicated therefore to be taken in consideration.

RESULTS & DISCUSSION

Connecting to MongoDB using localhost port and taking data from our database in which is already imported in the MongoDB compass.

```
In [45]: client=MongoClient('localhost',27017)

In [46]: client.list_database_names()

Out[46]: ['LibMS',
          'admin',
          'config',
          'invetman',
          'local',
          'priceprediction',
          'sample_mflix',
          'stdportal',
          'test']

In [47]: client = pymongo.MongoClient("mongodb://localhost:27017/")

         # Database Name
         db = client["priceprediction"]

         # Collection Name
         col = db["df"]


         df = col.find()
         _id=[]
         Car_Name=[]
         Year=[]
         Selling_Price=[]
         Present_Price=[]
         Kms_Driven=[]
         Fuel_Type=[]
         Seller_Type=[]
         Transmission=[]
         Owner=[]
         for df in df:
             _id.append(df["_id"])
             Car_Name.append(df["Car_Name"])
             Year.append(df["Year"])
             Selling_Price.append(df["Selling_Price"])
             Present_Price.append(df["Present_Price"])
             Kms_Driven.append(df["Kms_Driven"])
             Fuel_Type.append(df["Fuel_Type"])
             Seller_Type.append(df["Seller_Type"])
             Transmission.append(df["Transmission"])
             Owner.append(df["Owner"])
```

Reading data which has been taken from MongoDB compass

```
In [48]: df=pd.DataFrame({"_id":_id,"Car_Name":Car_Name,"Year":Year,"Selling_Price":Selling_Price,"Present_Price":Present_Price,"Kms_Drive
         df.head(5)
```

Out[48]:

	_id	Car_Name	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner
0	636d337e0311f0a4442ac953	ritz	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0
1	636d337e0311f0a4442ac954	sx4	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0
2	636d337e0311f0a4442ac955	ciaz	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0
3	636d337e0311f0a4442ac956	wagon r	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0
4	636d337e0311f0a4442ac957	swift	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0

Checking if there are any missing values in our data.

```
In [52]: df.isnull().sum()
```

```
Out[52]: Car_Name          0
         Year              0
         Selling_Price     0
         Present_Price     0
         Kms_Driven        0
         Fuel_Type         0
         Seller_Type       0
         Transmission      0
         Owner             0
         dtype: int64
```

Here we are calculating how old is the vehicle, by subtracting the current year – the year car bought.
So we will be getting to know how old is the car.

```
In [58]: final_dataset['Current_Year'] = 2022
```

```
In [59]: final_dataset.head()
```

Out[59]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2022
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2022
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2022
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2022
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2022

```
In [60]: final_dataset['No_Of_Years'] = final_dataset['Current_Year'] - final_dataset['Year']
```

```
In [61]: final_dataset.head()
```

Out[61]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current_Year	No_Of_Years
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2022	8
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2022	9
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2022	5
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2022	11
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2022	8

Here dummies is used to divide the categorical data separately

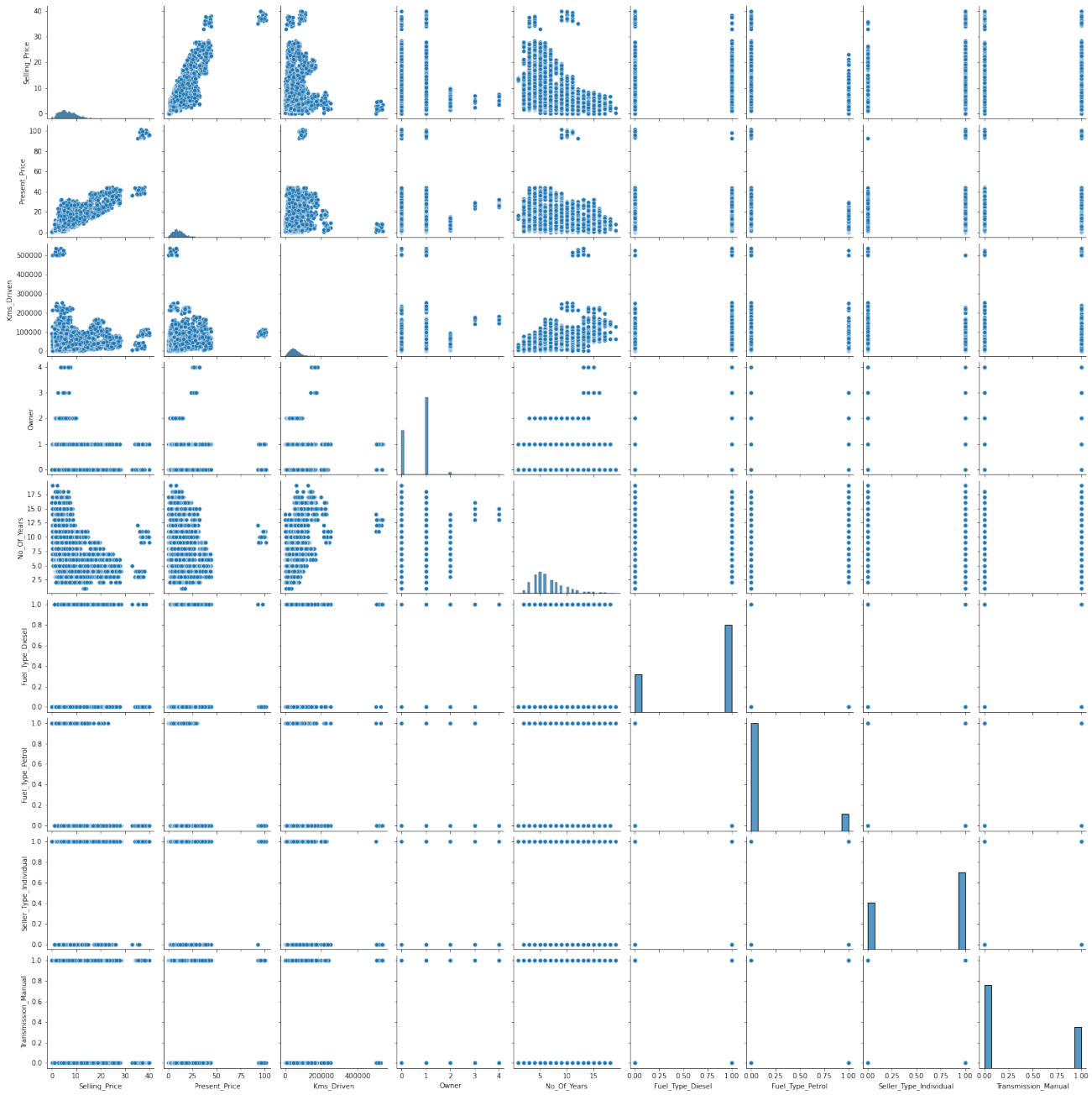
```
In [64]: final_dataset = pd.get_dummies(final_dataset, drop_first=True)
```

```
In [65]: final_dataset.head()
```

Out[65]:

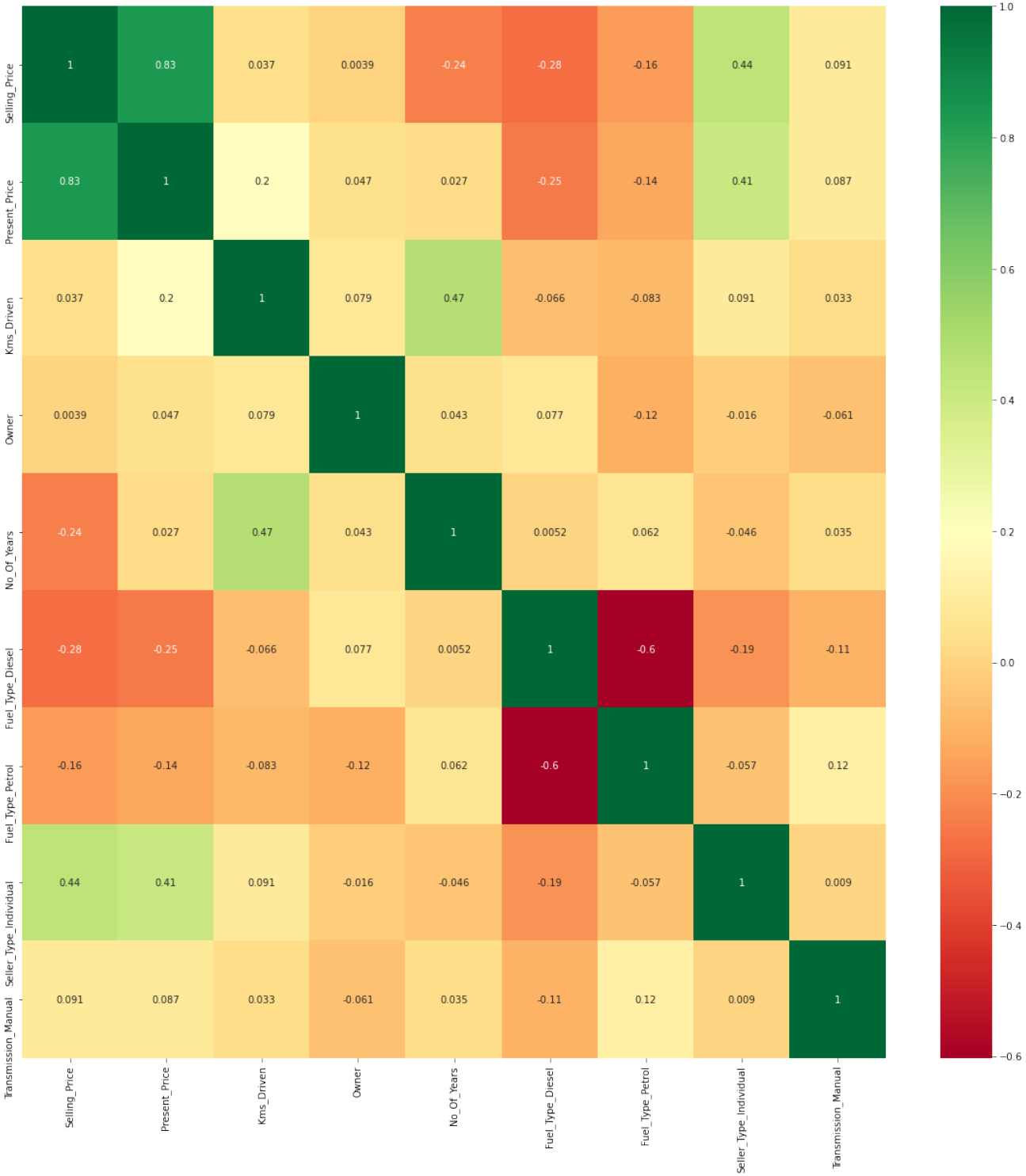
	Selling_Price	Present_Price	Kms_Driven	Owner	No_Of_Years	Fuel_Type_Diesel	Fuel_Type_Petrol	Seller_Type_Individual	Transmission_Manual
0	3.35	5.59	27000	0	8	0	1	0	1
1	4.75	9.54	43000	0	9	1	0	0	1
2	7.25	9.85	6900	0	5	0	1	0	1
3	2.85	4.15	5200	0	11	0	1	0	1
4	4.60	6.87	42450	0	8	1	0	0	1

The below diagram is the Pairplot for the categorical data when it is divided which is used to plot multiple pairwise bivariate distributions in a dataset.



Heat Maps are used to show relationships between two variables, one plotted on each axis.

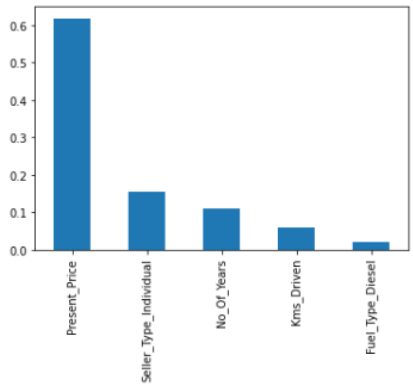
By observing how cell colors change across each axis, you can observe if there are any patterns in value for one or both variables.



Here the diagram shows which features have much importance in our project like as we can see the present_price plays a important role in predicting the price.

```
In [73]: feature_importance = pd.Series(model.feature_importances_, index=X.columns)
         feature_importance.nlargest(5).plot(kind='bar')
         print(model.feature_importances_)
```

```
[0.61898364 0.05842407 0.01217536 0.10830129 0.02130353 0.01837955
 0.15387072 0.00856185]
```



Defining Random Forest with its Hyper parameters
Random forest gave accuracy of 0.98 (highest).

```
In [77]: from sklearn.model_selection import RandomizedSearchCV
```

```
In [78]:
#Randomized Search CV

# Number of trees in random forest
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num = 12)]
# Number of features to consider at every split
max_features = ['auto', 'sqrt']
# Maximum number of levels in tree
max_depth = [int(x) for x in np.linspace(5, 30, num = 6)]
# max_depth.append(None)
# Minimum number of samples required to split a node
min_samples_split = [2, 5, 10, 15, 100]
# Minimum number of samples required at each leaf node
min_samples_leaf = [1, 2, 5, 10]
```

```
In [79]: # Create the random grid
random_grid = {'n_estimators': n_estimators,
               'max_features': max_features,
               'max_depth': max_depth,
               'min_samples_split': min_samples_split,
               'min_samples_leaf': min_samples_leaf}

print(random_grid)
```

```
{'n_estimators': [100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200], 'max_features': ['a
uto', 'sqrt'], 'max_depth': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_s
amples_leaf': [1, 2, 5, 10]}
```

```
In [80]: rf = RandomForestRegressor()
```

```
In [81]:
rf.fit(X_train,y_train)
rf.score(X_train,y_train)
```

```
Out[81]: 0.9831208989520389
```

Defining Decision tree regressor model

```
In [84]: from sklearn import metrics

In [85]: #Decision Tree
         from sklearn.tree import DecisionTreeRegressor
         dtree = DecisionTreeRegressor()
         dtree.fit(X_train, y_train)

Out[85]: DecisionTreeRegressor()

In [86]: y_pred_dtree = dtree.predict(X_test)
         y_pred_dtree

Out[86]: array([ 5.22110803,  6.50734454,  9.20112581, ..., 14.07810869,
                23.2146383 ,  5.71958444])

In [87]: dtree.score(X_train,y_train)
         #acc_dtree = metrics.accuracy_score(y_test,y_pred_dtree)
         #acc_dtree

Out[87]: 1.0
```

Defining Linear regression model and got accuracy of 0.78

```
In [88]: #Linear Regression
         from sklearn.linear_model import LinearRegression
         import matplotlib.pyplot as plt

In [89]: lr = LinearRegression()
         lr.fit(X_train, y_train)

Out[89]: LinearRegression()

In [90]: y_pred_lr =  lr.predict(X_test)
         y_pred_lr

Out[90]: array([ 8.80058261,  6.52651873,  7.99626477, ..., 10.01748243,
                22.1480835 ,  4.18978873])

In [91]: lr.score(X_train, y_train)

Out[91]: 0.7871901060537148
```

Defining Ridge regression and got accuracy of 0.78

```
In [92]: #Ridge Regression
         from sklearn.linear_model import Ridge
         ridge = Ridge(alpha=1.0)
         ridge.fit(X_train, y_train)

Out[92]: Ridge()

In [93]: y_pred_ridge = ridge.predict(X_test)
         y_pred_ridge

Out[93]: array([ 8.79451721,  6.52688795,  7.99713012, ..., 10.01172067,
                22.14569145,  4.19036925])

In [94]: ridge.score(X_train, y_train)

Out[94]: 0.7871898626244254
```

Defining ElasticNet regressor and got accuracy of 0.76

```
In [95]: #ElasticNet Regressor
         from sklearn.linear_model import ElasticNet
         elas = ElasticNet(random_state=1046)
         elas.fit(X_train, y_train)

Out[95]: ElasticNet(random_state=1046)

In [96]: y_pred_elas = elas.predict(X_test)
         y_pred_elas

Out[96]: array([ 6.67062981,  6.227596  ,  7.94657094, ...,  8.80086646,
                21.20887399,  4.82303478])

In [97]: elas.score(X_train, y_train)

Out[97]: 0.7601522141337576
```

Defining Support vector regressor and got accuracy of 0.823

```
In [98]: sc = StandardScaler()
         X1_train = sc.fit_transform(X_train)
         X_test1 = sc.transform(X_test)

In [99]: #SVM
         from sklearn import svm
         svm = svm.SVR()
         svm.fit(X1_train, y_train)
         svm.score(X_test1,y_test)
         svm.score(X1_train,y_train)
         y_pred = svm.predict(X_test1)

In [100]: svm.score(X1_train,y_train)

Out[100]: 0.8239177864457375
```

Cross validation Score:

Cross-validation is a technique for evaluating ML models by training several ML models on subsets of the available input data and evaluating them on the complementary subset of the data. Use cross-validation to detect overfitting, ie, failing to generalize a pattern.

We will use k-fold cross validation method which improves model prediction and in which random forest has highest cross validation score we will take random forest for the used car price prediction.

In [102]:
```
#Cross Validation Score
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import RandomForestClassifier
```

In [103]:
```
clf1 = RandomForestClassifier(n_estimators=100, criterion='gini',max_depth=2,oob_score=True)
rf1 = rf.fit(X1_train, y_train)
cross_val_1 = cross_val_score(rf, X1_train, y_train)
print('cross val:',cross_val_1)
```

```
cross val: [0.87568707 0.86917585 0.88562323 0.88842232 0.86906737]
```

In [104]:
```
cross_val_2 = cross_val_score(dtree, X1_train, y_train)
print('cross val:',cross_val_2)
```

```
cross val: [0.77771209 0.77059477 0.79001196 0.79574234 0.76696908]
```

In [105]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(dtree , X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.7794756965880215
Avg accuracy when k=6:0.7814401008669969
Avg accuracy when k=7:0.771000913398602
Avg accuracy when k=8:0.7792737399053686
Avg accuracy when k=9:0.7678490454045577
```

In [106]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(rf, X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.8767113239393746
Avg accuracy when k=6:0.878609668113187
Avg accuracy when k=7:0.878219473022294
Avg accuracy when k=8:0.8780586035442508
Avg accuracy when k=9:0.878730605947219
```

In [107]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(lr, X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.7852174730912604
Avg accuracy when k=6:0.7853569130872474
Avg accuracy when k=7:0.7840435127523901
Avg accuracy when k=8:0.7845955854316511
Avg accuracy when k=9:0.7848692496019767
```

In [108]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(ridge, X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.7852190509200115
Avg accuracy when k=6:0.7853587990525588
Avg accuracy when k=7:0.7840458024326672
Avg accuracy when k=8:0.7845976201475799
Avg accuracy when k=9:0.7848712028787872
```

In [109]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(elas, X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

```
Avg accuracy when k=5:0.6449806695025921
Avg accuracy when k=6:0.645432540294224
Avg accuracy when k=7:0.6449759469789631
Avg accuracy when k=8:0.6449696272841281
Avg accuracy when k=9:0.6448813565006914
```

In [110]:
```
for u in range(5,10):

    kf = KFold(n_splits=u, random_state=None)
    result = cross_val_score(svm, X1_train, y_train, cv = kf)
    print("Avg accuracy when k="+ str(u) +   ":"   "{}".format(result.mean()))
```

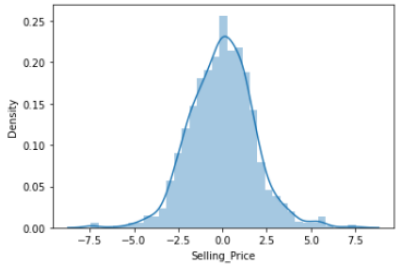
```
Avg accuracy when k=5:0.7999719980310854
Avg accuracy when k=6:0.801060823342849
Avg accuracy when k=7:0.8031800295410686
Avg accuracy when k=8:0.804333123466461
Avg accuracy when k=9:0.8051134782490924
```

Predicting the price using Random Forest model
Here random forest is trained with its Hyperparameters.

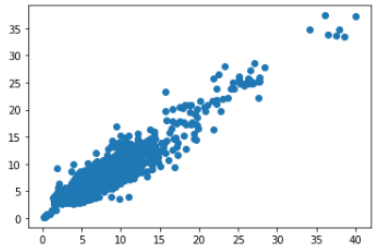
```
In [68]: # Random search of parameters, using 3 fold cross validation,
         # search across 100 different combinations
         rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10, c

In [69]: rf_random.fit(X_train,y_train)

         Fitting 5 folds for each of 10 candidates, totalling 50 fits
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.1s
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.7s
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.7s
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   2.9s
         [CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   3.2s
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.3s
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.3s
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.4s
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.3s
         [CV] END max_depth=15, max_features=sqrt, min_samples_leaf=2, min_samples_split=10, n_estimators=1100; total time=   4.3s
```

Here first graph is plotted for the y-test and selling_price
Second graph shows the scatter plot between y-test and predictions

```
In [73]: sns.distplot(y_test-predictions)

         C:\Users\Kasam Abhishek\Miniconda3\lib\site-packages\seabor
         unction and will be removed in a future version. Please ada
         imilar flexibility) or `histplot` (an axes-level function f
           warnings.warn(msg, FutureWarning)

Out[73]: <matplotlib.axes._subplots.AxesSubplot at 0x23516645d08>
```




```
In [74]: plt.scatter(y_test,predictions)

Out[74]: <matplotlib.collections.PathCollection at 0x2352e4e87c8>
```

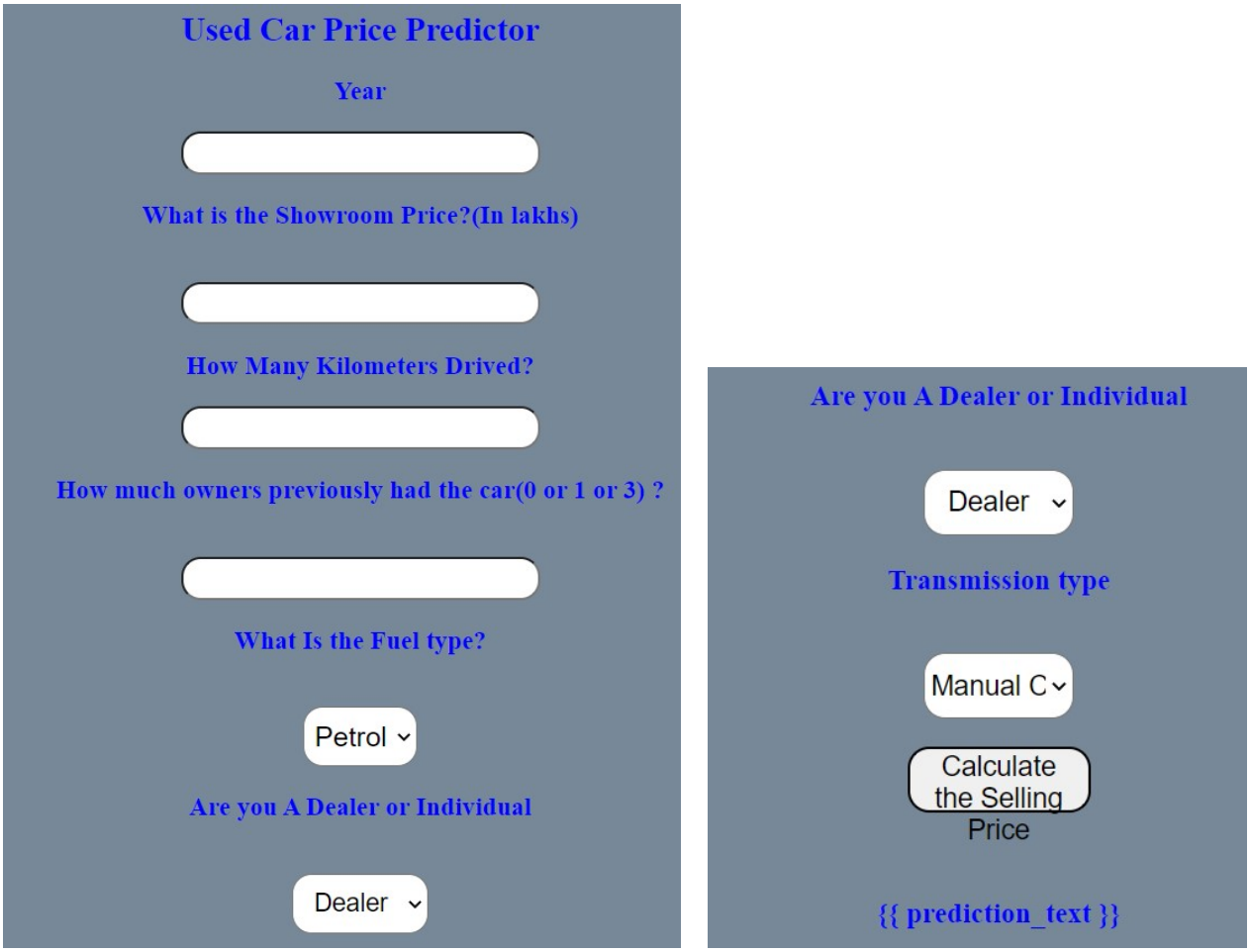


pkl can serialize a very wide range of objects, not just text data. Using pickle will save the dataframe as a dataframe object, rather than exporting it.

```
In [77]: import pickle
         file = open('random_forest_regression_model.pkl', 'wb')

         # dump information to that file
         pickle.dump(rf_random, file)
```

The user interface for the Used_car_price_prediction will be as follows:

The prediction test shown in the below image shows the resulted selling price in lakhs.



CONCLUSION

- The increased prices of new cars and the financial incapability of the customers to buy them, Used Car sales are on a global increase. Therefore, there is an urgent need for a Used Car Price Prediction system which effectively determines the worthiness of the car using a variety of features. The proposed system will help to determine the accurate price of used car price prediction
- So in this project we have developed different models for the best accuracy and finally , Random forest model has the highest accuracy and been selected as a best model for our project .
- Random Forest model is considered as best model for Used car Price Prediction.

FUTURE WORKS

In future this machine learning model may bind with various website which can provide real time data for price prediction. Also, we may add large historical data of car price which can help to improve accuracy of the machine learning model. We can build an android app as user interface for interacting with user. For better performance, we plan to judiciously design deep learning network structures, use adaptive learning rates and train on clusters of data rather than the whole data.

CONTRIBUTION

Worklet Tasks	Contributor's Names
Database connection and integration using MongoDB	Abhishek & Venkat Reddy
Preprocessing	Sree Reddy& Aravind Reddy
Model building	Abhishek, Sree Reddy & Aravind Reddy
Visualization	Abhishek & Sree Reddy
Report writing & Literature Survey	Sree Reddy, Abhishek & Aravind Reddy
Creating GUI using python	Sree Reddy, Abhishek & Venkat Reddy
Presentation preparation	Sree Reddy, Abhishek, Venkat Reddy & Aravind Reddy

REFERENCES

1. Sameerchand Pudaruth, "Predicting the Price of Used Cars using Machine Learning Techniques";(IJICT 2014)

2. Enis gegic, Becir Isakovic, Dino Keco, Zerina Masetic, Jasmin Kevric, "Car Price Prediction Using Machine Learning"; (TEM Journal 2019)

3. https://scikit-learn.org/stable/modules/classes.html: Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

4. Dataset link was taken from https://www.kaggle.com/shindenikhil/car-dekho-data