CAR SELLING PRICE PREDICTION APP

A Project Report

Submitted by

Mr. Balram Thakur

Roll No. 180935106037

in partial fulfillment for the award of the degree

Of

(BACHELOR OF COMPUTER APPLICATION)

UNDER THE GUIDANCE OF

Mr. Gaurav Kaushik

ASST. PROFESSOR-IT

AT

INSTITUTE OF MANAGEMENT STUDIES GHAZIABAD

UNIVERSITY COURSES CAMPUS, NH 24, ADHYATAMIK NAGAR,
GHAZIABAD (U.P)

(2018-2021)

TABLE OF CONTENTS

ACKNOWLEDGEMENT

It gives me a great of pleasure to present the report of the BCA project undertaken during BCA final year. I owe special debt of gratitude to Mr. Gaurav Kaushik, assistant professor of BCA department, institute of management studies, Ghaziabad for his constant support and guidance throughout the course of my work. His sincerity, thoroughness and perseverance have been a constant source of inspiration for me. It is only his efforts that my endeavors have seen light of the day.

I also take the opportunity to acknowledge the contribution of DR. GAGAN VARSHNEY head of department of bachelor of computer application, institute of management studies, Ghaziabad for his full support and assistance during the development of the project.

I also do not like to miss the opportunity to acknowledge the contribution of all faculty members of the department for their kind assistance and cooperation during the development of my project.

Balram thakur

INTRODUCTION

Car selling price prediction system is a portable application which is executed in Android, iOS and Windows platform. This is an Android based application that is developed so that ,suppose you have a car and you want to sell it so this application will help you to know the price of your car so you will not get loss even if you want to buy any second hand car then you can know the price of car that what should be in actual.

Car selling price prediction system is a software developed for everyone and for all operating system

The purpose of Car selling price prediction system is to computerized the tradition way of selling or buying any second-hand car

This application was particularly developed by using previous data from Car Dekho website and based on practical and logical approach not imagination.

This application is totally based on data from more than 500 users from Car Dekho and totally based on Machine learning and Artificial intelligence. This application comes under data science area

OBJECTIVE

- Easy to know the price of car if you want to sell
- Avoid sell loss
- Help you to know the price if you want to purchase any second-hand car from anywhere
- Will help you to get good amount of your car if you want to sell

TOOLS /ENVIRONMENT USED

OS Requirement

- Android, IOS and Windows

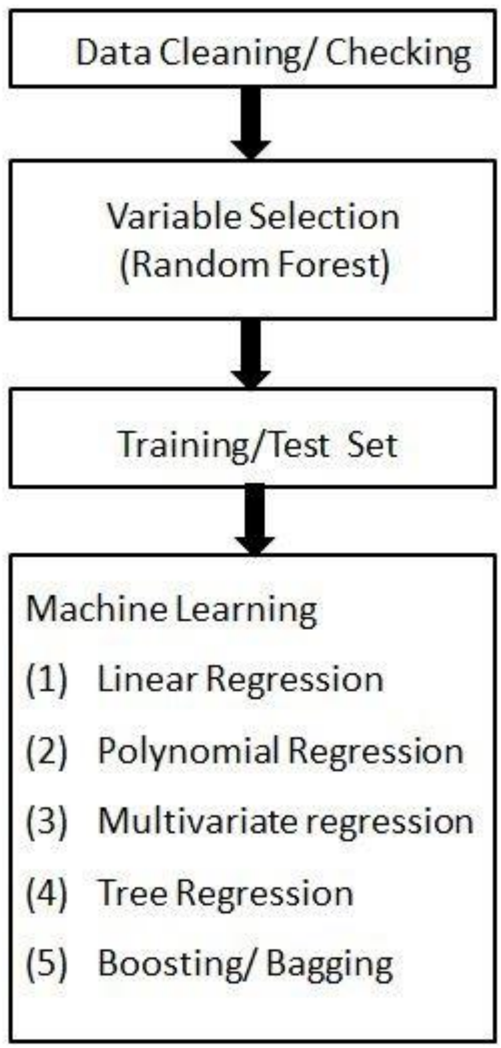
Tools Used to Develop

- Anaconda

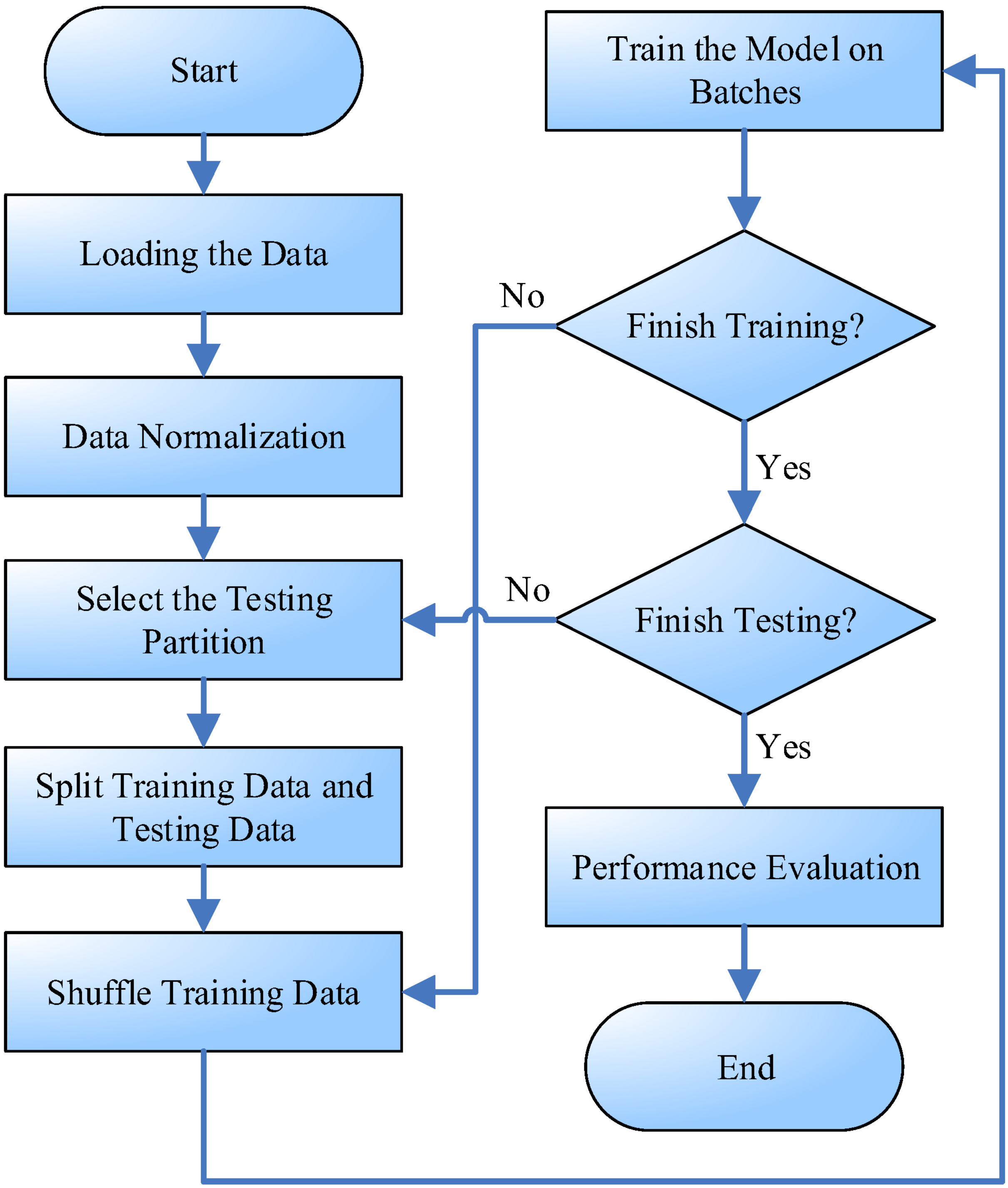
Languages Used for Development

- HTML
- JAVA
- CSS
- Python
- Flask

Data Flow Diagram (Level 0)

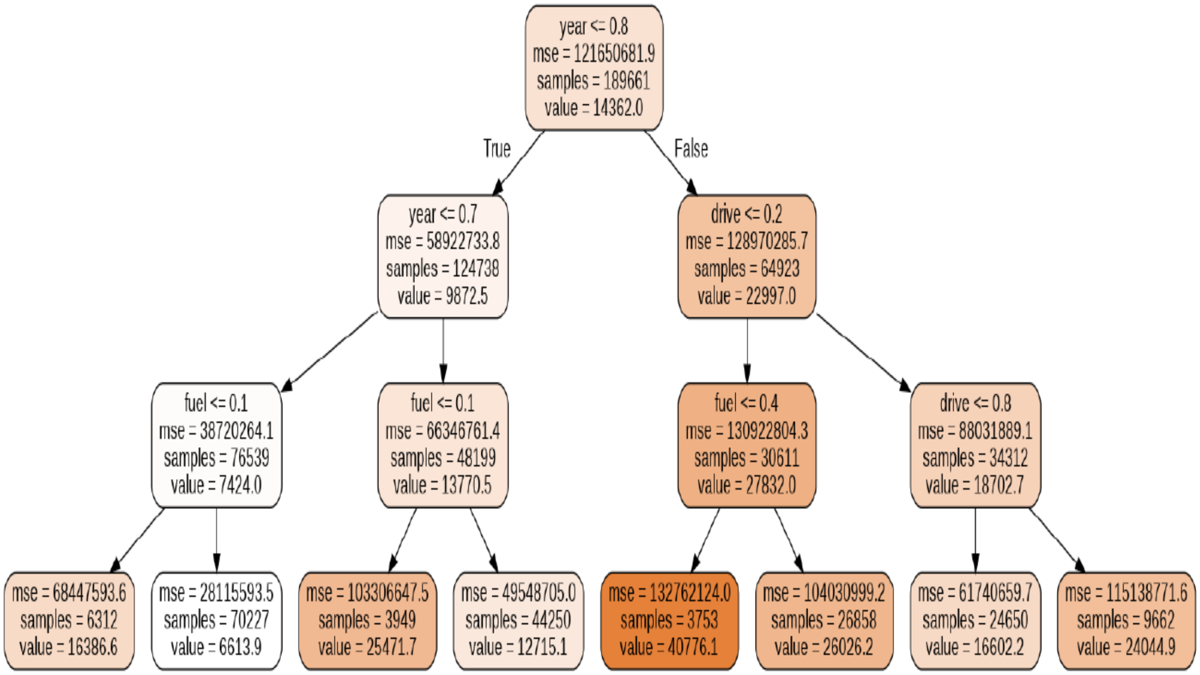


FLOWCHART

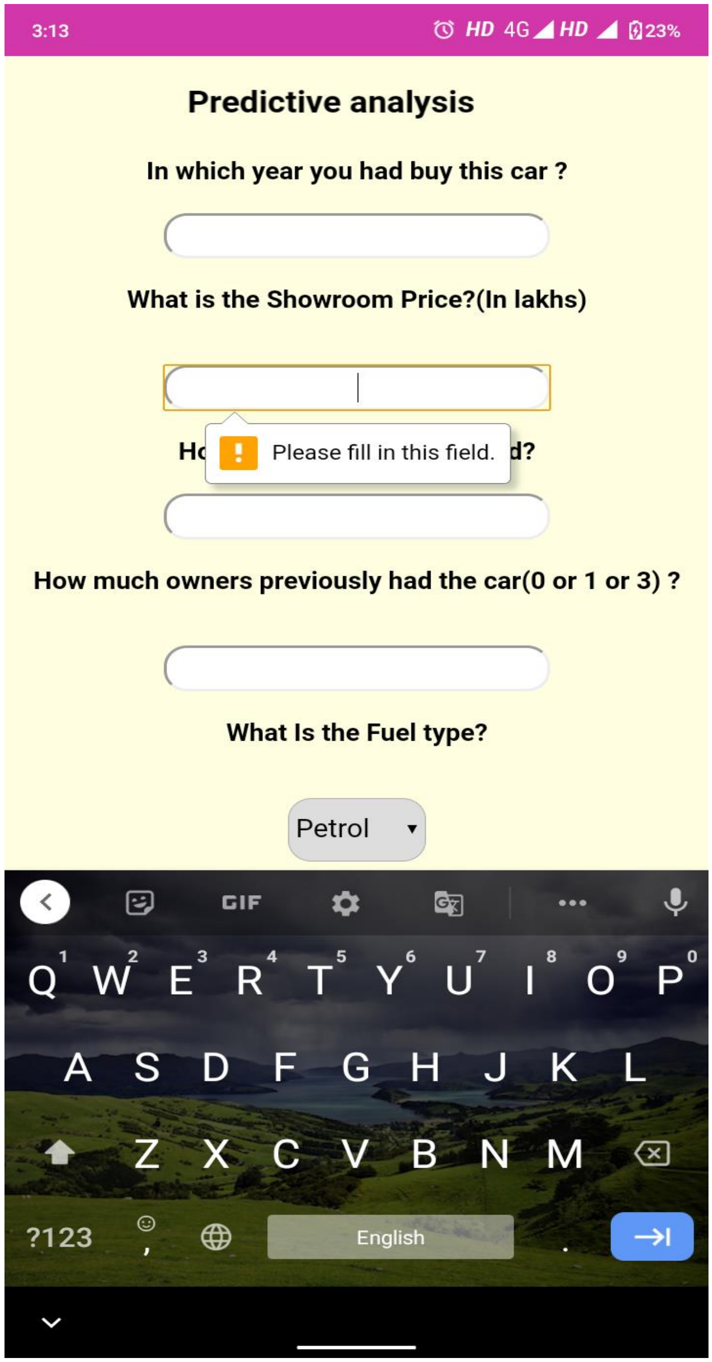


Start
Loading the Data
Data Normalization
Select the Testing Partition
Split Training Data and Testing Data
Shuffle Training Data
Train the Model on Batches
Finish Training?
No
Yes
Finish Testing?
No
Yes
Performance Evaluation
End

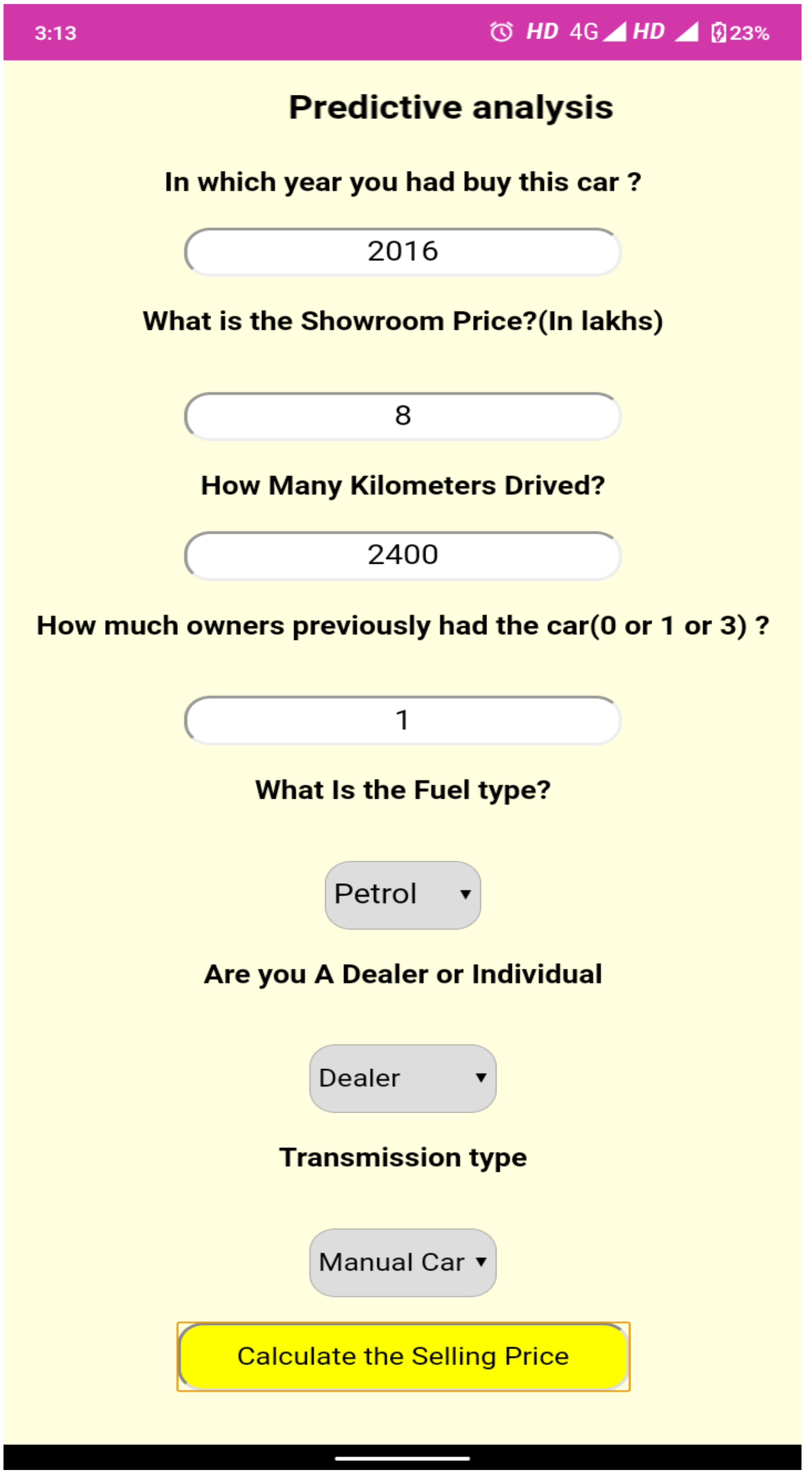
Data Flow Diagram (Level 1)



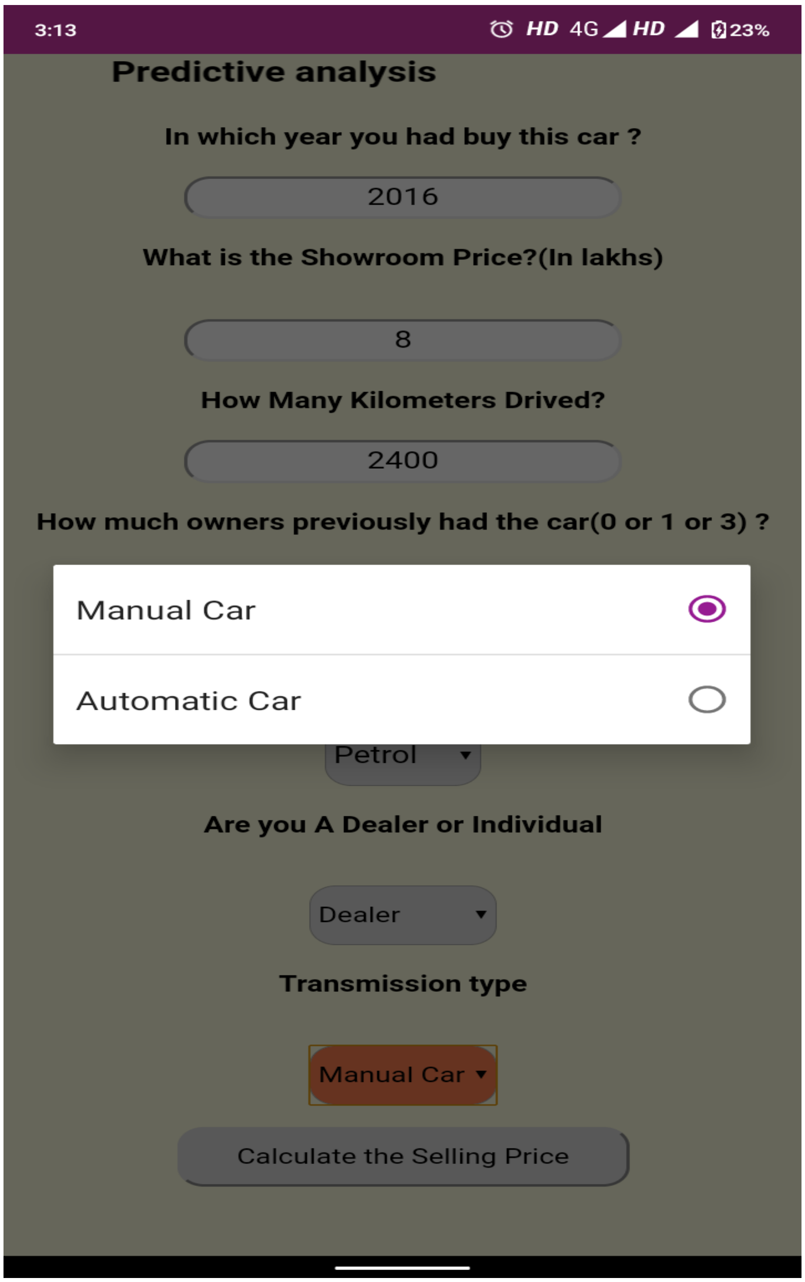
SCREENSHOTS



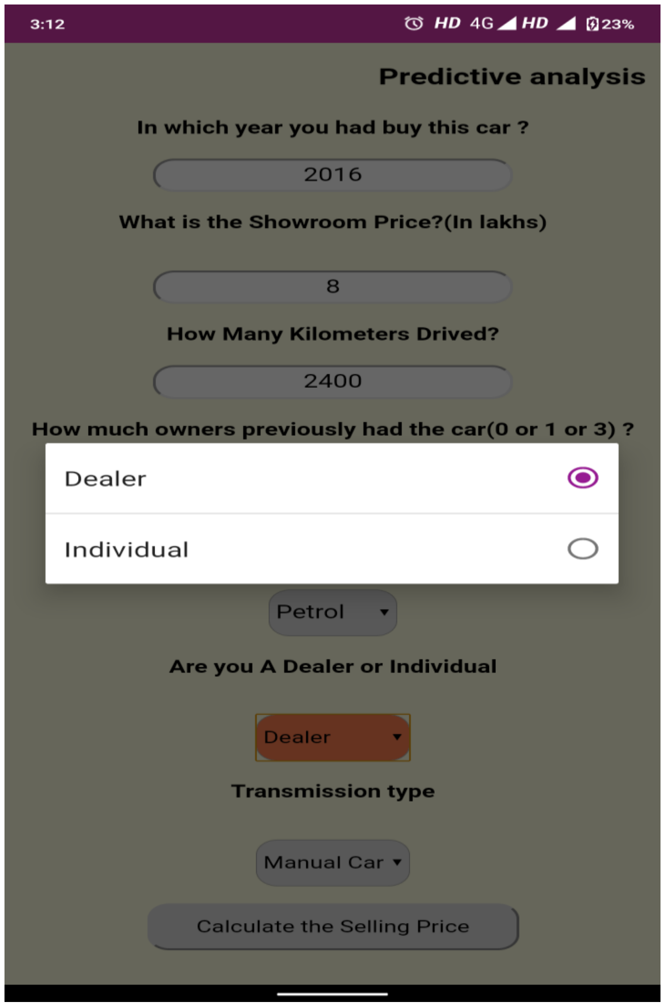
SCREENSHOTS



SCREENSHOTS

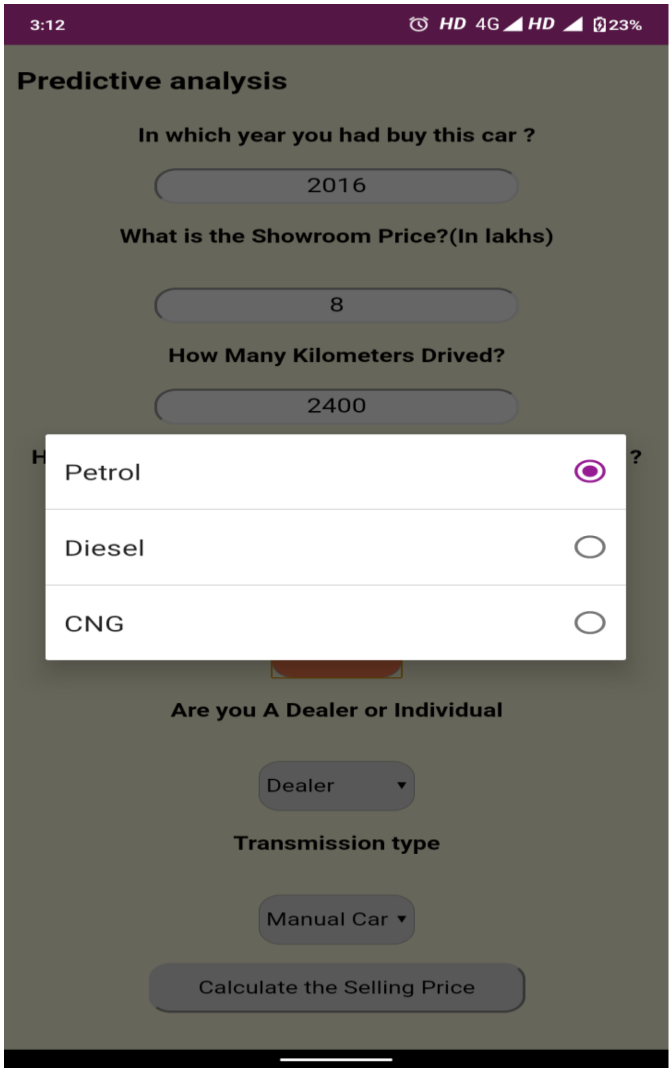


SCREENSHOT

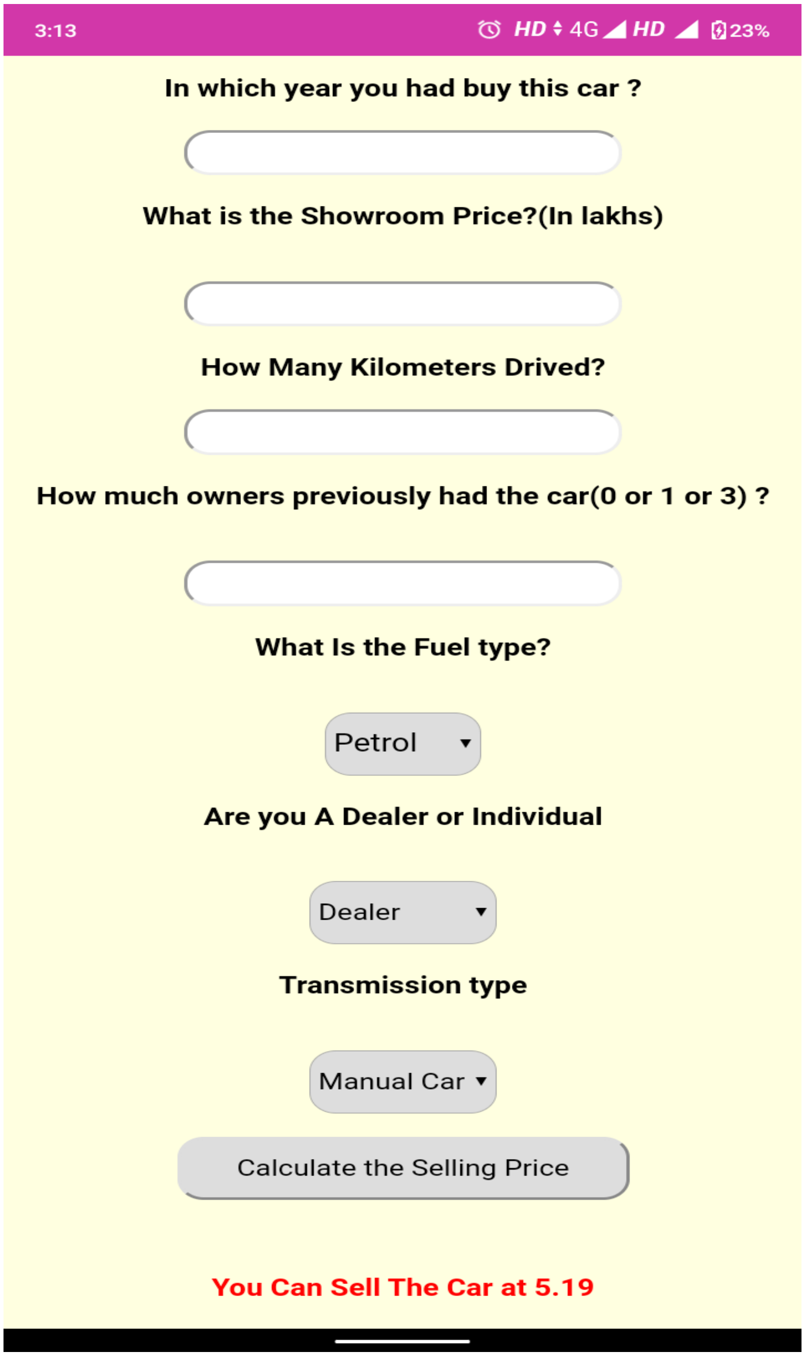


3:12
HD 4G HD 23%
Predictive analysis
In which year you had buy this car ?
2016
What is the Showroom Price?(In lakhs)
8
How Many Kilometers Drived?
2400
How much owners previously had the car(0 or 1 or 3) ?
Dealer
Individual
Petrol
Are you A Dealer or Individual
Dealer
Transmission type
Manual Car
Calculate the Selling Price

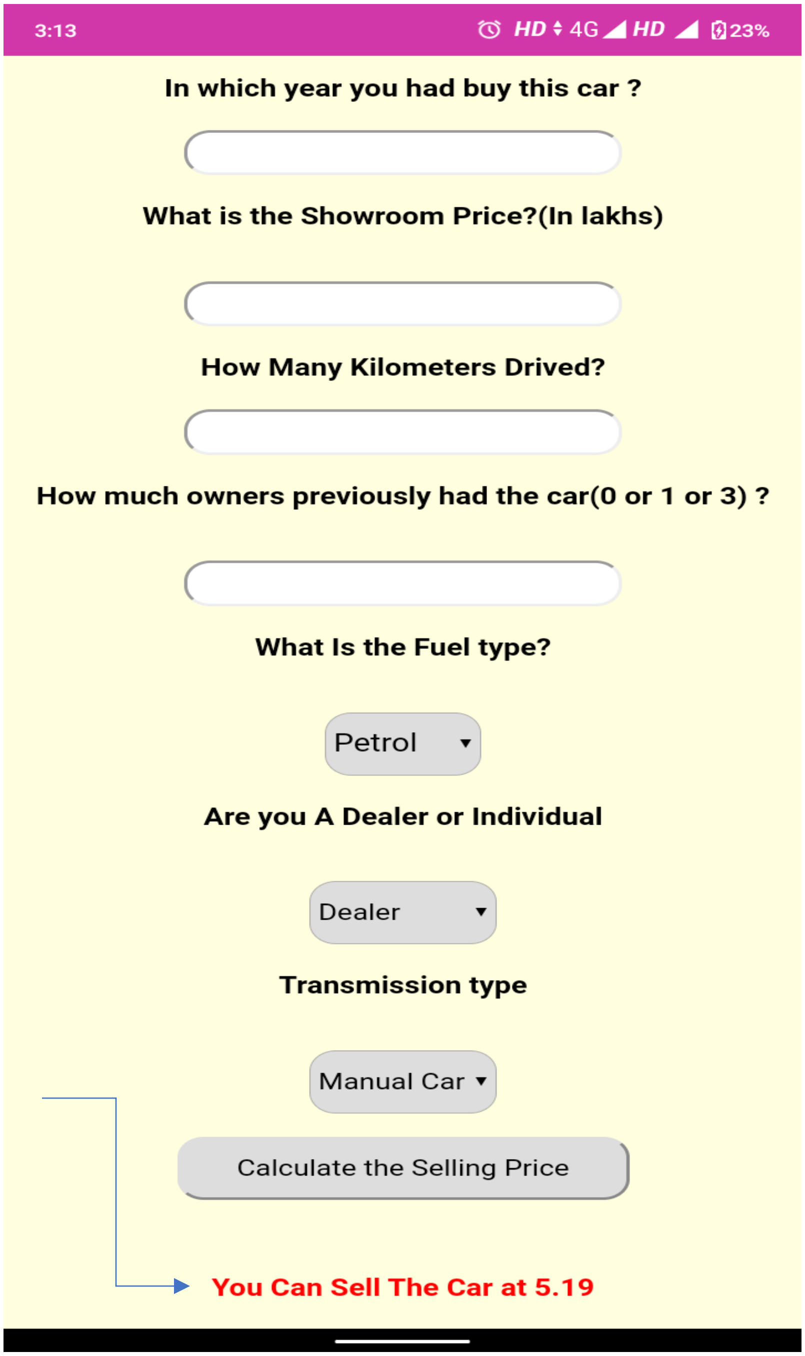
SCREENSHOTS



SCREENSHOT

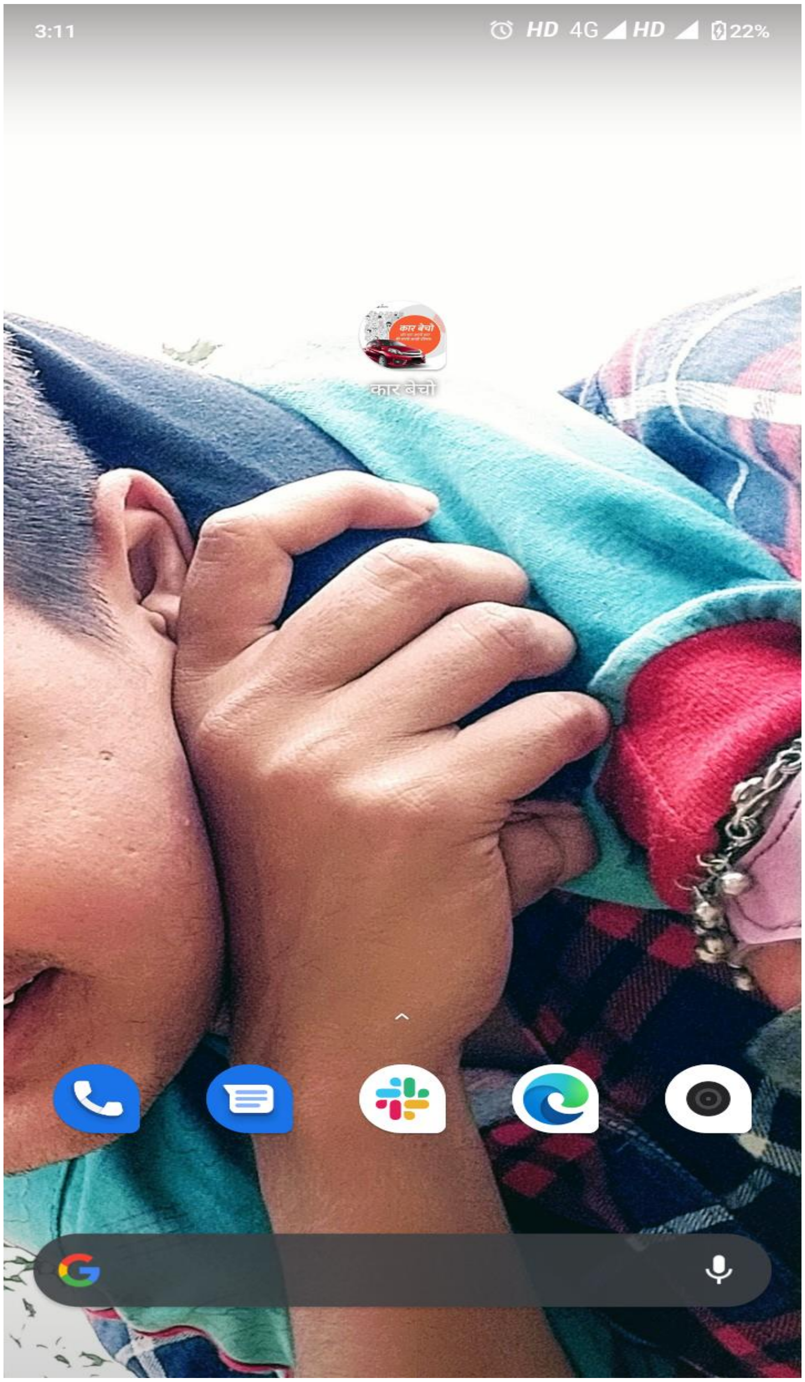

3:13
HD 4G HD 23%
In which year you had buy this car ?
What is the Showroom Price?(In lakhs)
How Many Kilometers Drived?
How much owners previously had the car(0 or 1 or 3) ?
What Is the Fuel type?
Petrol
Are you A Dealer or Individual
Dealer
Transmission type
Manual Car
Calculate the Selling Price
You Can Sell The Car at 5.19

OUTPUT WILL BE LIKE BELOW SHOWN AND STILL UNDER DEVELOPMENT



You can see the output using this arrow above which is in Lakhs

APK GENERATED



COADING

Index.html

```
<!DOCTYPE
html>
          <html lang=”en”>

          <head>
              <meta charset=”UTF-8”>
              <meta name=”viewport” content=”width=device-width,
          initial-scale=1.0”>
              <title>Document</title>
          </head>

          <body>

              <div style=”color:black”>
                  <form action=”{{ url_for(‘predict’)}}”
          method=”post”>
                      <h2><marquee behavior=”alternate”
          scrollamount=”10”>Predictive analysis</marquee></h2>
                      <h3>In which year you had buy this car ?</h3>
                      <input id=”first” name=”Year” type=”number “>
                      <h3>What is the Showroom Price?(In
          lakhs)</h3><br><input id=”second” name=”Present_Price”
          required=”required”>
                      <h3>How Many Kilometers Drived?</h3><input
          id=”third” name=”Kms_Driven” required=”required”>
                      <h3>How much owners previously had the car(0
          or 1 or 3) ?</h3><br><input id=”fourth” name=”Owner”
          required=”required”>
                      <h3>What Is the Fuel type?</h3><br><select
          name=”Fuel_Type_Petrol” id=”fuel” required=”required”>
                          <option value=”Petrol”>Petrol</option>
                          <option value=”Diesel”>Diesel</option>
                          <option value=”CNG”>CNG</option>
                      </select>
```

```
            <h3>Are you A Dealer or
Individual</h3><br><select name=”Seller_Type_Individual”
id=”resea” required=”required”>
                <option value=”Dealer”>Dealer</option>
                <option
value=”Individual”>Individual</option>
            </select>
            <h3>Transmission type</h3><br><select
name=”Transmission_Mannual” id=”research”
required=”required”>
                <option value=”Mannual”>Manual
Car</option>
                <option value=”Automatic”>Automatic
Car</option>
            </select>
            <br><br><button  id=”sub”>Calculate the
Selling Price</button>
            <br>




        </form>



        <br><br><h3><font color=”red”>
{{prediction_text}}</font><h3>

    </div>





    <style>
        body {
            background-color: lightyellow;
            text-align: center;
            padding: 0px;
        }


        #research {
            font-size: 18px;
```

```css
    width: 100px;
    height: 23px;
    top: 23px;
}

#box {
    border-radius: 60px;
    border-color: 45px;
    border-style: solid;
    font-family: cursive;
    text-align: center;
    background-color: rgb(168, 131, 61);
    font-size: medium;
    position: absolute;
    width: 700px;
    bottom: 9%;
    height: 850px;
    right: 30%;
    padding: 0px;
    margin: 0px;
    font-size: 14px;
}

#fuel {
    width: 83px;
    height: 43px;
    text-align: center;
    border-radius: 14px;
    font-size: 20px;
}

#fuel:hover {
    background-color: coral;
}

#research {
    width: 99px;
    height: 43px;
    text-align: center;
    border-radius: 14px;
    font-size: 18px;
}
```

```css
#research:hover {
    background-color: coral;
}

#resea {
    width: 99px;
    height: 43px;
    text-align: center;
    border-radius: 14px;
    font-size: 18px;
}

#resea:hover {
    background-color: coral;
}

#sub {
    width: 240px;
    height: 43px;
    text-align: center;
    border-radius: 14px;
    font-size: 18px;
}

#sub:hover {
    background-color: yellow;
}

#first {
    border-radius: 14px;
    height: 25px;
    font-size: 20px;
    text-align: center;
}

#second {
    border-radius: 14px;
    height: 25px;
    font-size: 20px;
    text-align: center;
}

#third {
```

```
            border-radius: 14px;
            height: 25px;
            font-size: 20px;
            text-align: center;
        }

        #fourth {
            border-radius: 14px;
            height: 25px;
            font-size: 20px;
            text-align: center;
        }
    </style>

</body>

</html>
```

app.py

```
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=2020-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)
```

main.py

```
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=2020-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_Man
nual]])
        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)
```

requirement.txt

eature==2020.6.20

chardet==3.0.4

click==7.1.2

Flask==1.1.2

idna==2.10

itsdangerous==1.1.0

Jinja2==2.11.2

joblib==0.15.1

jsonify==0.5

MarkupSafe==1.1.1

numpy==1.19.0

requests==2.24.0

scikit-learn==0.23.1

scipy==1.5.0

sklearn==0.0

threadpoolctl==2.1.0

urllib3==1.25.9

Werkzeug==1.0.1

wincertstore==0.2

gunicorn

Procfile

```
web: gunicorn app:app
```

prediction.ipynb

```
In [1]: import pandas as pd
In [2]: df=pd.read_csv('car data.csv')
In [3]: df.shape
Out[3]☹301, 9)
In [4]:print(df['Seller_Type'].unique())
       print(df['Fuel_Type'].unique())
       print(df['Transmission'].unique())
       print(df['Owner'].unique())
 Out[4]:['Dealer' 'Individual']
        ['Petrol' 'Diesel' 'CNG']
        ['Manual' 'Automatic']
        [0 1 3]
In [5]:##check missing values
        df.isnull().sum()
Out[5]:
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()
Out[7]:
In [8]:final_dataset=df[['Year','Selling_Price','Present_Price','Kms_Driven',
'Fuel_Type','Seller_Type','Transmission','Owner']]
In [9]:final_dataset.head()
Out[9]:

In [ ]:
In [10]:final_dataset['Current Year']=2020
In [11]:final_dataset.head()
Out[11]:
In [12]:final_dataset['no_year']=final_dataset['Current Year']-
final_dataset['Year']
In [13]:final_dataset.head()
Out[13]:
```

In [14]:

```
final_dataset=pd.get_dummies(final_dataset,drop_first=True)
```

In [15]:

```
final_dataset.head()
```

Out[15]:

In [16]:

```
final_dataset=final_dataset.drop(['Current Year'],axis=1)
```

In [17]:

```
final_dataset.head()
```

Out[17]:

In [18]:

```
final_dataset.corr()
```

Out[18]:

In [19]:

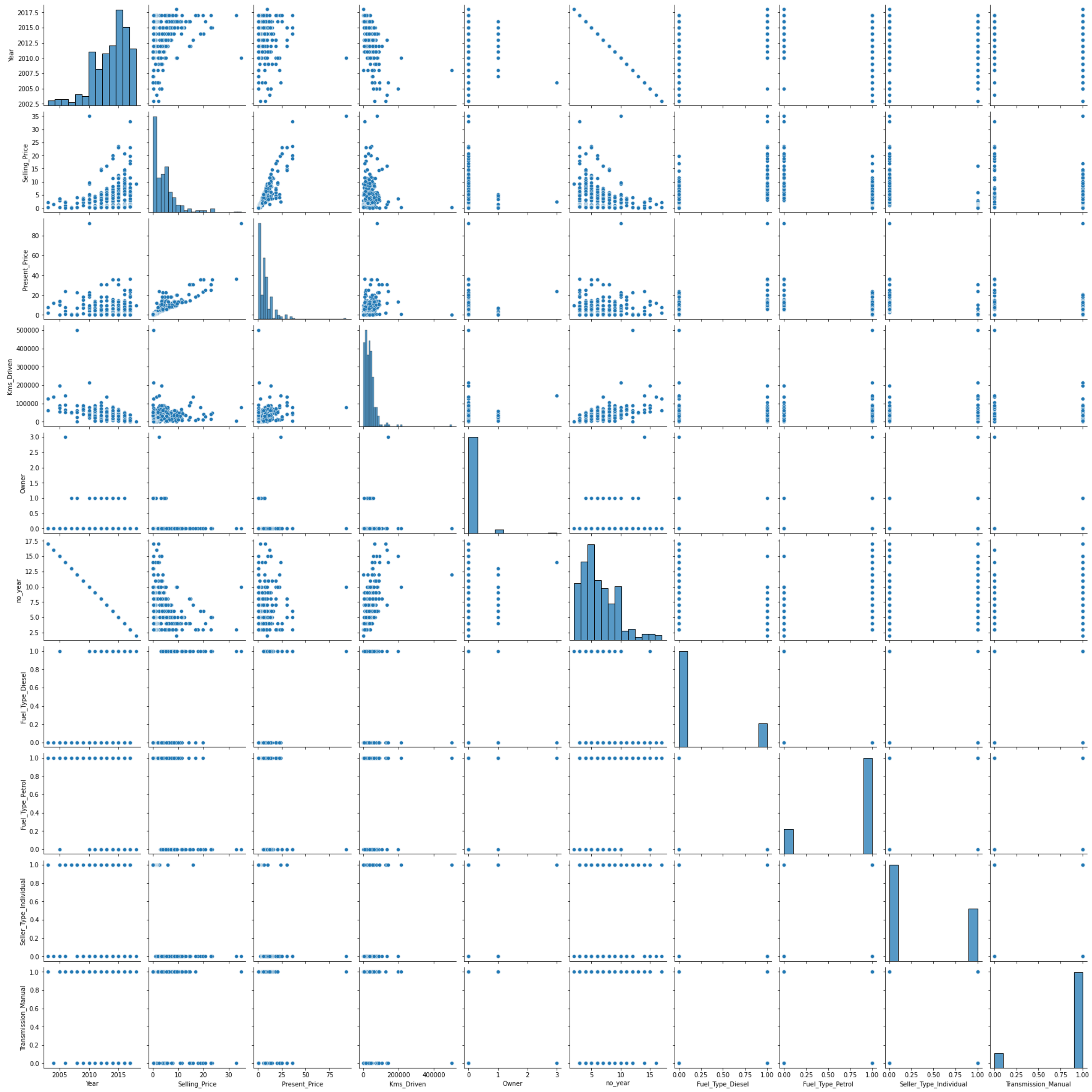
```
import seaborn as sns
```

In [20]:

```
sns.pairplot(final_dataset)
```

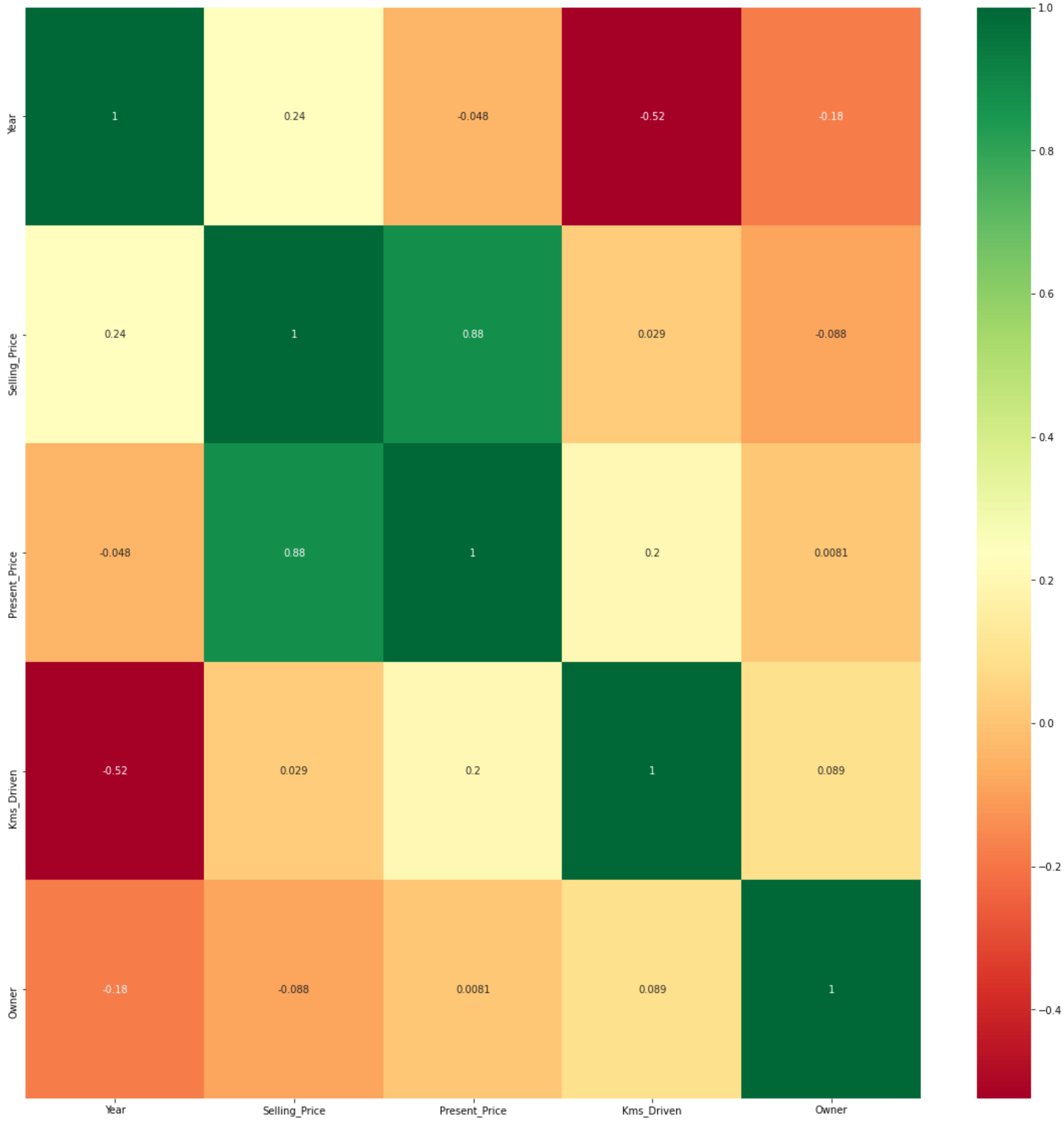
Out[20]:

```
<seaborn.axisgrid.PairGrid at 0x1deac35c108>
```



In [24]:

```
import seaborn as sns
import matplotlib.pyplot as plt
#get correlations of each features in dataset
corrmat = df.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(df[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



In [25]:

```
X=final_dataset.iloc[:,1:]
y=final_dataset.iloc[:,0]
```

In [26]:

```
X['Owner'].unique()
```

Out[26]:

```
array([0, 1, 3], dtype=int64)
```

In [27]:

```
X.head()
```

Out[27]:

In [28]:

```
y.head()
```

Out[28]:

```
   3    2014
   3    2013
2    2017
3    2011
4    2014
Name: Year, dtype: int64
```

In [30]:

```
### Feature Importance

from sklearn.ensemble import ExtraTreesRegressor
import matplotlib.pyplot as plt
model = ExtraTreesRegressor()
model.fit(X,y)
```

```
C:\Users\balram\Anaconda3\lib\site-packages\sklearn\ensemble\forest.py:245: F
utureWarning: The default value of n_estimators will change from 10 in versio
n 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

Out[30]:

```
ExtraTreesRegressor(bootstrap=False, criterion='mse', max_depth=None,
                    max_features='auto', max_leaf_nodes=None,
                    min_impurity_decrease=0.0, min_impurity_split=None,
                    min_samples_leaf=1, min_samples_split=2,
                    min_weight_fraction_leaf=0.0, n_estimators=10, n_jobs=Non
e,
                    oob_score=False, random_state=None, verbose=0,
                    warm_start=False)
```

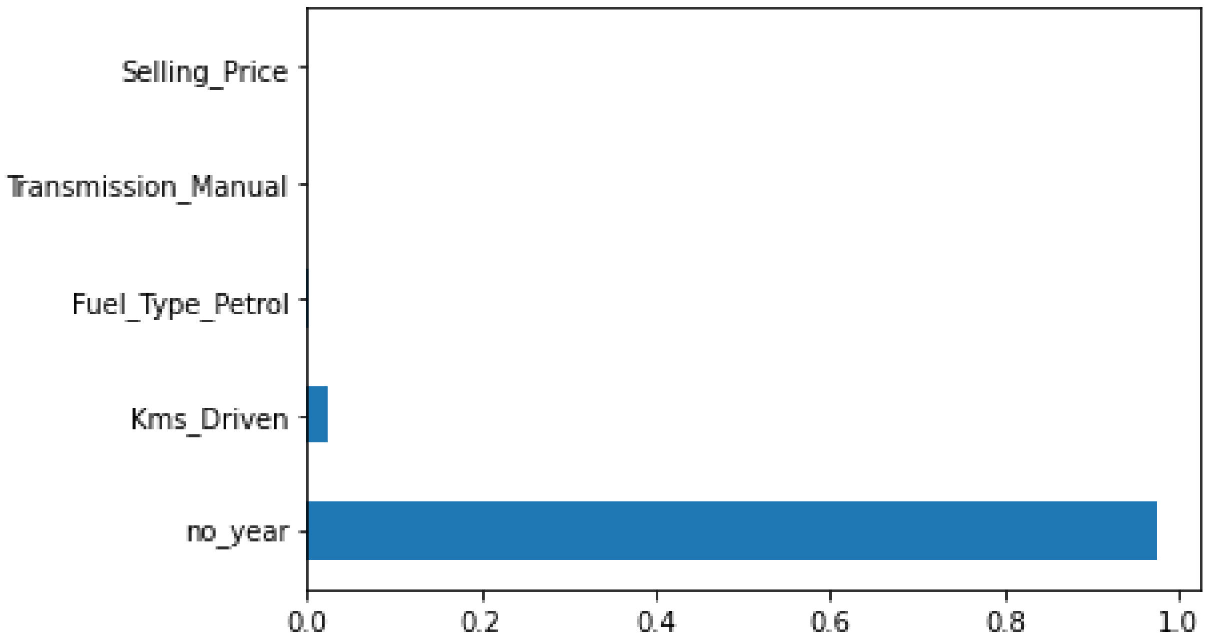
In [31]:

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

```
[3.98672328e-05 1.99336164e-05 2.32319105e-02 0.00000000e+00
 9.75260444e-01 0.00000000e+00 1.11694697e-03 2.65781553e-05
 3.04319878e-04]
```

In [32]:

```
#plot graph of feature importances for better visualization
feat_importances = pd.Series(model.feature_importances_, index=X.columns)
feat_importances.nlargest(5).plot(kind='barh')
plt.show()
```



In [33]:

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, ran
dom_state=0)
```

In [34]:

```
from sklearn.ensemble import RandomForestRegressor
```

In [35]:

```
regressor=RandomForestRegressor()
```

In [37]:

```
import numpy as np
n_estimators = [int(x) for x in np.linspace(start = 100, stop = 1200, num =
12)]
print(n_estimators)
```

```
[100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1100, 1200]
```

In [38]:

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

In [39]:

```
#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 [41]:

```
# 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, 12
00], 'max_features': ['auto', 'sqrt'], 'max_depth': [5, 10, 15, 20, 25, 30],
'min_samples_split': [2, 5, 10, 15, 100], 'min_samples_leaf': [1, 2, 5, 10]}
```

In [42]:

```
# Use the random grid to search for best hyperparameters
# First create the base model to tune
rf = RandomForestRegressor()
```

In [43]:

```
# 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, cv = 5, verbose=2, rando
m_state=42, n_jobs = 1)
```

In [44]:

```
rf_random.fit(X_train,y_train)
```

```
Fitting 5 folds for each of 10 candidates, totalling 50 fits
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=
sqrt, max_depth=10
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent worke
rs.
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features
=sqrt, max_depth=10, total=   3.5s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=
sqrt, max_depth=10
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    3.4s remaining:    0.
0s
```

```
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features
=sqrt, max_depth=10, total=   3.8s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=
sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features
=sqrt, max_depth=10, total=   4.3s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=
sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features
=sqrt, max_depth=10, total=   4.3s
[CV] n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features=
sqrt, max_depth=10
[CV]  n_estimators=900, min_samples_split=5, min_samples_leaf=5, max_features
=sqrt, max_depth=10, total=   3.1s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_featur
es=sqrt, max_depth=15, total=   3.9s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_featur
es=sqrt, max_depth=15, total=   3.8s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_featur
es=sqrt, max_depth=15, total=   3.2s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_featur
es=sqrt, max_depth=15, total=   3.6s
[CV] n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=15
[CV]  n_estimators=1100, min_samples_split=10, min_samples_leaf=2, max_featur
es=sqrt, max_depth=15, total=   4.1s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feature
s=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_featur
es=auto, max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feature
s=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_featur
es=auto, max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feature
s=auto, max_depth=15
```

```
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_featur
es=auto, max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feature
s=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_featur
es=auto, max_depth=15, total=   1.0s
[CV] n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_feature
s=auto, max_depth=15
[CV]  n_estimators=300, min_samples_split=100, min_samples_leaf=5, max_featur
es=auto, max_depth=15, total=   1.0s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=
auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features
=auto, max_depth=15, total=   1.3s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=
auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features
=auto, max_depth=15, total=   1.4s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=
auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features
=auto, max_depth=15, total=   1.5s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=
auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features
=auto, max_depth=15, total=   1.7s
[CV] n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features=
auto, max_depth=15
[CV]  n_estimators=400, min_samples_split=5, min_samples_leaf=5, max_features
=auto, max_depth=15, total=   1.5s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feature
s=auto, max_depth=20, total=   2.6s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feature
s=auto, max_depth=20, total=   1.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feature
s=auto, max_depth=20, total=   1.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features
=auto, max_depth=20
```

```
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feature
s=auto, max_depth=20, total=   1.9s
[CV] n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=5, min_samples_leaf=10, max_feature
s=auto, max_depth=20, total=   1.9s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features
=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feature
s=sqrt, max_depth=25, total=   2.8s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features
=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feature
s=sqrt, max_depth=25, total=   3.4s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features
=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feature
s=sqrt, max_depth=25, total=   3.7s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features
=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feature
s=sqrt, max_depth=25, total=   3.3s
[CV] n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_features
=sqrt, max_depth=25
[CV]  n_estimators=1000, min_samples_split=2, min_samples_leaf=1, max_feature
s=sqrt, max_depth=25, total=   3.6s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featur
es=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featu
res=sqrt, max_depth=5, total=   3.3s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featur
es=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featu
res=sqrt, max_depth=5, total=   4.1s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featur
es=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featu
res=sqrt, max_depth=5, total=   4.9s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featur
es=sqrt, max_depth=5
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featu
res=sqrt, max_depth=5, total=   3.7s
[CV] n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featur
es=sqrt, max_depth=5
```

```
[CV]  n_estimators=1100, min_samples_split=15, min_samples_leaf=10, max_featu
res=sqrt, max_depth=5, total=   3.7s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features
=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feature
s=sqrt, max_depth=15, total=   1.2s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features
=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feature
s=sqrt, max_depth=15, total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features
=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feature
s=sqrt, max_depth=15, total=   0.8s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features
=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feature
s=sqrt, max_depth=15, total=   0.9s
[CV] n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_features
=sqrt, max_depth=15
[CV]  n_estimators=300, min_samples_split=15, min_samples_leaf=1, max_feature
s=sqrt, max_depth=15, total=   2.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features
=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=5, total=   4.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features
=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=5, total=   2.5s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features
=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=5, total=   3.2s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features
=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=5, total=   4.1s
[CV] n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_features
=sqrt, max_depth=5
[CV]  n_estimators=700, min_samples_split=10, min_samples_leaf=2, max_feature
s=sqrt, max_depth=5, total=   3.2s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features
=auto, max_depth=20
```

```
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feature
s=auto, max_depth=20, total=   3.7s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feature
s=auto, max_depth=20, total=   3.5s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feature
s=auto, max_depth=20, total=   3.6s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feature
s=auto, max_depth=20, total=   3.5s
[CV] n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_features
=auto, max_depth=20
[CV]  n_estimators=700, min_samples_split=15, min_samples_leaf=1, max_feature
s=auto, max_depth=20, total=   3.8s
[Parallel(n_jobs=1)]: Done  50 out of  50 | elapsed:  2.3min finished
Out[44]:
RandomizedSearchCV(cv=5, error_score='raise-deprecating',
                   estimator=RandomForestRegressor(bootstrap=True,
                                                   criterion='mse',
                                                   max_depth=None,
                                                   max_features='auto',
                                                   max_leaf_nodes=None,
                                                   min_impurity_decrease=0.0,
                                                   min_impurity_split=None,
                                                   min_samples_leaf=1,
                                                   min_samples_split=2,
                                                   min_weight_fraction_leaf=0
.0,
                                                   n_estimators='warn',
                                                   n_jobs=None, oob_score=Fal
se,
                                                   random_sta…
                   iid='warn', n_iter=10, n_jobs=1,
                   param_distributions={'max_depth': [5, 10, 15, 20, 25, 30],
                                        'max_features': ['auto', 'sqrt'],
                                        'min_samples_leaf': [1, 2, 5, 10],
                                        'min_samples_split': [2, 5, 10, 15,
                                                              100],
                                        'n_estimators': [100, 200, 300, 400,
                                                         500, 600, 700, 800,
```

```
                                                       900, 1000, 1100,
                                                       1200]},
                   pre_dispatch='2*n_jobs', random_state=42, refit=True,
                   return_train_score=False, scoring='neg_mean_squared_error'
,
                   verbose=2)
```

In [45]:

```
rf_random.best_params_
```

Out[45]:

```
{'n_estimators': 400,
 'min_samples_split': 5,
 'min_samples_leaf': 5,
 'max_features': 'auto',
 'max_depth': 15}
```

In [46]:

```
rf_random.best_score_
```

Out[46]:

```
-0.11307985565758434
```

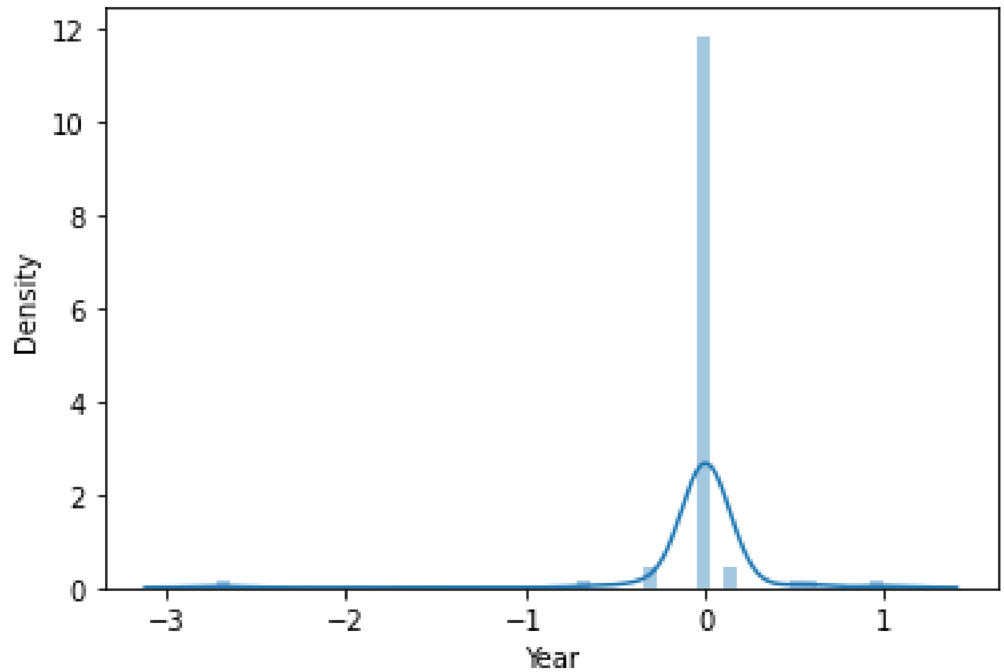
In [47]:

```
predictions=rf_random.predict(X_test)
```

In [48]:

```
sns.distplot(y_test-predictions);
```

```
C:\Users\balram\Anaconda3\lib\site-packages\seaborn\distributions.py:2551: Fu
tureWarning: `distplot` is a deprecated function and will be removed in a fut
ure version. Please adapt your code to use either `displot` (a figure-level f
unction with similar flexibility) or `histplot` (an axes-level function for h
istograms).
  Warnings.warn(msg, FutureWarning)
```

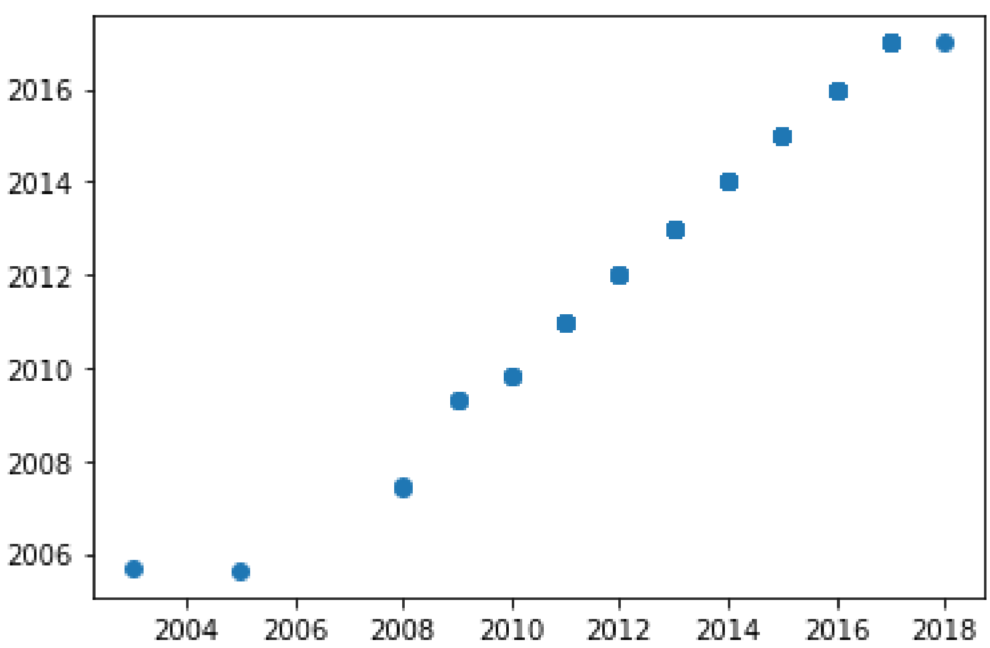


In [49]:

```
plt.scatter(y_test,predictions)
```

Out[49]:

<matplotlib.collections.PathCollection at 0x1deb8828448>



In [50]:

```
from sklearn import metrics
```

In [51]:

```
print('MAE:', metrics.mean_absolute_error(y_test, predictions))
```

```python
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 0.07763673725398203
MSE: 0.10766318255952667
RMSE: 0.3281206829194506
```

In [52]:

```python
import pickle
# open a file, where you ant to store the data
file = open('random_forest_regression_model.pkl', 'wb')

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

LIMITATIONS/ SCOPE

- Scope of this project is to the Device in which it has been installed.
- You can also use this project on any system without any installation using web link: - http://carpricebalram.herokuapp.com/
- Working on GUI (graphical user interface is still under development)

TESTING TECHONOLOGIES USED

- This app was test by SELENIUM
- and also, by installing in more than 10 users
- this app was tested in all operating system like windows, IOS and Android

SECURITY

Android has built-in security features that significantly reduce the frequency and impact of application security issues. The system is designed so that you can typically build your apps with the default system and file permissions and avoid difficult decisions about security.

The following core security features help you build secure apps:

- The Android Application Sandbox, which isolates your app data and code execution from other apps.
- An application framework with robust implementations of common security functionality such as cryptography, permissions, and secure IPC.
- An encrypted file system that can be enabled to protect data on lost or stolen devices.
- User-granted permissions to restrict access to system features and user data.
- Application-defined permissions to control application data on a per-app basis
- It is important that you be familiar with the Android security best practices in this document. Following these practices as general coding habits reduces the likelihood of inadvertently introducing security issues that adversely affect your users.

Bibliography

- MINOR PROJECT ALL DETAILS FIND HERE-: https://github.com/imbalram
- About testing tool: - www.selenium.dev