VEHICLE RESALE VALUE PREDICTION

Using Random Forest Regression

Developed by: Ashik Hatter, Anusha A R, Arpitha MC,

Navami B G, Rashika N

Smart Bridge-Remote Summer Internship Program

1. INTRODUCTION:

A **used vehicle**, a **pre-owned vehicle**, or a **secondhand vehicle**, is a vehicle that has previously had one or more retail owners. Used vehicles are sold through a variety of outlets, including franchise and independent vehicle dealers, rental vehicle companies, buy here pay here dealerships, leasing offices, auctions, and private party sales. Some vehicle retailers offer "no-haggle prices," "certified" used vehicles, and extended service plans or warranties.

History: Established in 1898, the Empire State Motor Wagon Company in Catskill, New York was one of the very first American used vehicle lots.

The used vehicle market is substantially larger than other large retail sectors, such as the school and office products market.

Used vehicle pricing:

Used vehicle pricing reports typically produce three forms of pricing information.

- Dealer or retail price is the price expected to pay if buying from a licensed new-vehicle or used-vehicle dealer.
- Dealer trade-in price or wholesale price is the price a shopper should expect to receive from a dealer if trading in a vehicle. This is also the price that a dealer will typically pay for a vehicle at a dealer wholesale auction.
- Private-party price is the price expected to pay if buying from an individual. A private-party seller is hoping to get more money than they would with a trade-in to a dealer. A private-party buyer is hoping to pay less than the dealer retail price.

Pricing of used vehicles can be affected by geography. For example, convertibles have a higher demand in warmer climates than in cooler areas. Similarly, pickup trucks may be more in demand in rural than urban settings. The overall condition of the vehicle has a major impact on pricing. Condition is based on appearances, vehicle history, mechanical condition, and mileage. There is much subjectivity in how the condition of a vehicle is evaluated.

There are various theories as to how the market determines the prices of used vehicles sold by private parties, especially relative to new vehicles. One theory suggests that new vehicle dealers are able to put more effort into selling a vehicle, and can therefore stimulate stronger demand. Another theory suggests that owners of problematic vehicles ("lemons") are more likely to want to sell their vehicles than owners of perfectly functioning vehicles. Therefore, someone buying a used vehicle bears a higher risk of buying a lemon, and the market price tends to adjust downwards to reflect that.

1.1 Overview:

Determining whether the listed price of a used vehicle 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 vehicle 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, but is compute heavy. Conventional linear regression also yielded satisfactory results, with the advantage of a significantly lower training time in comparison to the aforementioned methods.

1.2 Purpose:

Deciding whether a used vehicle is worth the posted price when you see listings online can be difficult. Several factors, including mileage, make, model, year, etc. can influence the actual worth of a vehicle. From the perspective of a seller, it is also a dilemma to price a used vehicle appropriately. Based on existing data, the aim is to use machine learning algorithms to develop models for predicting used vehicle prices.

1.3 Data Description

Dataset For this project, we are using the dataset on used vehicle sales from all over the United States, available on Kaggle. Dataset contains information about used vehicles name, brand, vehicle type, abtest, powerPS, model, gear type, kilometers driven, year of registration, month of registration,

2. Existing problem:

The prices of new vehicles 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 vehicle can be assured of the money they invest to be worthy. But due to the increased price of new vehicles and the incapability of customers to buy new vehicles due to the lack of funds, used vehicles sales are on a global increase (Pal, Arora and Palakurthy, 2018). There is a need for a used vehicle price prediction system to effectively determine the worthiness of the vehicle using a variety of features. Even though there are websites that offers this service, their prediction method may not be the best. Besides, different models and systems may contribute on predicting power for a used vehicle’s actual market value. It is important to know their actual market value while both buying and selling.

3. T**he Client:**

To be able to predict used vehicles market value can help both buyers and sellers**.**

Used vehicle sellers (dealers): They are one of the biggest target group that can be interested in results of this study. If used vehicle sellers better understand what makes a vehicle desirable, what the important features are for a used vehicle, then they may consider this knowledge and offer a better service.

Online pricing services: There are websites that offers an estimate value of a vehicle. They may have a good prediction model. However, having a second model may help them to give a better prediction to their users. Therefore, the model developed in this study may help online web services that tells a used vehicle’s market value.

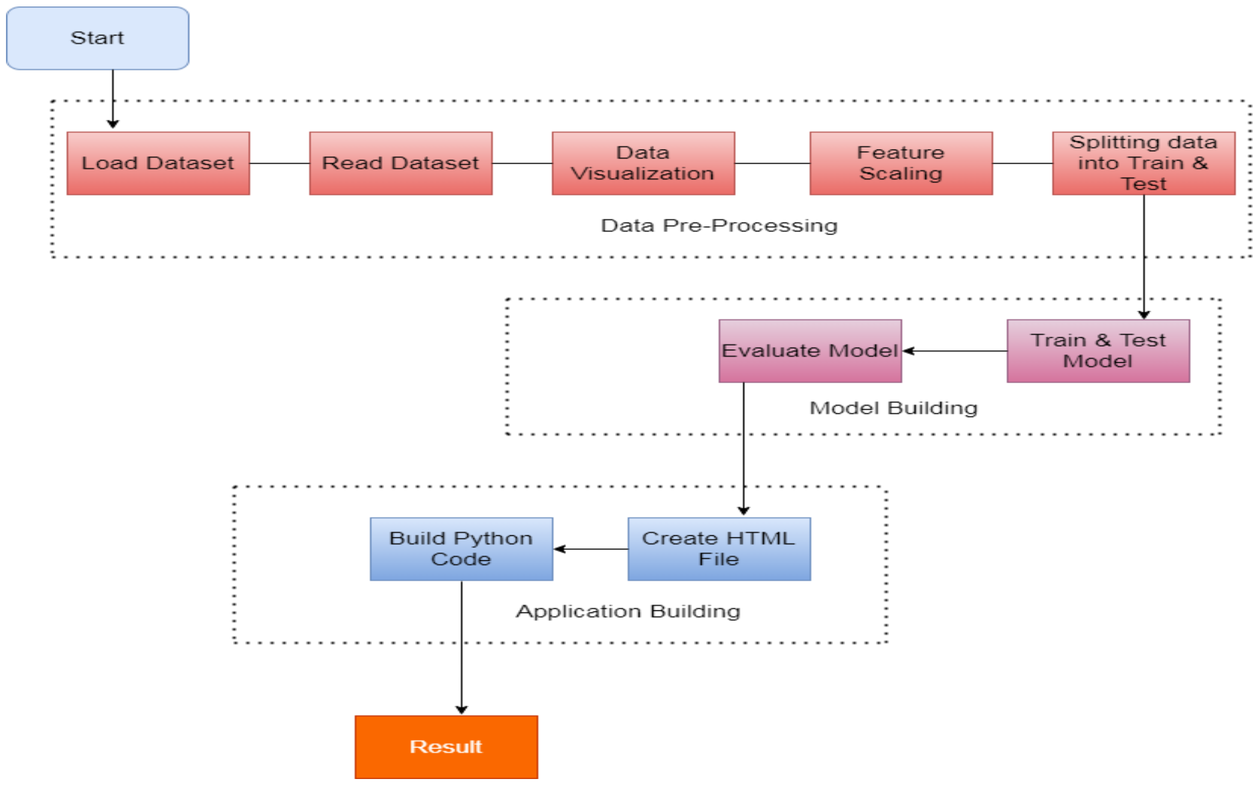
Individuals: There are lots of individuals who are interested in the used vehicle market at some points in their life because they wanted to sell their vehicle or buy a used vehicle. In this process, it’s a big corner to pay too much or sell less then it’s market value.

4. Machine-Learning model with high accuracy:

Random Forest is an ensemble learning based regression model. It uses a model called decision tree, specifically as the name suggests, multiple decision trees to generate the ensemble model which collectively produces a prediction. The benefit of this model is that the trees are produced in parallel and are relatively uncorrelated, thus producing good results as each tree is not prone to individual errors of other trees. This uncorrelated behavior is partly ensured by the use of Bootstrap Aggregation or bagging providing the randomness required to produce robust and uncorrelated trees. This model was hence chosen to account for the large number of features in the dataset and compare a bagging technique with the following gradient boosting methods. This model gave an accuracy of **81%.**

Random forest is a set of multiple decision trees. Deep decision trees may suffer from overfitting, but random forest prevents overfitting by creating trees on random subsets. That's why, it's a good model to in the analysis**.**

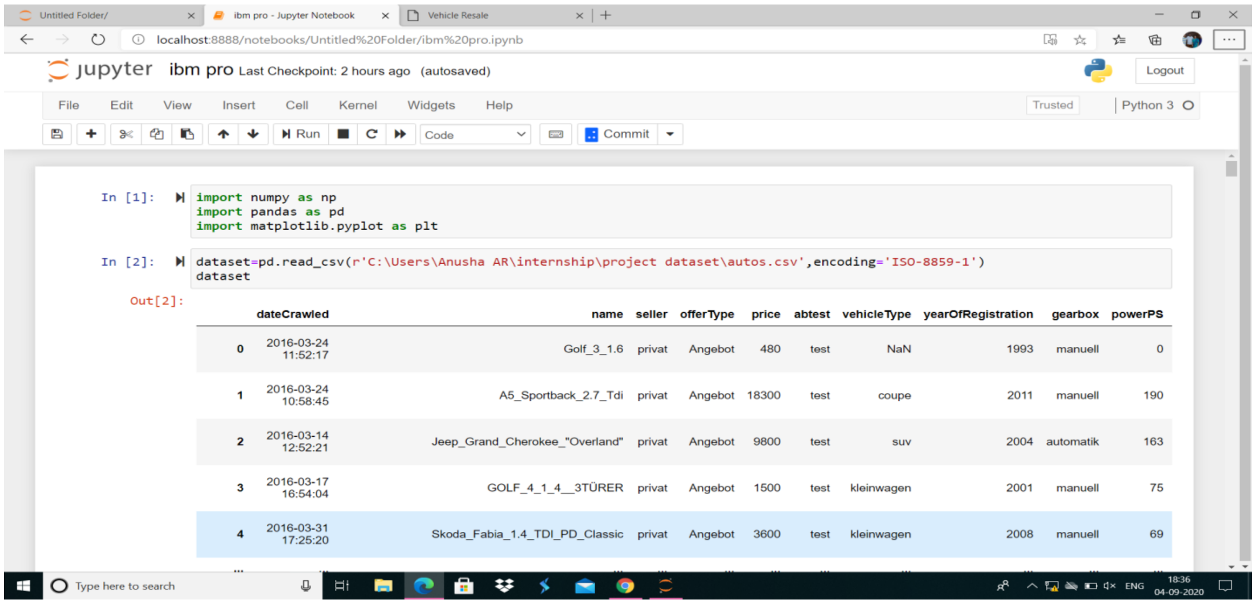
5. Flow Chart:



. Fig 1.1 :Flow Chart

6. Screenshot:

6.1 Importing Libraries and Dataset:

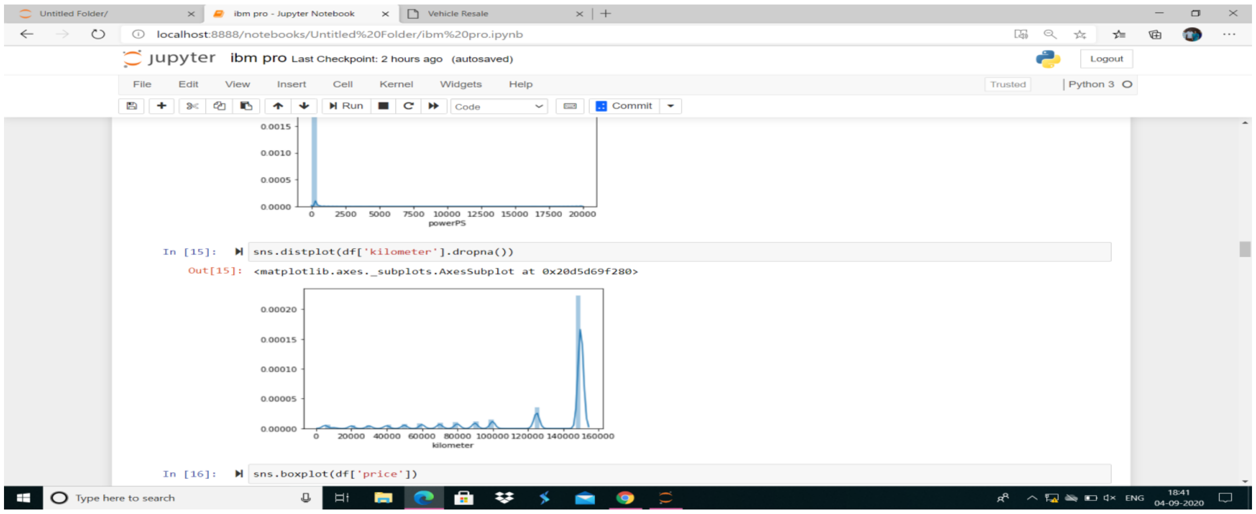


```
In [1]:  import numpy as np
         import pandas as pd
         import matplotlib.pyplot as plt

In [2]:  dataset=pd.read_csv(r'C:\Users\Anusha AR\internship\project dataset\autos.csv',encoding='ISO-8859-1')
         dataset
Out[2]:
```

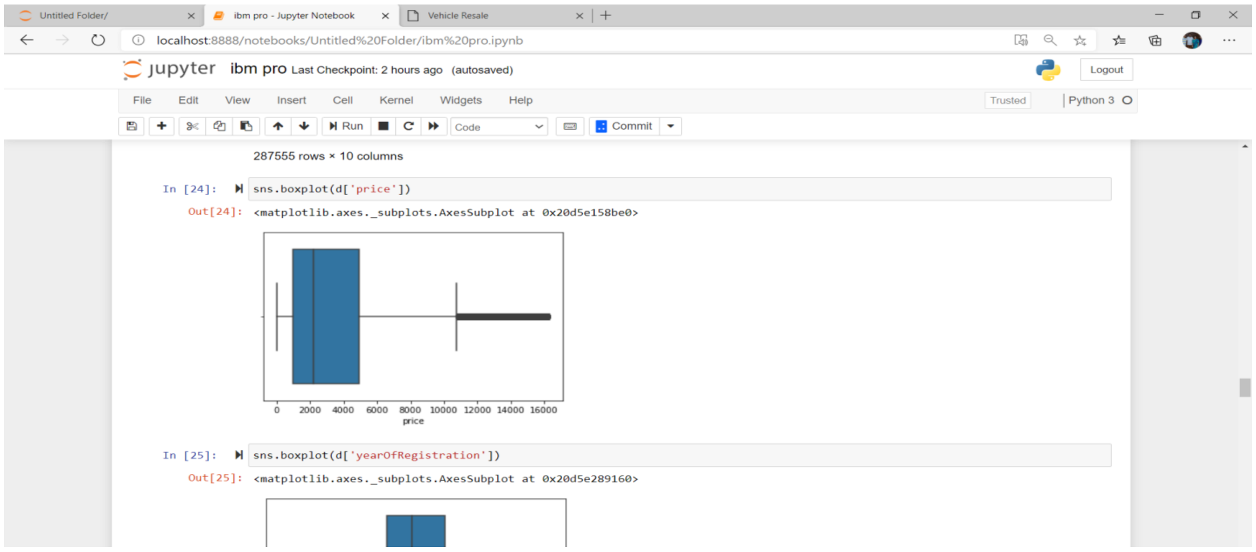
	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800	test	suv	2004	automatik	163
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500	test	kleinwagen	2001	manuell	75
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600	test	kleinwagen	2008	manuell	69

Fig 1.2 : Screenshot of Libraries and Dataset

6.2 Data Visualisation by Seaborn:



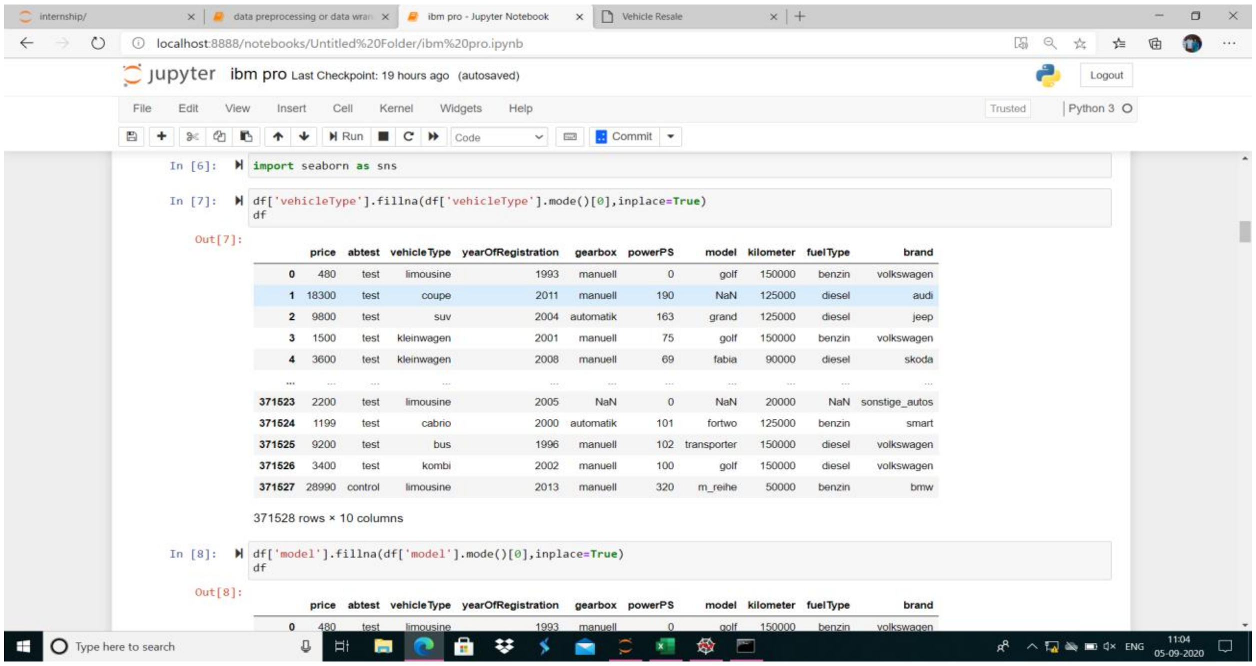
```
In [15]:  sns.distplot(df['kilometer'].dropna())
Out[15]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d5d69f280>

In [16]:  sns.boxplot(df['price'])
```


Untitled Folder/
ibm pro - Jupyter Notebook
Vehicle Resale
localhost:8888/notebooks/Untitled%20Folder/ibm%20pro.ipynb
jupyter ibm pro Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Commit
0 2000 4000 6000 8000 10000 12000 14000 16000
price

```
287555 rows × 10 columns

In [24]:  sns.boxplot(d['price'])
Out[24]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d5e158be0>

In [25]:  sns.boxplot(d['yearOfRegistration'])
Out[25]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d5e289160>
```

Fig 1.3 : Screenshot of Data Visualisation

6.3 Taking Care Of Missing Data:


internship/
data preprocessing or data wran
ibm pro - Jupyter Notebook
Vehicle Resale
localhost:8888/notebooks/Untitled%20Folder/ibm%20pro.ipynb
jupyter ibm pro Last Checkpoint: 19 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Commit
Type here to search
ENG
11:04
05-09-2020

```
In [6]:  import seaborn as sns

In [7]:  df['vehicleType'].fillna(df['vehicleType'].mode()[0],inplace=True)
         df
Out[7]:
```

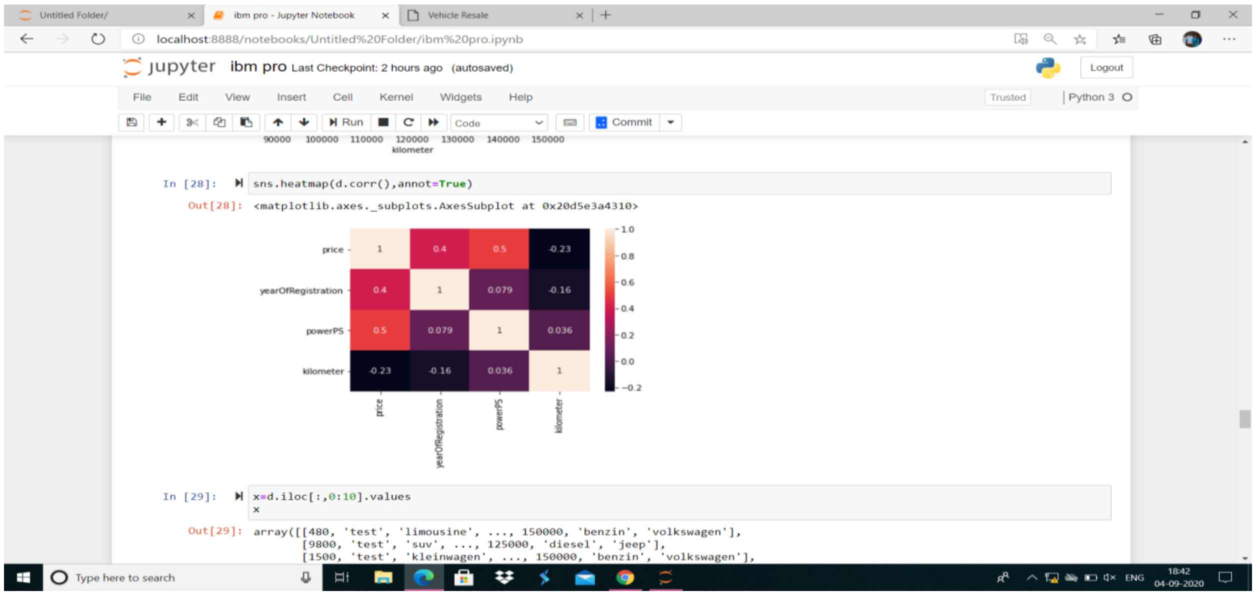
	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	fuelType	brand
0	480	test	limousine	1993	manuell	0	golf	150000	benzin	volkswagen
1	18300	test	coupe	2011	manuell	190	NaN	125000	diesel	audi
2	9800	test	suv	2004	automatik	163	grand	125000	diesel	jeep
3	1500	test	kleinwagen	2001	manuell	75	golf	150000	benzin	volkswagen
4	3600	test	kleinwagen	2008	manuell	69	fabia	90000	diesel	skoda
...	...	...	...	...	...	...	...	...	...	...
371523	2200	test	limousine	2005	NaN	0	NaN	20000	NaN	sonstige_autos
371524	1199	test	cabrio	2000	automatik	101	fortwo	125000	benzin	smart
371525	9200	test	bus	1996	manuell	102	transporter	150000	diesel	volkswagen
371526	3400	test	kombi	2002	manuell	100	golf	150000	diesel	volkswagen
371527	28990	control	limousine	2013	manuell	320	m_reihe	50000	benzin	bmw

```
371528 rows × 10 columns

In [8]:  df['model'].fillna(df['model'].mode()[0],inplace=True)
         df
Out[8]:
```

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	fuelType	brand
0	480	test	limousine	1993	manuell	0	golf	150000	benzin	volkswagen

Fig 1.4 : Screenshot of handling of missing values

6.4 Heatmap:



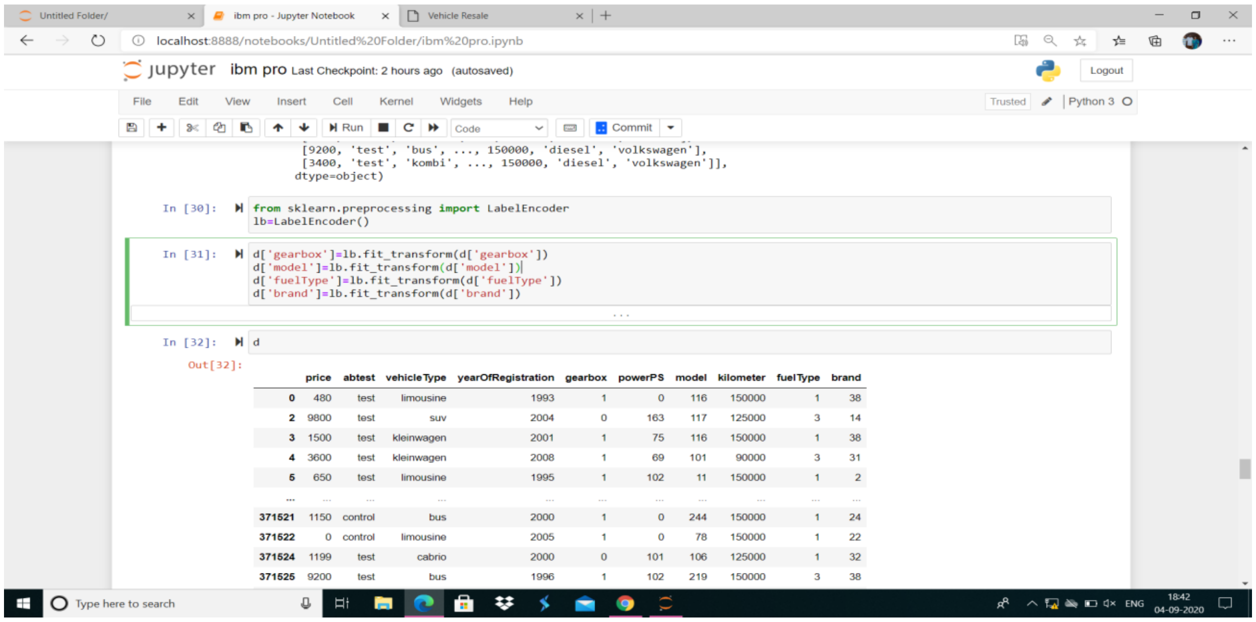
```
In [28]:  sns.heatmap(d.corr(),annot=True)
Out[28]:  <matplotlib.axes._subplots.AxesSubplot at 0x20d5e3a4310>
```

	price	yearOfRegistration	powerPS	kilometer
price	1	0.4	0.5	-0.23
yearOfRegistration	0.4	1	0.079	-0.16
powerPS	0.5	0.079	1	0.036
kilometer	-0.23	-0.16	0.036	1

```
In [29]:  x=d.iloc[:,0:10].values
          x
Out[29]:  array([[480, 'test', 'limousine', ..., 150000, 'benzin', 'volkswagen'],
                 [9800, 'test', 'suv', ..., 125000, 'diesel', 'jeep'],
                 [1500, 'test', 'kleinwagen', ..., 150000, 'benzin', 'volkswagen'],
```

Fig 1.5 : Screenshot of heatmap

6.5 Label Encoding:

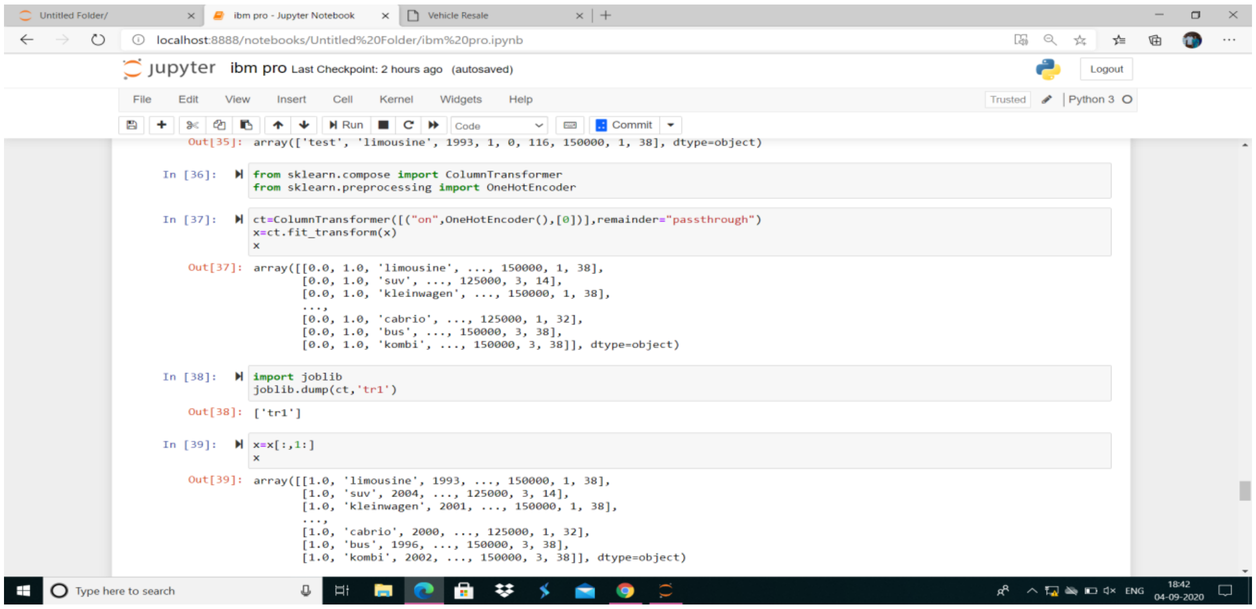

Untitled Folder/
ibm pro - Jupyter Notebook
Vehicle Resale
localhost:8888/notebooks/Untitled%20Folder/ibm%20pro.ipynb
jupyter ibm pro Last Checkpoint: 2 hours ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code
Commit
Type here to search
ENG
18:42
04-09-2020

```
                 [9200, 'test', 'bus', ..., 150000, 'diesel', 'volkswagen'],
                 [3400, 'test', 'kombi', ..., 150000, 'diesel', 'volkswagen']],
                dtype=object)

In [30]:  from sklearn.preprocessing import LabelEncoder
          lb=LabelEncoder()

In [31]:  d['gearbox']=lb.fit_transform(d['gearbox'])
          d['model']=lb.fit_transform(d['model'])
          d['fuelType']=lb.fit_transform(d['fuelType'])
          d['brand']=lb.fit_transform(d['brand'])

In [32]:  d
Out[32]:
```

	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	fuelType	brand
0	480	test	limousine	1993	1	0	116	150000	1	38
2	9800	test	suv	2004	0	163	117	125000	3	14
3	1500	test	kleinwagen	2001	1	75	116	150000	1	38
4	3600	test	kleinwagen	2008	1	69	101	90000	3	31
5	650	test	limousine	1995	1	102	11	150000	1	2
...	...	...	...	...	...	...	...	...	...	...
371521	1150	control	bus	2000	1	0	244	150000	1	24
371522	0	control	limousine	2005	1	0	78	150000	1	22
371524	1199	test	cabrio	2000	0	101	106	125000	1	32
371525	9200	test	bus	1996	1	102	219	150000	3	38

Fig 1.6 : Screenshot of label encoding

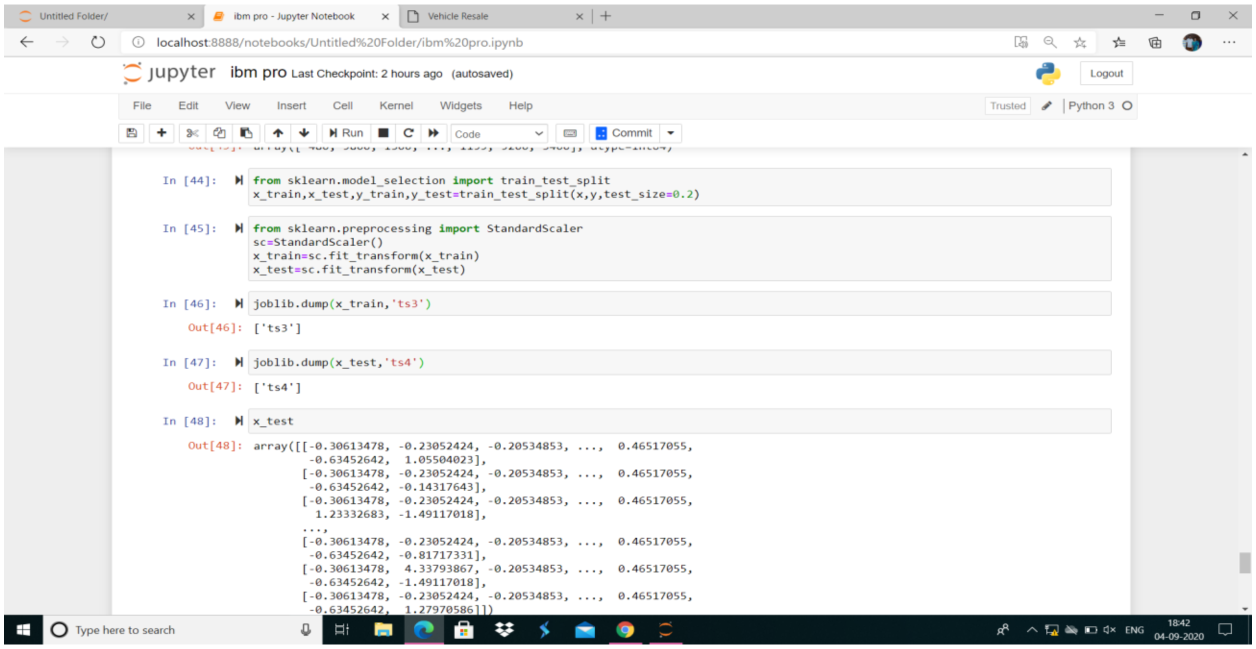
6.6 One Hot Encoding:



```
Out[35]: array(['test', 'limousine', 1993, 1, 0, 116, 150000, 1, 38], dtype=object)

In [36]:  from sklearn.compose import ColumnTransformer
          from sklearn.preprocessing import OneHotEncoder

In [37]:  ct=ColumnTransformer([("on",OneHotEncoder(),[0])],remainder="passthrough")
          x=ct.fit_transform(x)
          x

Out[37]:  array([[0.0, 1.0, 'limousine', ..., 150000, 1, 38],
                 [0.0, 1.0, 'suv', ..., 125000, 3, 14],
                 [0.0, 1.0, 'kleinwagen', ..., 150000, 1, 38],
                 ...,
                 [0.0, 1.0, 'cabrio', ..., 125000, 1, 32],
                 [0.0, 1.0, 'bus', ..., 150000, 3, 38],
                 [0.0, 1.0, 'kombi', ..., 150000, 3, 38]], dtype=object)

In [38]:  import joblib
          joblib.dump(ct,'tr1')

Out[38]:  ['tr1']

In [39]:  x=x[:,1:]
          x

Out[39]:  array([[1.0, 'limousine', 1993, ..., 150000, 1, 38],
                 [1.0, 'suv', 2004, ..., 125000, 3, 14],
                 [1.0, 'kleinwagen', 2001, ..., 150000, 1, 38],
                 ...,
                 [1.0, 'cabrio', 2000, ..., 125000, 1, 32],
                 [1.0, 'bus', 1996, ..., 150000, 3, 38],
                 [1.0, 'kombi', 2002, ..., 150000, 3, 38]], dtype=object)
```

Fig 1.7 : Screenshot of one hot encoding

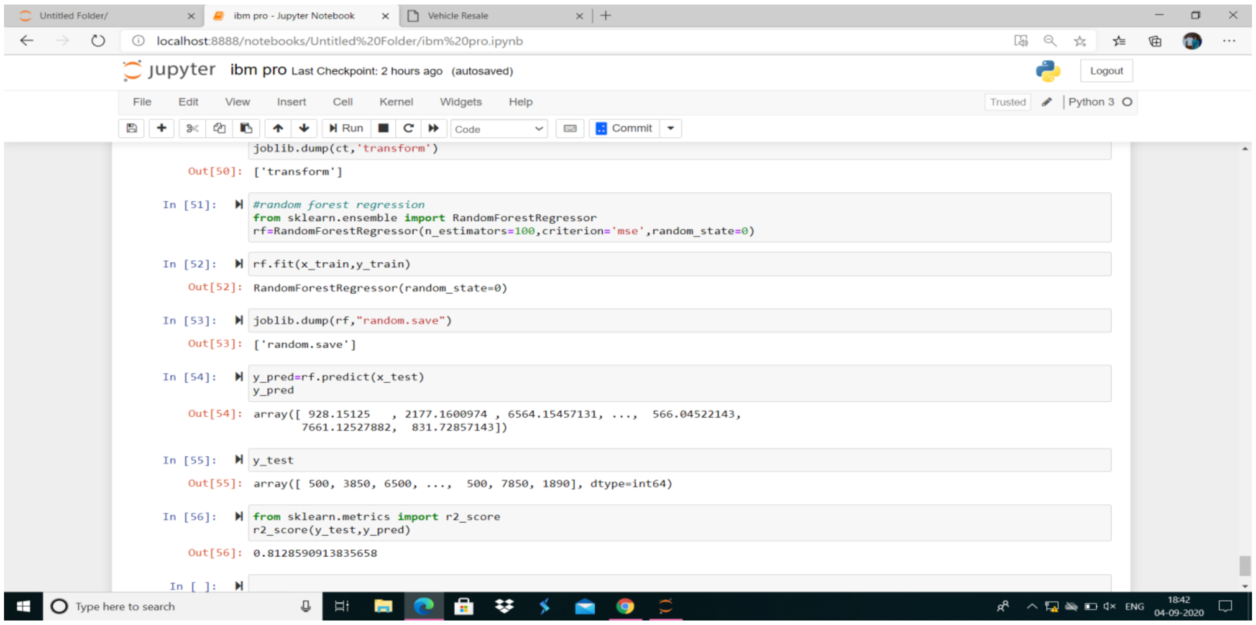
6.7 Feature Scaling and Splitting Data Into Test and Train:



```
In [44]:  from sklearn.model_selection import train_test_split
          x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2)

In [45]:  from sklearn.preprocessing import StandardScaler
          sc=StandardScaler()
          x_train=sc.fit_transform(x_train)
          x_test=sc.fit_transform(x_test)

In [46]:  joblib.dump(x_train,'ts3')

Out[46]:  ['ts3']

In [47]:  joblib.dump(x_test,'ts4')

Out[47]:  ['ts4']

In [48]:  x_test

Out[48]:  array([[-0.30613478, -0.23052424, -0.20534853, ...,  0.46517055,
                  -0.63452642,  1.05504023],
                 [-0.30613478, -0.23052424, -0.20534853, ...,  0.46517055,
                  -0.63452642, -0.14317643],
                 [-0.30613478, -0.23052424, -0.20534853, ...,  0.46517055,
                   1.23332683, -1.49117018],
                 ...,
                 [-0.30613478, -0.23052424, -0.20534853, ...,  0.46517055,
                  -0.63452642, -0.81717331],
                 [-0.30613478,  4.33793867, -0.20534853, ...,  0.46517055,
                  -0.63452642, -1.49117018],
                 [-0.30613478, -0.23052424, -0.20534853, ...,  0.46517055,
                  -0.63452642,  1.27970586]])
```

Fig 1.8 : Screenshot of splitting data

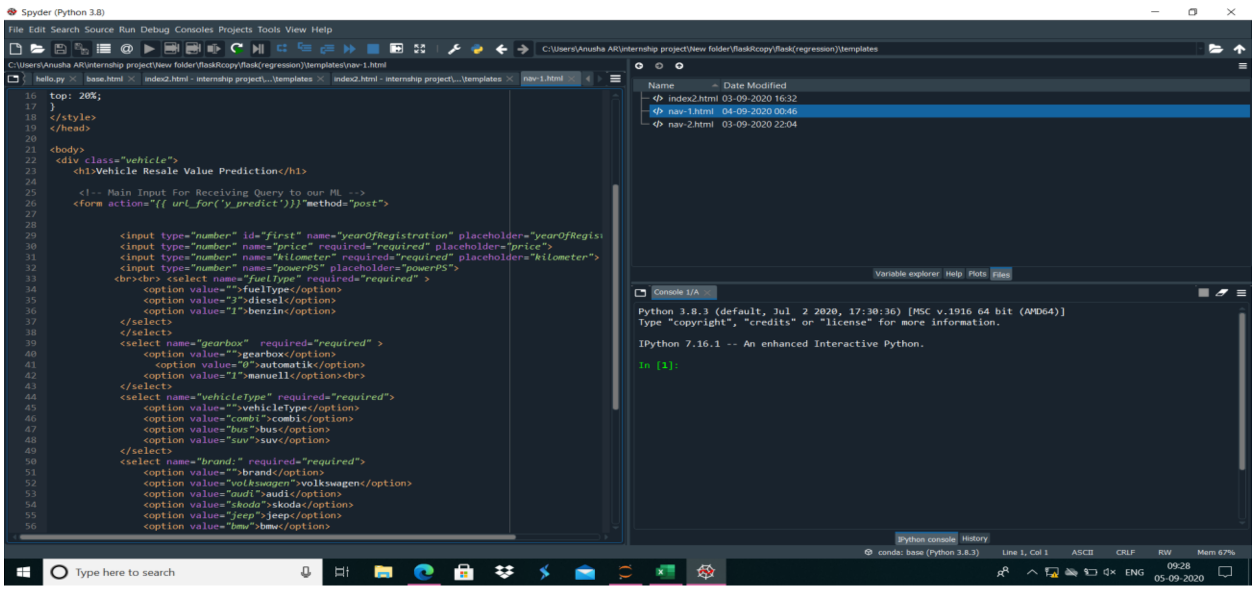
6.8 Model Building:



```
joblib.dump(ct,'transform')

Out[50]: ['transform']

In [51]: #random forest regression
         from sklearn.ensemble import RandomForestRegressor
         rf=RandomForestRegressor(n_estimators=100,criterion='mse',random_state=0)

In [52]: rf.fit(x_train,y_train)

Out[52]: RandomForestRegressor(random_state=0)

In [53]: joblib.dump(rf,"random.save")

Out[53]: ['random.save']

In [54]: y_pred=rf.predict(x_test)
         y_pred

Out[54]: array([ 928.15125   , 2177.1600974 , 6564.15457131, ...,  566.04522143,
                7661.12527882,  831.72857143])

In [55]: y_test

Out[55]: array([ 500, 3850, 6500, ...,  500, 7850, 1890], dtype=int64)

In [56]: from sklearn.metrics import r2_score
         r2_score(y_test,y_pred)

Out[56]: 0.8128590913835658

In [ ]:
```

Fig 1.9 : Screenshot of model building

6.9 HTML File in Spyder:

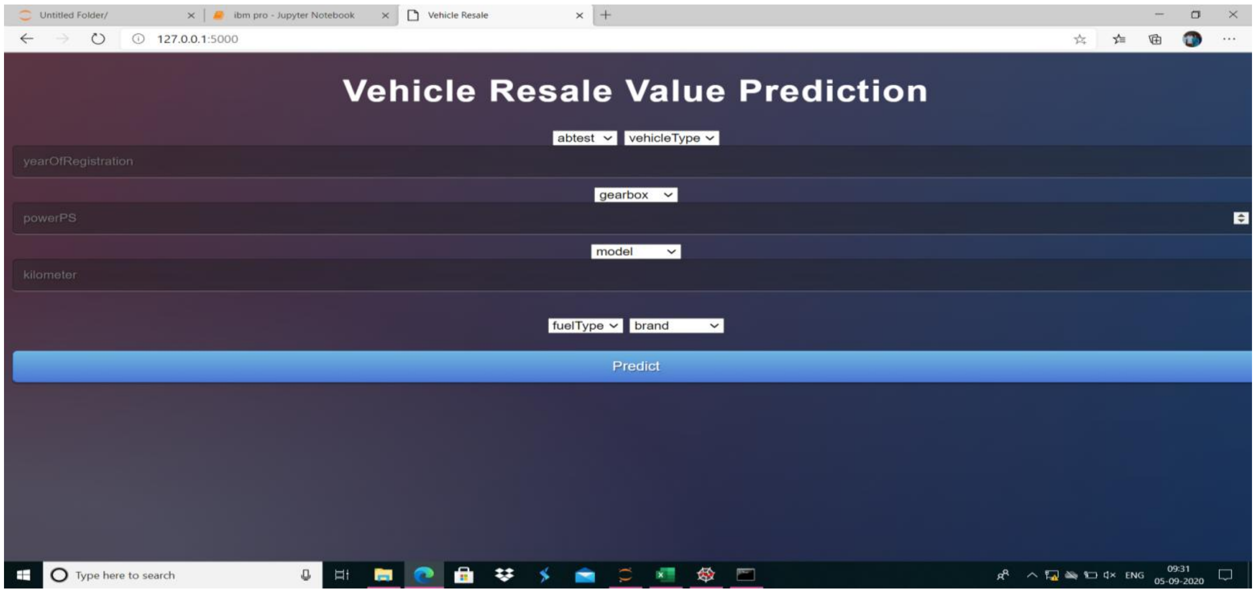


```
top: 20%;
}
</style>
</head>

<body>
 <div class="vehicle">
    <h1>Vehicle Resale Value Prediction</h1>

     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('y_predict')}}"method="post">


            <input type="number" id="first" name="yearOfRegistration" placeholder="yearOfRegis
            <input type="number" name="price" required="required" placeholder="price">
            <input type="number" name="kilometer" required="required" placeholder="kilometer">
            <input type="number" name="powerPS" placeholder="powerPS">
           <br><br> <select name="fuelType" required="required" >
                <option value="">fuelType</option>
                <option value="3">diesel</option>
                <option value="1">benzin</option>
            </select>
            </select>
            <select name="gearbox"  required="required" >
                <option value="">gearbox</option>
                  <option value="0">automatik</option>
                <option value="1">manuell</option><br>
            </select>
            <select name="vehicleType" required="required">
                <option value="">vehicleType</option>
                <option value="combi">combi</option>
                <option value="bus">bus</option>
                <option value="suv">suv</option>
            </select>
            <select name="brand:" required="required">
                <option value="">brand</option>
                <option value="volkswagen">volkswagen</option>
                <option value="audi">audi</option>
                <option value="skoda">skoda</option>
                <option value="jeep">jeep</option>
                <option value="bmw">bmw</option>
```

```
Python 3.8.3 (default, Jul  2 2020, 17:30:36) [MSC v.1916 64 bit (AMD64)]
Type "copyright", "credits" or "license" for more information.

IPython 7.16.1 -- An enhanced Interactive Python.

In [1]:
```

Fig 1.10 : Screenshot of html file

6.10 Web Page Before Entering The Inputs:



Vehicle Resale Value Prediction

abtest | vehicleType

yearOfRegistration

gearbox

powerPS

model

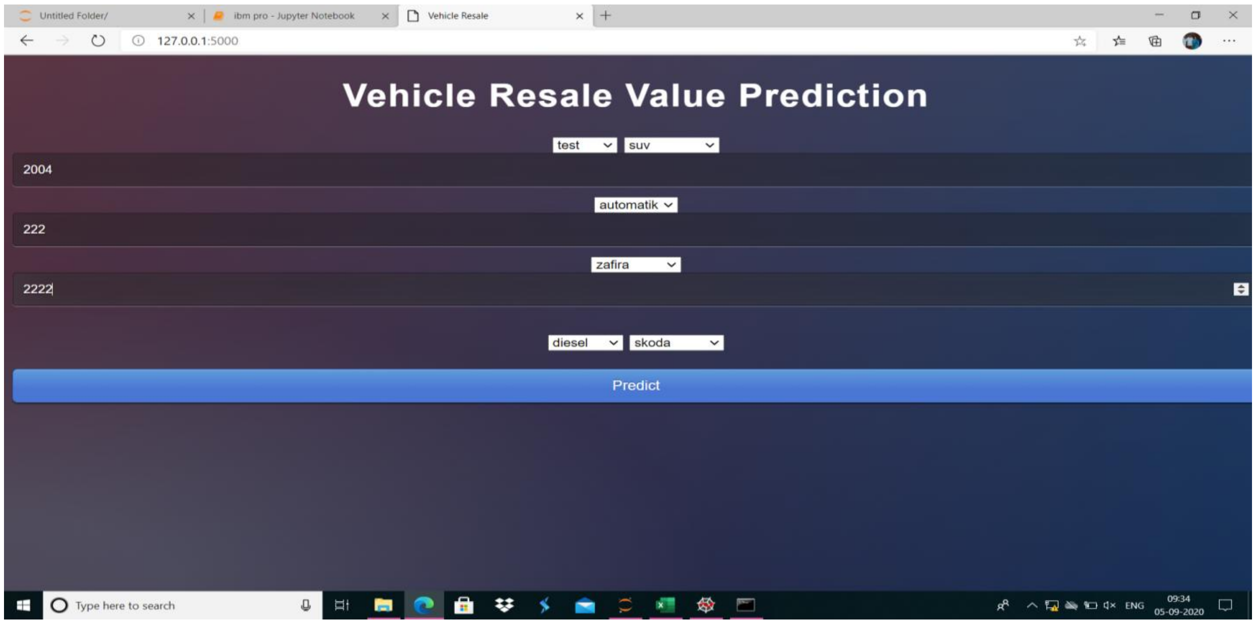
kilometer

fuelType | brand

Predict

Fig 1.11 : Screenshot of output

6.11 Web Page After Entering The Input:



Vehicle Resale Value Prediction

test | suv

2004

automatik

222

zafira

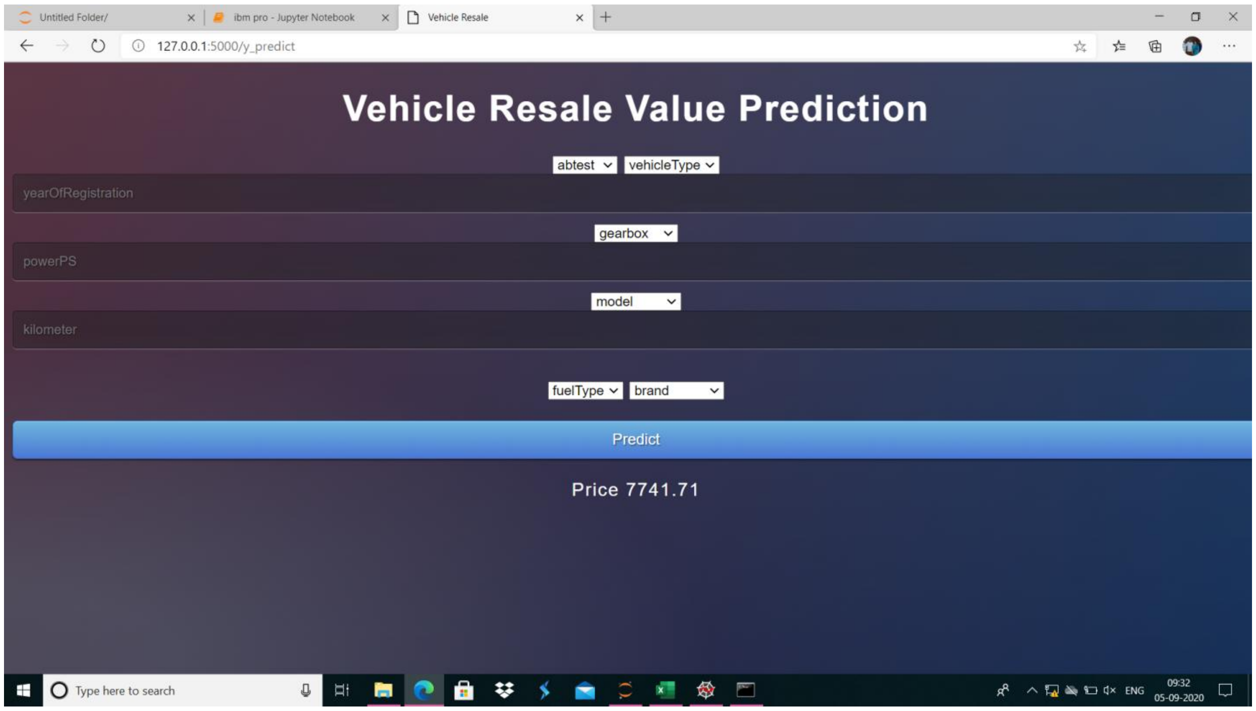
2222

diesel | skoda

Predict

Fig 1.12 : Screenshot of output

6.12 Output:



Price 7741.71

Fig 1.13 : Screenshot of output

7. Software Designing:

● Jupyter Notebook Environment

● Spyder Ide

● Machine Learning Algorithms

● Python (pandas, numpy, matplotlib, seaborn, sklearn)

● HTML

● Flask

We developed this resale value prediction by using the Python language which is a interpreted and high level programming language and usng the Machine Learning algorithms. for coding we used the Jupyter Notebook environment of the Anaconda distributions and the Spyder, it is an integrated scientific programming in the python language. For creating an user interface for the prediction we used the Flask. It is a micro web framework written in Python. It is classified as a microframework because it does not require particular tools or libraries. It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions, and a scripting language to create a webpage is HTML by creating the templates to use in the functions of the Flask and HTML.

8. Results and conclusion:

The results of our tests were quantified in terms of the R score of our predictions. score is a statistical 2 R2 measure of how close the data are to the fitted regression line. Random Forest method which marginally outperforms Linear Regression. However Random Forests tend to overfit the dataset due to the tendency of growing longer trees. This was worked upon by restricting the depth of trees to different values.

9. Limitations:

This study used different models in order to predict used vehicle prices. However, there was a relatively dataset for making a strong inference because number of observations was only 6 lakh. Gathering more data can yield more robust predictions. Secondly, there could be more features that can be good predictors. For example, here are some variables that might improve the model: number of doors, number of seats, gas/mile (per gallon), color, mechanical and cosmetic reconditioning time, used-to-new ratio, appraisal-to-trade ratio.

11. Bibliography:

1. https://www.kaggle.com/orgesleka/used-cars-dataset

2. N. Monburinon, P. Chertchom, T. Kaewkiriya, S. Rungpheung, S. Buya and P. Boonpou, "Prediction of prices for used vehicle by using regression models," 2018 5th International Conference on Business and Industrial Research (ICBIR), Bangkok, 2018, pp. 115-119.

3. Listiani M. 2009. Support Vector Regression Analysis for Price Prediction in a Vehicle Leasing Application. Master Thesis. Hamburg University of Technology

4. Chen, Tianqi, and Vehiclelos Guestrin. "Xgboost: A scalable tree boosting system." Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining. ACM, 2016.

5. Ke, Guolin, et al. "Lightgbm: A highly efficient gradient boosting decision tree." Advances in Neural Information Processing Systems. 2017.

6. Fisher, Walter D. "On grouping for maximum homogeneity." Journal of the American statistical Association 53.284 (1958): 789-798. 7. https://scikit-learn.org/stable/modules/classes.html: Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.

Appendix:

HTML:

```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Vehicle Resale</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet' type='text/css'>
<link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">

<style>
.login{
top: 20%;
}
</style>
</head>
<body>
```

```
 <div class="vehicle">
          <h1>Vehicle Resale Value Prediction</h1>
     <!-- Main Input For Receiving Query to our ML -->
    <form action="{{ url_for('y_predict')}}"method="post">

           <select name="abtest:" required="required">
               <option value="">abtest</option>
                                          <option value="1">test</option>
               <option value="0">control</option><br>
           </select>
           <select name="vehicleType" required="required">
                                   <option value="">vehicleType</option>
                                    <option value="limousine">limousine</option>
               <option value="kombi">kombi</option>
                                          <option value="bus">bus</option>
                                          <option value="suv">suv</option>
           </select>
           <input type="number" id="first" name="yearOfRegistration"
placeholder="yearOfRegistration" >
           <select name="gearbox"  required="required" >
               <option value="">gearbox</option>
                                           <option value="0">automatik</option>
               <option value="1">manuell</option><br>
           </select>
           <input type="number" name="powerPS" placeholder="powerPS">
            <select name="model:" required="required">
                                   <option value="">model</option>
               <option value="116">golf</option>
               <option value="117">grand</option>
                                          <option value="101">fabia</option>
                                    <option value="11">3er</option>

                                    <option value="244">zafira</option>

                                    <option value="78">colt</option>

                                    <option value="106">fortow</option>

                                    <option value="219">transporter</option><br>

          </select>

          <input type="number" name="kilometer" required="required" placeholder="kilometer">


         <br><br> <select name="fuelType" required="required" >

             <option value="">fuelType</option>

                                    <option value="3">diesel</option>

                                    <option value="1">benzin</option>

          </select>


          <select name="brand:" required="required">

                                <option value="">brand</option>

             <option value="38">volkswagen</option>

             <option value="32">smart</option>

                                    <option value="31">skoda</option>

                                    <option value="14">jeep</option>

                                    <option value="2">bmw</option>

                                    <option value="22">mitsubishi</option><br>

          </select>

        <br><br><button type="submit" class="btn btn-primary btn-block btn-
large">Predict</button>

        </form><br>

  {{ prediction_text }}

 </div>

</body>

</html>
```

App.py

```python
import numpy as np
from flask import Flask, request, jsonify, render_template
from joblib import load
app = Flask(__name__)
model= load('random1.save')
trans1=load('ts1')
trans2=load('ts2')


@app.route('/')
def home():
    return render_template('index.html')

@app.route('/y_predict',methods=['POST'])
def y_predict():
    '''
    For rendering results on HTML GUI
    '''
    x_test = [[x for x in request.form.values()]]
    print(x_test)

    test=trans1.transform(x_test)
    test=test[:,1:]

    test=trans2.transform(x_test)
    test=test[:,1:]



    print(test)
    prediction = model.predict(test)
    print(prediction)
    output=prediction[0]

    return render_template('index.html', prediction_text='Price {}'.format(output))

'''@app.route('/predict_api',methods=['POST'])
def predict_api():

    #For direct API calls trought request

    data = request.get_json(force=True)
    prediction = model.y_predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)'''

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