BANGLADESH ARMY INTERNATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY (BAIUST)

Course Title: *Artificial intelligence Sessional*

Course Code: *CSE-404*

Project report on used car price prediction using machine learning

Submitted By

Hritu raz Banik - 1105033

Shubha Bhuiyan - 1105037

Salman Karim - 1105066

Susanta Sarkar - 1105080

Supervised By,

Shovon Bhowmik

Lecturer, Dept. of CSE

Acknowledgement

At the very beginning, we would express our gratitude to Almighty Allah, the most gracious, most merciful for granting us the strength and patience to see through the completion of this project successfully. Without His endless blessings, it would have not been possible to complete such a task within the given time.

We would also like to show our sincere gratitude to our project supervisor and course teacher Shovon Bhowmik for his endless support and supervision. His support throughout the project had helped us better understand the task we are to perform, and the result we are to produce. His helped

we look into matters that would have been overlooked otherwise.

We recall our teachers, friends, and all others who had inspired and helped us throughout the completion of the project.

Date:

The Author,

Department of Computer Science and Engineering,

Bangladesh Army International University of Science and Technology.

Abstract

In many branches of materials science it is now routine to generate data sets of such large size and dimensionality that conventional methods of analysis fail. Paradigms and tools from data science and machine learning can provide scalable approaches to identify and extract trends and patterns within voluminous data sets, To understand what a dataset is, we must first discuss the components of a dataset. A single row of data is called an instance. Datasets are a collection of instances that all share a common attribute. Machine learning models will generally contain a few different datasets, each used to fulfill various roles in the system.

For machine learning models to understand how to perform various actions, training datasets must first be fed into the machine learning algorithm, followed by validation datasets (or testing datasets) to ensure that the model is interpreting this data accurately.

Once you feed these training and validation sets into the system, subsequent datasets can then be used to sculpt your machine learning model going forward. The more data you provide to the ML system, the faster that model can learn and improve.

Work Summary

A data set is a collection of data. ... In Machine Learning projects, we need a training data set. It is the actual data set used to train the model for performing various actions

This dataset contains information about used cars.
This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning.
The columns in the given dataset are as follows:

1. name
2. year
3. selling_price
4. km_driven
5. fuel
6. seller_type
7. transmission
8. Owner

About this file

This dataset contains information about used cars.
This data can be used for a lot of purposes such as price prediction to exemplify the use of linear regression in Machine Learning.

name	year	selling_price	km_driven	fuel	
Name of the cars	Year of the car when it was bought	Price at which the car is being sold	Number of Kilometres the car is driven	Fuel type of car (petrol / diesel / CNG / LPG / electric)	Te In
Maruti Swift Dzire VDI 2% Maruti Alto 800 LXI 1% Other (4212) 97%	1992 … 2020	20000.00 - 908000.00 Count: 3,908 20.0k … 8.90m	1 … 807k	Diesel 50% Petrol 49% Other (64) 1%	In De Ot
Maruti 800 AC	2007	60000	70000	Petrol	Ir
Maruti Wagon R LXI Minor	2007	135000	50000	Petrol	Ir
Hyundai Verna 1.6 SX	2012	600000	100000	Diesel	Ir
Datsun RediGO T Option	2017	250000	46000	Petrol	Ir
Honda Amaze VX i-DTEC	2014	450000	141000	Diesel	Ir
Maruti Alto LX BSIII	2007	140000	125000	Petrol	Ir
Hyundai Xcent 1.2 Kappa S	2016	550000	25000	Petrol	Ir
Tata Indigo Grand Petrol	2014	240000	60000	Petrol	Ir

This data set from https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho contains information about used car by using this data set implement a prediction model to predict selling price for car.

Work description

We have created a new environment in anaconda prompt as every model has different dependencies.

Then we start working in jupyter notebook in python 3.7

```
In [1]: import pandas as pd
```

```
In [2]: df=pd.read_csv('car data.csv')
```

```
In [3]: df.head()
```

Out[3]:

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

```
In [4]: df.shape
Out[4]: (301, 9)
```

```
In [5]: print(df['Seller_Type'].unique())
        print(df['Fuel_Type'].unique())
        print(df['Transmission'].unique())
        print(df['Owner'].unique())

        ['Dealer' 'Individual']
        ['Petrol' 'Diesel' 'CNG']
        ['Manual' 'Automatic']
        [0 1 3]
```

```
In [6]: df.isnull().sum()
Out[6]: 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
```

```
In [7]: df.describe()
```

As there is no null value no need to determine mean value.

Next thing we have done, we used df.describe( ) to see some feature of data set such as standard deviation, count etc.

Preprocessing

Then we created our final data set we exclude car_name as it doesn't include mathematical importance.

After that we created no_year

```
In [12]: final_dataset['no_year']=final_dataset['Current Year']- final_dataset['Year']
```

```
In [13]: final_dataset.head()
```

Out[13]:

	Year	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current Year	no_year
0	2014	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	2013	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	2017	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2011	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	2014	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

```
In [14]: final_dataset.drop(['Year'],axis=1,inplace=True)
```

```
In [15]: final_dataset.head()
```

Out[15]:

	Selling_Price	Present_Price	Kms_Driven	Fuel_Type	Seller_Type	Transmission	Owner	Current Year	no_year
0	3.35	5.59	27000	Petrol	Dealer	Manual	0	2021	7
1	4.75	9.54	43000	Diesel	Dealer	Manual	0	2021	8
2	7.25	9.85	6900	Petrol	Dealer	Manual	0	2021	4
3	2.85	4.15	5200	Petrol	Dealer	Manual	0	2021	10
4	4.60	6.87	42450	Diesel	Dealer	Manual	0	2021	7

And after we dropped Year column.

Convert categorical variable into dummy/indicator variables

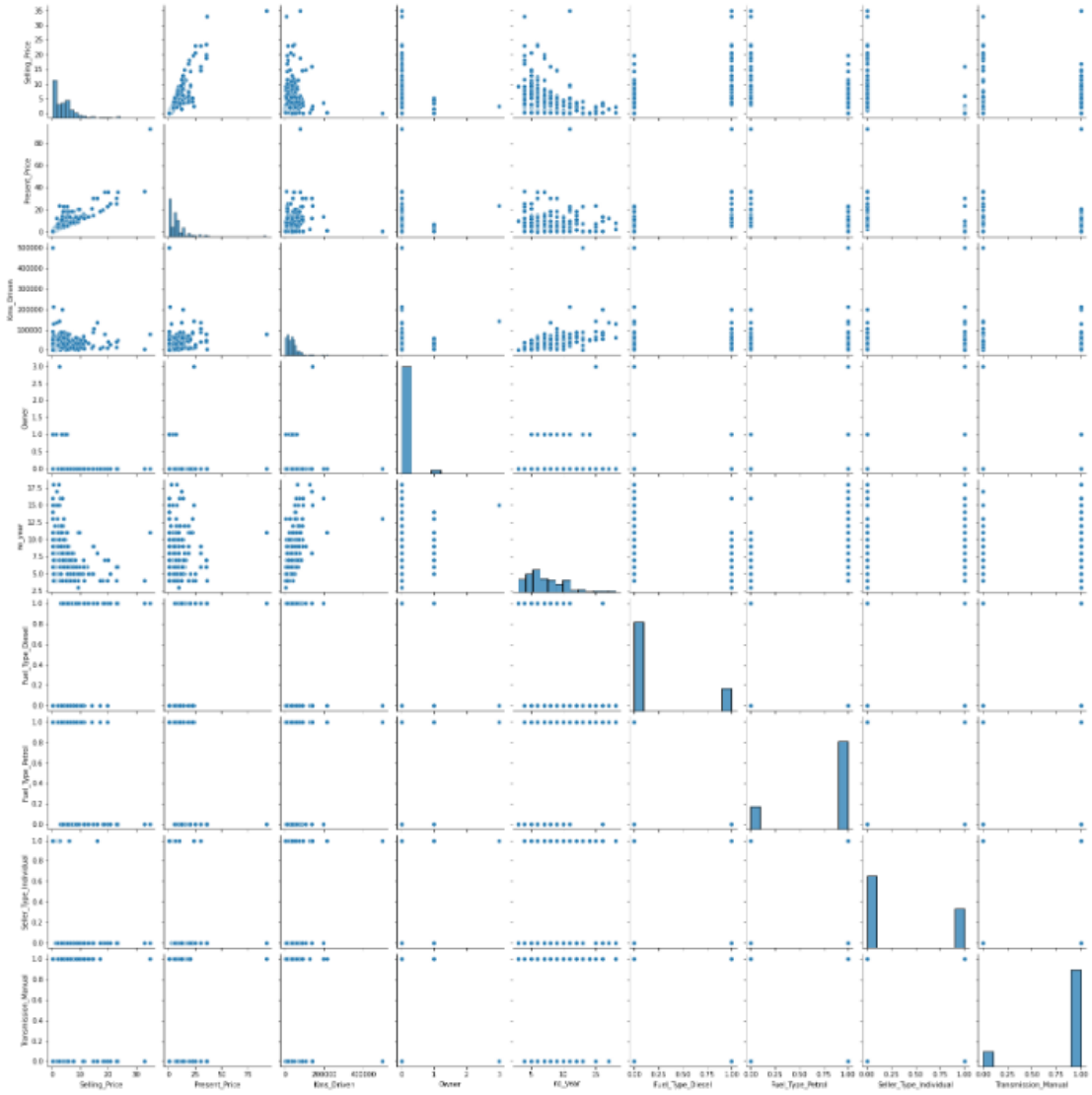
Here fuel_type is a categorical variable.
final_dataset=pd.get_dummies(final_dataset,drop_first=True) is used to convert it.

Next step is finding co-relation using .corr function. Co-relation say how one feature is connected to another.

Using seaborn we can see pairplot

Which is shown bellow

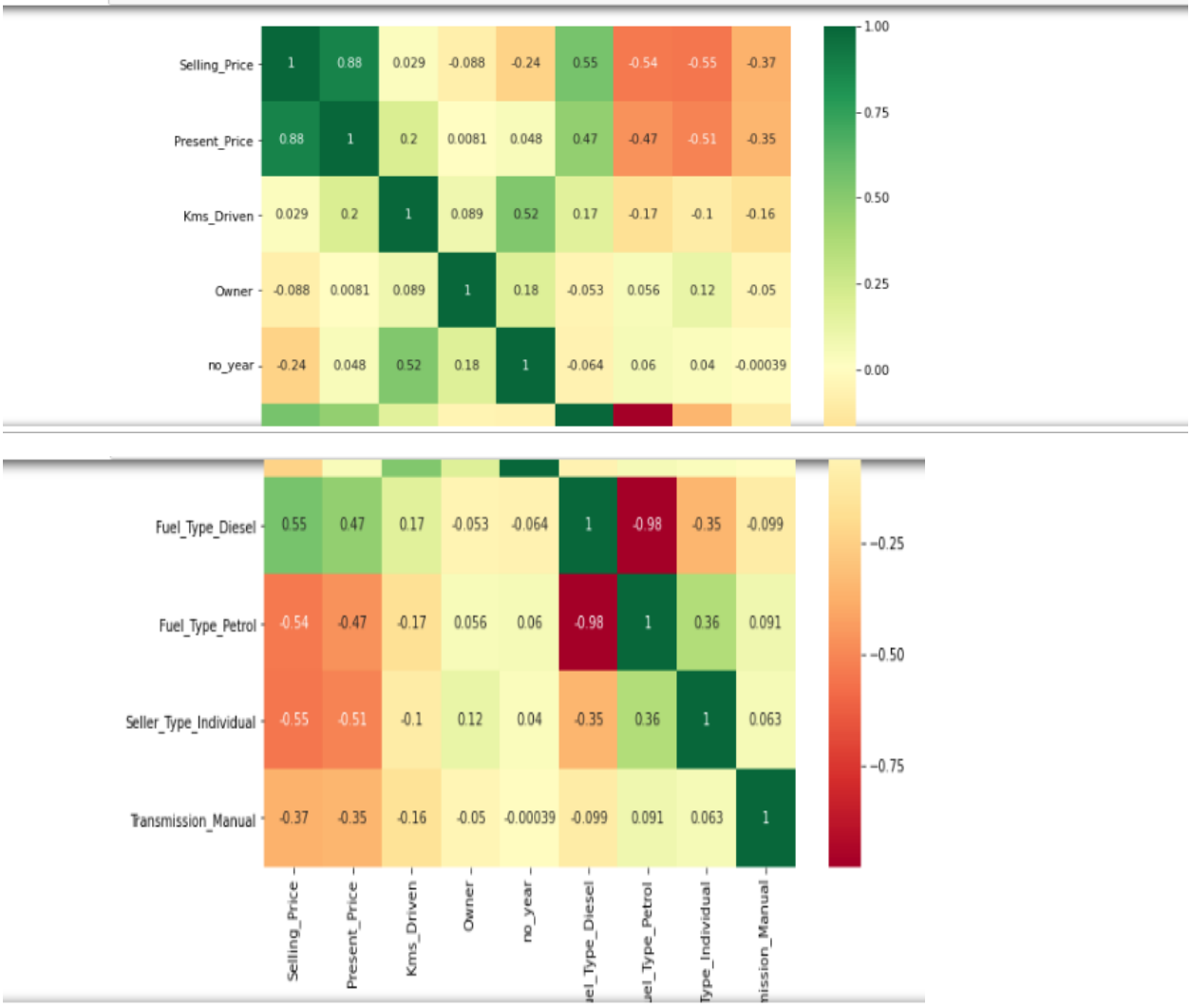
```
In [22]: sns.pairplot(final_dataset)
Out[22]: <seaborn.axisgrid.PairGrid at 0xf3872245c8>
```



Next step is plot it in heat map

A **heatmap** is a two-dimensional graphical representation of data where the individual values that are contained in a matrix are represented as colors. The seaborn **python** package allows the creation of annotated heatmaps which can be tweaked using Matplotlib tools as per the requirement.

```
In [24]: corrmat=final_dataset.corr()
         top_corr_features = corrmat.index
         plt.figure(figsize=(10,10))
         g=sns.heatmap(final_dataset[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



Then we allocate all our independent feature

X = final_dataset.iloc[:,1:] these are final_dataset=df[[Present_Price, Kms_Driven, Fuel_Type, Seller_Type, Transmission, Owner, Current, Year no_year]]

{1: is representing column starting form Present_Price}

our dependent feature is selling price

y = final_dataset.iloc[:,0]

Feature Importance and Doing a train test split

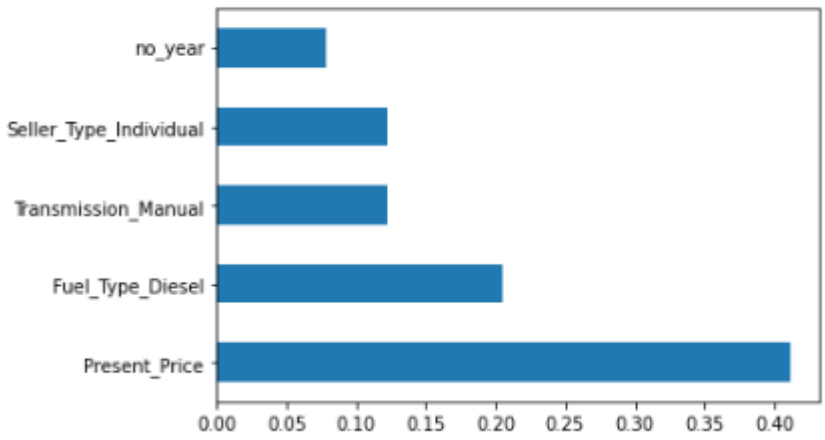
As it is regression problem we are using ExtraTreesRegressor to find important feature

```
ExtraTreesRegressor()
```

```
print(model.feature_importances_)
```

```
[4.11946112e-01 4.15172773e-02 3.99537471e-04 7.78993520e-02
 2.04928051e-01 1.89998368e-02 1.21892158e-01 1.22417674e-01]
```

```
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



We are taking all feature but only top 5 is shown in chart.

Algorithm

Random forest algorithm

Random forest is a flexible, easy to use machine learning algorithm that produces, even without hyper-parameter tuning, a great result most of the time. It is also one of the most used algorithms, because of its simplicity and diversity (it can be used for both classification and regression tasks). In this post we'll learn how the random forest algorithm works, how it differs from other algorithms and how to use it.

HOW RANDOM FOREST WORKS

Random forest is a supervised learning algorithm. The "forest" it builds, is an ensemble of decision trees, usually trained with the “bagging” method. The general idea of the bagging method is that a combination of learning models increases the overall result.

Put simply: random forest builds multiple decision trees and merges them together to get a more accurate and stable prediction.

One big advantage of random forest is that it can be used for both classification and regression problems, which form the majority of current machine learning systems. Let's look at random forest in classification, since classification is sometimes considered the building block of machine learning

```
from sklearn.model_selection import RandomizedSearchCV
```

```
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': ['auto', 'sqrt'], 'max_dept
h': [5, 10, 15, 20, 25, 30], 'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

```
rf = RandomForestRegressor()
```

```
rf_random = RandomizedSearchCV(estimator = rf, param_distributions = random_grid,scoring='neg_mean_squared_error', n_iter = 10, c
```

```
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=   1.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.8s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.9s
[CV] END max_depth=10, max_features=sqrt, min_samples_leaf=5, min_samples_split=5, n_estimators=900; total time=   1.8s
```

Performing hyper parameter tuning using RandomizedSearchCV

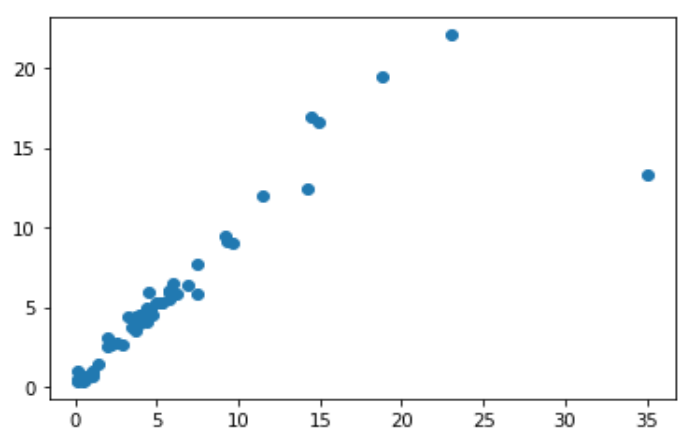
RandomizedSearchCV helps to find best parameter form n_estimators

Next step if fitting

Prediction

predictions=rf_random.predict(X_test)

```
In [45]: plt.scatter(y_test,predictions)
Out[45]: <matplotlib.collections.PathCollection at 0xf38ea20288>
```



Plotting is linearly available which represent our prediction is efficient and better.

Deployment

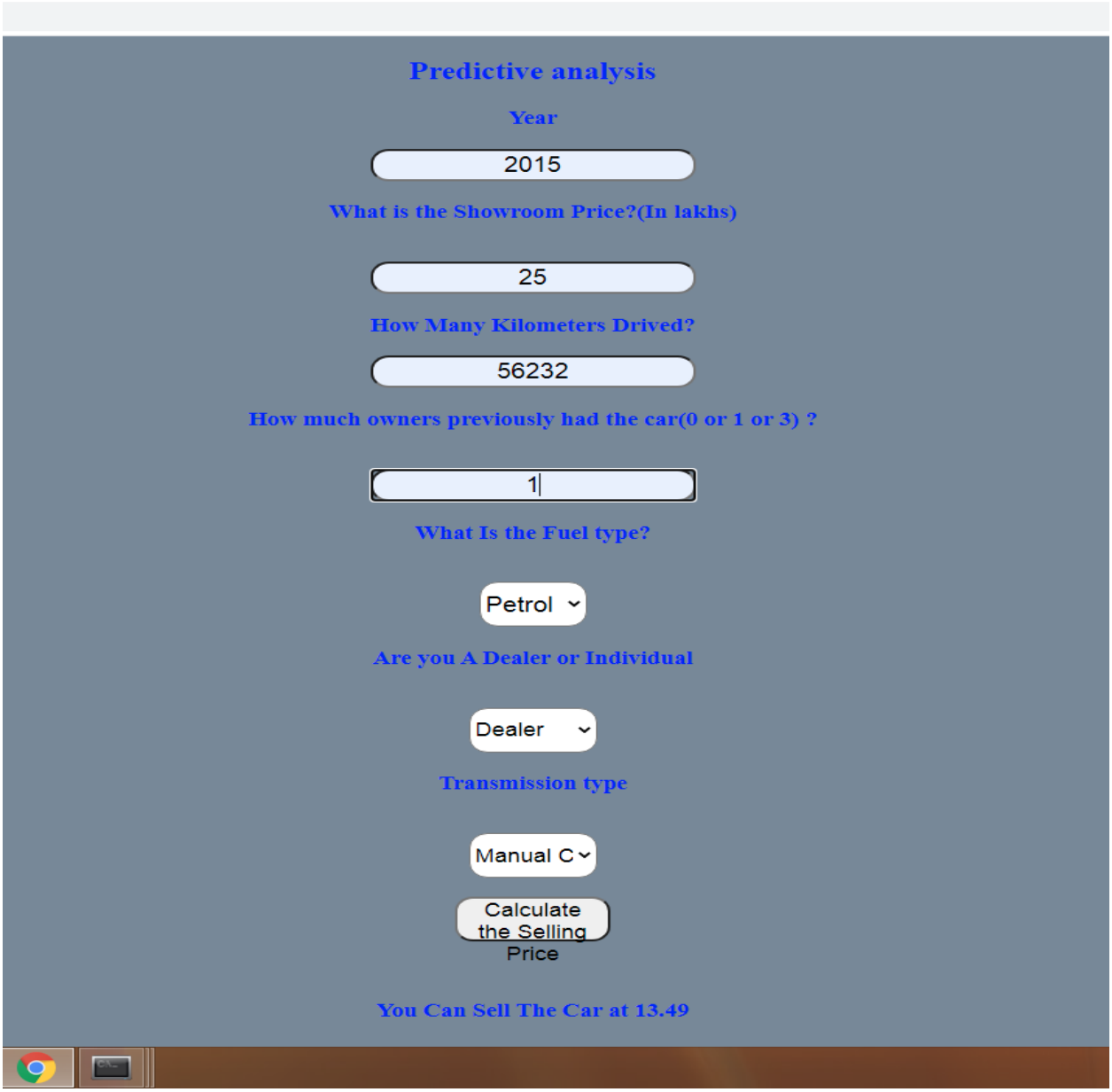
For frontend app.py is used. Using flask micro web framework we rendered index.html our model random_forest_regression_model.pkl we loaded the pickle file to get prediction value

```python
from flask import Flask, render_template, request
import jsonify
import requests
import pickle
import numpy as np
import sklearn
from sklearn.preprocessing import StandardScaler
app = Flask(__name__)
model = pickle.load(open('random_forest_regression_model.pkl', 'rb'))
@app.route('/',methods=['GET'])
def Home():
    return render_template('index.html')


standard_to = StandardScaler()
@app.route("/predict", methods=['POST'])
def predict():
    Fuel_Type_Diesel=0
    if request.method == 'POST':
        Year = int(request.form['Year'])
        Present_Price=float(request.form['Present_Price'])
        Kms_Driven=int(request.form['Kms_Driven'])
        Kms_Driven2=np.log(Kms_Driven)
        Owner=int(request.form['Owner'])
        Fuel_Type_Petrol=request.form['Fuel_Type_Petrol']
        if(Fuel_Type_Petrol=='Petrol'):
                Fuel_Type_Petrol=1
                Fuel_Type_Diesel=0
        else:
            Fuel_Type_Petrol=0
            Fuel_Type_Diesel=1
        Year=2021-Year
        Seller_Type_Individual=request.form['Seller_Type_Individual']
        if(Seller_Type_Individual=='Individual'):
            Seller_Type_Individual=1
        else:
            Seller_Type_Individual=0
        Transmission_Mannual=request.form['Transmission_Mannual']
        if(Transmission_Mannual=='Mannual'):
            Transmission_Mannual=1
        else:
            Transmission_Mannual=0
        prediction=model.predict([[Present_Price,Kms_Driven2,Owner,Year,Fuel_Type_Diesel,Fuel_Type_Petrol,Seller_Type_Individual,Transmission_Mannual]])
        output=round(prediction[0],2)
        if output<0:
            return render_template('index.html',prediction_texts="Sorry you cannot sell this car")
        else:
            return render_template('index.html',prediction_text="You Can Sell The Car at {}".format(output))
    else:
        return render_template('index.html')

if __name__=="__main__":
    app.run(debug=True)
```

Output

As we used flask we are getting web output we run app.py in http://127.0.0.1:5000/

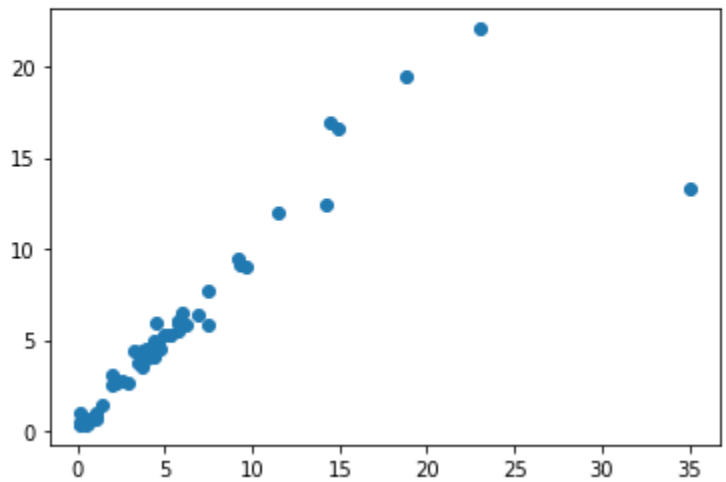
Show in output after giving input.



Performance

Plotting Cross-Validated Predictions for an un-aggregated model, an un-aggregated (i.e. the usual) cross validation can be used as approximation for predictive performance/generalization error estimate

```
In [45]: plt.scatter(y_test,predictions)

Out[45]: <matplotlib.collections.PathCollection at 0xf38ea20288>
```



In the x axis we have measured data and y axis we have prediction

Line is liner indicate good accuracy of model.

Discussion: there are many external factor in measuring the value of car such as engine hp, torque produce, condition etc. can play an important role for determining the price of a used car and can play an important role in feature importance if they were included in csv.

Reference:

1.Vehicle dataset.Used Cars data form websites(https://www.kaggle.com/vehicle-dataset-from-cardekho)