1. **Used Car Valuation Using IBM Watson Machine Learning Service**

2. INTRODUCTION

a. OVERVIEW

With difficult economic conditions, it is likely that sales of second-hand imported (reconditioned) cars and used cars will increase. In many developed countries, it is common to lease a car rather than buying it outright. 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. Thus, it is of commercial interest to sellers/financers to be able to predict the salvage value (residual value) of cars with accuracy.

b. PURPOSE

In order to predict the resale value of the car, we proposed an intelligent, flexible, and effective system that is based on using regression algorithms. Considering the main factors which would affect the resale value of a vehicle a regression model is to be built that would give the nearest resale value of the vehicle. We will be using various regression algorithms and algorithm with the best accuracy will be taken as a solution, then it will be integrated to the web-based application where the user is notified with the status of his product.

3. LITREATURE SURVEY

a. EXISTING SYSTEM

There are various sources to help you find out the value of your used car; three popular sources are cardekho(cardekho.com), cars24(cars24.com) and orangebookvalue.

Different factors will affect the value of your vehicle such as the mileage, the condition, your location, and the colour of the car. Most of the existing system mentioned above -to get the estimated value of your car, what you have to do is add basic details of your car like, the manufacture year, model, kilometres driven, your city, your contact number and add few photographs of your car and as a result price range of our car is displayed.

Even though there is not much problem that we can see in any of the existing system, sometimes price range displayed for the used car may not be satisfiable to the seller, this usually happens when we enter fewer details like registration year, car model, kilometres driven, mileage etc and system displays only average price which may be far less than expected price by the seller.

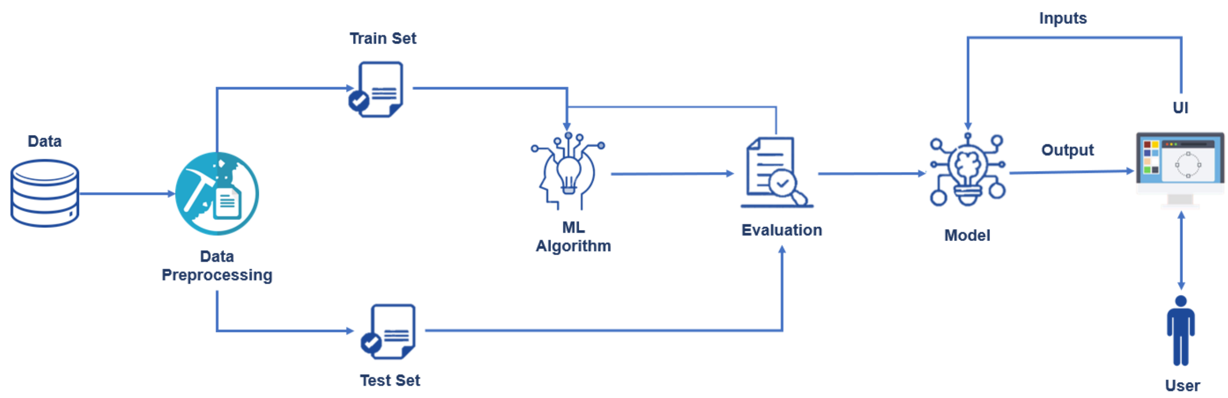
b. PROPOSED SYSTEM

Proposed system considers all the drawbacks of the existing system and implements an intelligent, flexible, and effective system that is based on using regression algorithms. The main factors which would affect the resale value of a vehicle such as registration year, month, kilometres driven, power, brand, fuel, vehicle type, damaged or not, gearbox type are considered.

Using those factors a regression model is to be built that would give the nearest resale value of the vehicle. We will be using various regression algorithms and algorithm with the best accuracy will be taken as a solution, then it will be integrated to the web-based application where the user is notified with the status of his product.

4. THEORITICAL ANALYSIS

a. BLOCK DIAGRAM



b. HARDWARE / SOFTWARE DESIGNING

IBM Watson Studio - IBM Watson Studio helps data scientists and analysts prepare data and build models at scale across any cloud.

IBM Watson Machine Learning - IBM Watson Machine Learning helps data scientists and developers accelerate AI and machine-learning deployment.

IBM Cloud Object Storage - IBM Cloud Object Storage makes it possible to store practically limitless amounts of data, simply and cost effectively.

Machine Learning Services - Machine learning as service is an umbrella term for collection of various cloud-based platforms that use machine learning tools to provide solutions that can help ML teams with: out-of-the box predictive analysis for various use cases, data pre-processing, model training and tuning.

5. EXPERIMENTAL INVESTIGATION

We are going to implement a machine learning model that predicts the resale value of the vehicle using main factors which would affect the resale value of a vehicle they are:

1. Registration year, month
2. Power of car
3. Kilometres driven
4. Gearbox type
5. Damaged or not
6. Fuel type
7. Brand, Model name
8. Vehicle type

Using the above factors our model can predict the resale value of the car in much accurate way.

b. EXPERIMENTAL ANALYSIS

Dataset is downloaded from the Kaggle which has 10 features

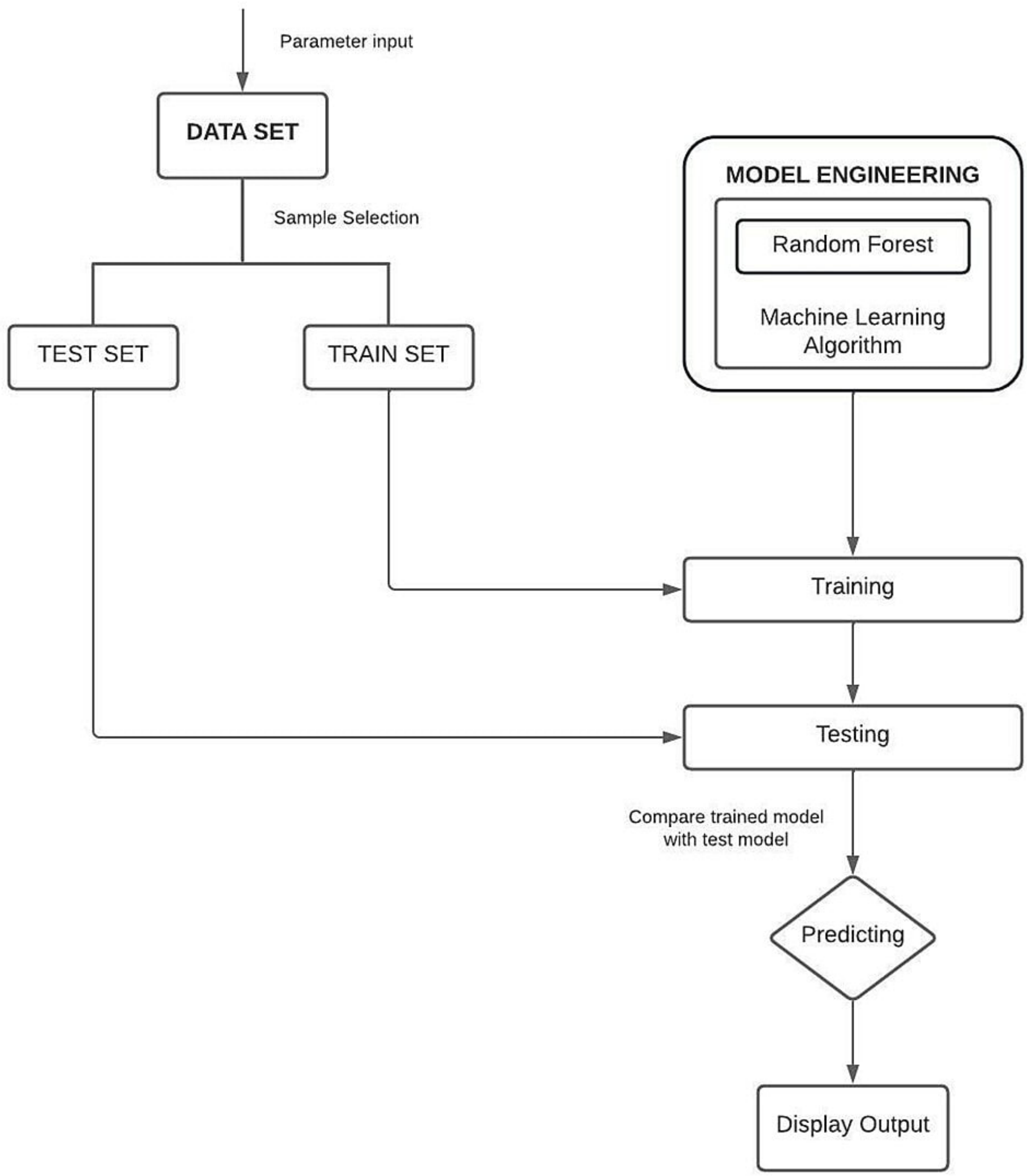
- Registration year
- Registration month of the car
- Power of car

- Kilometres driven by the car
- Gearbox type of the car
- Whether car is damaged or not
- Fuel type of the car
- Brand name of car
- Model name of car
- Vehicle type of car

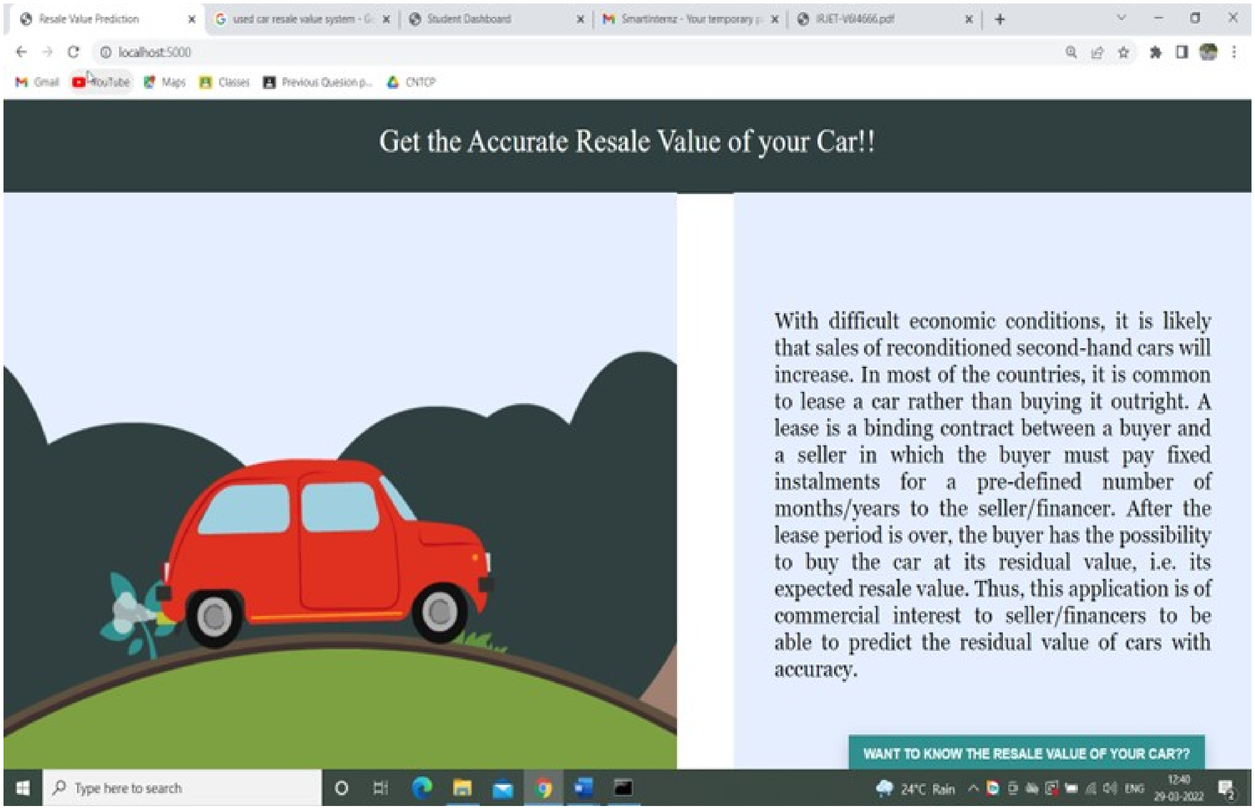
After all the pre-processing steps we performed on our dataset to get the final transformed dataset we split the data into train and test. For the prediction task we used Random-forest classifier.

Random-Forest: It is an ensemble technique which uses bagging technique. It uses number of meta-classifiers on various sub samples of the dataset and then averages the prediction to improve the final predictive outcome. This classifier can also control over fitting by proper parameter tuning.

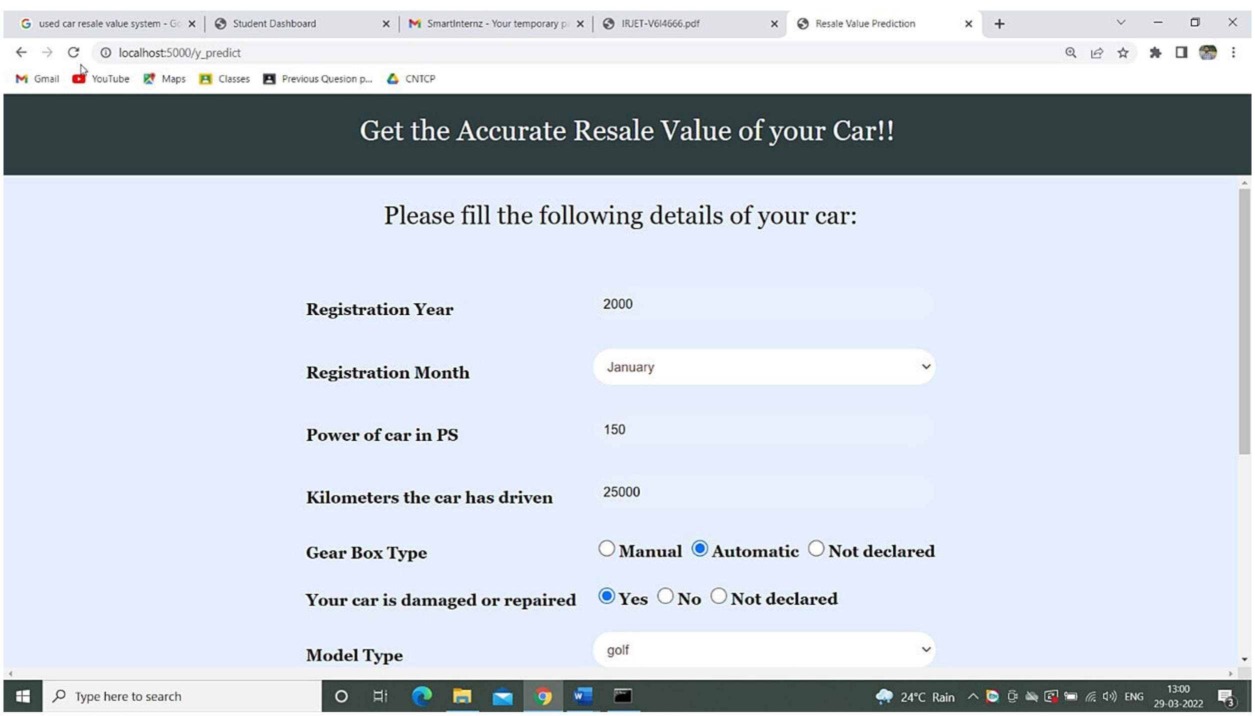
6. FLOWCHART

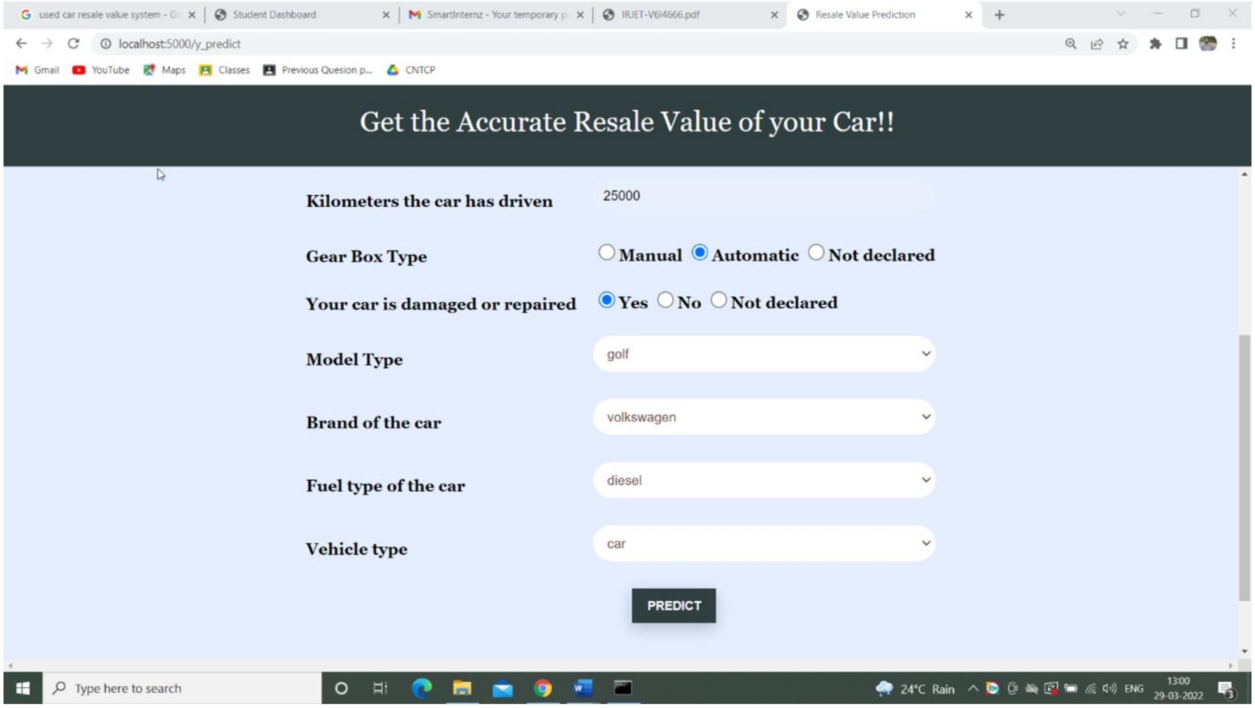

Parameter input
DATA SET
Sample Selection
TEST SET
TRAIN SET
MODEL ENGINEERING
Random Forest
Machine Learning Algorithm
Training
Testing
Compare trained model with test model
Predicting
Display Output

7. RESULT



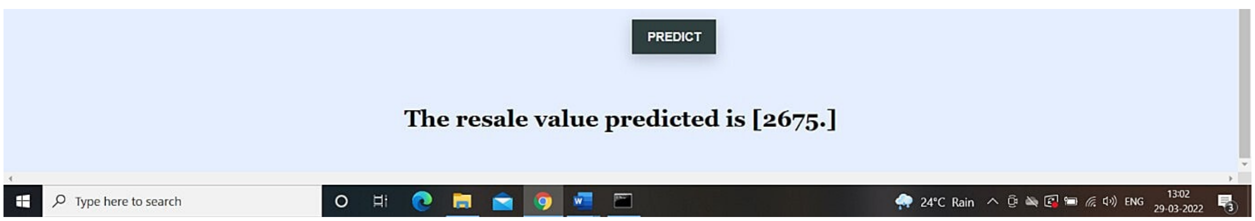
Above image shows the opening webpage, to know the resale value of your car click the button '**WANT TO KNOW THE RESALE VALUE OF YOUR CAR??**' shown below.





Now we fill each field with values ‘**2000**’, ‘**January**’, ‘**150**’,’**25000**’, ‘**Automatic**’, ‘**Yes**’, ‘**golf**’, ‘**Volkswagen**’, ‘**diesel**’, ‘**car**’.

After filling click the button ‘**PREDICT**’.



Above figure shows the predicted value of our car with the given details.

8. ADVANTAGES AND DISADVANTAGES

a. ADVANTAGES

Used car resale value system using machine learning can predict the resale value of our vehicle in much accurate way because it takes many factors that affect the value of the vehicle into consideration.

Also, prediction is done within seconds so this system is fast comparatively.

1. Automation of Everything

Machine Learning is responsible for cutting the workload and time. By automating things, we let the algorithm do the hard work for us. Automation is now being done almost everywhere. The reason is that it is very reliable. Also, it helps us to think more creatively.

Due to ML, we are now designing more advanced computers. These computers can handle various Machine Learning models and algorithms efficiently. Even though automation is spreading fast, we still don’t completely rely on it. ML is slowly transforming the industry with its automation.

2. Wide Range of Applications

ML has a wide variety of applications. This means that we can apply ML on any of the major fields. ML has its role everywhere from medical, business, banking to science and tech. This helps to create more opportunities.

3. Scope of Improvement

Machine Learning is the type of technology that keeps on evolving. There is a lot of scope in ML to become the top technology in the future. The reason is, it has a lot of research areas in it. This helps us to improve both hardware and software.

4. Efficient Handling of Data

Machine Learning has many factors that make it reliable. One of them is data handling. ML plays the biggest role when it comes to data at this time. It can handle any type of data.

Machine Learning can be multidimensional or different types of data. It can process and analyse these data that normal systems can't. Data is the most important part of any Machine Learning model. Also, studying and handling of data is a field in itself.

b. DISADVANTAGES

1. Possibility of High Error

In ML, we can choose the algorithms based on accurate results. For that, we have to run the results on every algorithm. The main problem occurs in the training and testing of data. The data is huge, so sometimes removing errors becomes nearly impossible. These errors can cause a headache to users. Since the data is huge, the errors take a lot of time to resolve.

2. Algorithm Selection

The selection of an algorithm in Machine Learning is still a manual job. We have to run and test our data in all the algorithms. After that only we can decide what algorithm we want. We choose them on the basis of result accuracy. The process is very much time-consuming.

3. Data Acquisition

In ML, we constantly work on data. We take a huge amount of data for training and testing. This process can sometimes cause data inconsistency. The reason is some data constantly keep on updating. So, we have to wait for the new data to arrive. If not, the old and new data might give different results. That is not a good sign for an algorithm.

4. Time and Space

Many ML algorithms might take more time than you think. Even if it's the best algorithm it might sometimes surprise you. If your data is large and advanced, the system will take time. This may sometimes cause the consumption of more CPU power. Even with GPUs alongside, it sometimes becomes hectic. Also, the data might use more than the allotted space.

9. APPLICATIONS

The basic objective of this system is to let user have a fair idea of what the vehicle could cost them.

1. On-road price: User can get an approximate on-road price of any model, on one's fingertips.

2. Analytics: The user can understand the trends in the variation of the car prices with respect to age, usage of the car to better understand the how prices are influenced based on these factors.

3. Accurate pricing: Since the price of the car is predicted by the model using large amount of data, there are less chances of any unrealistic pricing.

10. CONCLUSION

With the increase of car ownership, used car market shows great potential. An accurate used car price evaluation is essential for the healthy development of used car market. The automobile resale system would prove useful in helping the potential seller that is giving them a predicted price for the car, thereby proving to be a very useful tool helping people to put their car for sale with the predicted value which is a very popular and trending niche in the used cars market.

Also, the system can be easily implemented providing an overall satisfaction to the customer

11. FUTURE SCOPE

Given the current working and design of the proposed systems, there is definitely a place for future enhancements. For example, we can use the system as a platform for selling our vehicle as well as buying vehicle after predicting the used car value.

With future generations of the product, modules such as the comparison and payment gateways can be implemented into the system, leading towards development of a proper business model which can then be used by businesses along with refined and interactive analytics that can be used to understand the trends in the prices and different automobile types.

12. BIBLIOGRAPHY

- https://www.timesnownews.com/auto/features/article/how-to-evaluate-a-used-car-before-buying-preparing-for-life-after-covid-19-lock-down/584628
- https://motoroctane.com/news/232931-how-to-evaluate-a-used-car-properly
- https://www.enjoyalgorithms.com/blog/car-resale-value-predictor-using-random-forest-regressor
- https://medium.com/analytics-vidhya/predicting-vehicle-price-with-random-forest-regressor-d1c272668be5
- https://www.scribbr.com/statistics/multiple-linear-regression/

13. APPENDIX

In [1]:

```
import pandas as pd
import numpy as np
import matplotlib as plt
from sklearn.preprocessing import LabelEncoder
import pickle
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score
```

In [2]:

```
import os, types
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage. It includes your
credentials.
# You might want to remove those credentials before you share the notebook.
client_fe70859e9d014af0bbd7e181bafb7b29 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='9YfKIR0JI4PQWgoJtiX6qD7NxSnOUTQ6_lRl9qYL8UtK',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3.private.us.cloud-object-storage.appdomain.cloud')

body = client_fe70859e9d014af0bbd7e181bafb7b29.get_object(Bucket='usedcarvaluation-dono
tdelete-pr-su7qxvvlir2hn6',Key='autos.xlsx')['Body']

df = pd.read_excel(body.read())
df.head()
```

Out[2]:

	dateCrawled	name	seller	offerType	price	abtest	vehicleTy
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480.0	test	Na
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300.0	test	cou
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"	privat	Angebot	9800.0	test	s
3	2016-03-17 16:54:04	GOLF_4_1_4__3TÜRER	privat	Angebot	1500.0	test	kleinwag
4	2016-03-31 17:25:20	Skoda_Fabia_1.4_TDI_PD_Classic	privat	Angebot	3600.0	test	kleinwag

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371539 entries, 0 to 371538
Data columns (total 20 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   dateCrawled          371539 non-null  object
 1   name                 371524 non-null  object
 2   seller               371538 non-null  object
 3   offerType            371538 non-null  object
 4   price                371538 non-null  float64
 5   abtest               371538 non-null  object
 6   vehicleType          333669 non-null  object
 7   yearOfRegistration   371537 non-null  float64
 8   gearbox              351329 non-null  object
 9   powerPS              371538 non-null  float64
 10  model                351054 non-null  object
 11  kilometer            371538 non-null  object
 12  monthOfRegistration  371537 non-null  float64
 13  fuelType             338151 non-null  object
 14  brand                371537 non-null  object
 15  notRepairedDamage    299477 non-null  object
 16  dateCreated          371537 non-null  datetime64[ns]
 17  nrOfPictures         371537 non-null  float64
 18  postalCode           371537 non-null  float64
 19  lastSeen             371537 non-null  datetime64[ns]
dtypes: datetime64[ns](2), float64(6), object(12)
memory usage: 56.7+ MB
```

In [4]:

```
df.shape
```

Out[4]:

```
(371539, 20)
```

In [5]:

```
df.isnull().sum()
```

Out[5]:

```
dateCrawled             0
name                   15
seller                  1
offerType               1
price                   1
abtest                  1
vehicleType         37870
yearOfRegistration      2
gearbox             20210
powerPS                 1
model               20485
kilometer               1
monthOfRegistration     2
fuelType            33388
brand                   2
notRepairedDamage   72062
dateCreated             2
nrOfPictures            2
postalCode              2
lastSeen                2
dtype: int64
```

In [6]:

```
df=df.drop(['offerType','seller'],axis=1)
```

In [7]:

```
#Cars having power less than 50ps and above 900ps seems a little suspicious,
#let's remove them and see what we've got now
df = df[(df.powerPS > 50) & (df.powerPS < 900)]
print(df.shape)
#around 50000 cars ahave been removed which could have inrouduced error to our data
```

```
(319717, 18)
```

In [8]:

```
#simlarly, filtering our the cars having registeration years not in the mentioned range
#print(df.shape)
df = df[(df.yearOfRegistration >= 1950) & (df.yearOfRegistration < 2017)]
print(df.shape)
# not much of a difference but still, 10000 rows have been reduced. it's better to
#get rid of faulty data instead of keeping them just to increase the size.
```

```
(309179, 18)
```

In [9]:

```
#removing irrelevant columns which are either the same for all the cars in teh dataset,
or can
#introduce bias, so removing them too.
df.drop(['name','abtest', 'dateCrawled', 'nrOfPictures', 'lastSeen',
         'postalCode', 'dateCreated'], axis='columns', inplace=True)
```

In [10]:

```
#dropping the duplicates from the dataframe and stroing it in a new df.
#here all rows having same value in all the mentioned columns will be deleted and by de
fault,
#only first occurance of anysuch row is kept
new_df = df.copy()
new_df = new_df.drop_duplicates(['price','vehicleType','yearOfRegistration'
                         ,'gearbox','powerPS','model','kilometer','monthOfRegistration'
,'fuelType'
                         ,'notRepairedDamage'])

#after removing duplicates
print(new_df.shape)
```

```
(285151, 11)
```

In [11]:

```
#As the dataset contained some german words for many features, cahnging them to english
new_df.gearbox.replace(('manuell', 'automatik'), ('manual','automatic'), inplace=True)
new_df.fuelType.replace(('benzin','andere','elektro'),('petrol','others','electric'),in
place=True)
new_df.vehicleType.replace(('kleinwagen', 'cabrio','kombi','andere'),
                          ('small car','convertible','combination','others'),inplace=T
rue)
new_df.notRepairedDamage.replace(('ja','nein'),('Yes','No'),inplace=True)
```

In [12]:

```
#### Removing the outliers
new_df = new_df[(new_df.price >= 100) & (new_df.price <= 150000)]
```

In [13]:

```
#Filling NaN values for columns whose data might not be there with the information prov
ider,
#which might lead to some variance but our model
#but we will still be able to give some estimate to the user
new_df['notRepairedDamage'].fillna(value='not-declared', inplace=True)
new_df['fuelType'].fillna(value='not-declared', inplace=True)
new_df['gearbox'].fillna(value='not-declared', inplace=True)
new_df['vehicleType'].fillna(value='not-declared', inplace=True)
new_df['model'].fillna(value='not-declared', inplace=True)
```

In [14]:

```
new_df.isnull().sum()
```

Out[14]:

```
price                  0
vehicleType            0
yearOfRegistration     0
gearbox                0
powerPS                0
model                  0
kilometer              0
monthOfRegistration    0
fuelType               0
brand                  0
notRepairedDamage      0
dtype: int64
```

In [15]:

```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 278583 entries, 1 to 371538
Data columns (total 11 columns):
 #   Column               Non-Null Count   Dtype
---  ------               --------------   -----
 0   price                278583 non-null  float64
 1   vehicleType          278583 non-null  object
 2   yearOfRegistration   278583 non-null  float64
 3   gearbox              278583 non-null  object
 4   powerPS              278583 non-null  float64
 5   model                278583 non-null  object
 6   kilometer            278583 non-null  object
 7   monthOfRegistration  278583 non-null  float64
 8   fuelType             278583 non-null  object
 9   brand                278583 non-null  object
 10  notRepairedDamage    278583 non-null  object
dtypes: float64(4), object(7)
memory usage: 25.5+ MB
```

In [16]:

```
new_df.head()
```

Out[16]:

	price	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthO
1	18300.0	coupe	2011.0	manual	190.0	not-declared	125000	
2	9800.0	suv	2004.0	automatic	163.0	grand	125000	
3	1500.0	small car	2001.0	manual	75.0	golf	150000	
4	3600.0	small car	2008.0	manual	69.0	fabia	90000	
5	650.0	limousine	1995.0	manual	102.0	3er	150000	

In [17]:

```
lb=LabelEncoder()
new_df['vehicleType']=lb.fit_transform(new_df['vehicleType'])
```

In [18]:

```
new_df['gearbox']=lb.fit_transform(new_df['gearbox'])
```

In [19]:

```
new_df['model']=new_df['model'].astype('str')
```

In [20]:

```
new_df['model']=lb.fit_transform(new_df['model'])
```

In [21]:

```
new_df['fuelType']=lb.fit_transform(new_df['fuelType'])
```

In [22]:

```
new_df['brand']=lb.fit_transform(new_df['brand'])
```

In [23]:

```
new_df['notRepairedDamage']=lb.fit_transform(new_df['notRepairedDamage'])
```

In [24]:

```
new_df.head()
```

Out[24]:

	price	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRe
1	18300.0	3	2011.0	1	190.0	162	125000	
2	9800.0	8	2004.0	0	163.0	118	125000	
3	1500.0	7	2001.0	1	75.0	117	150000	
4	3600.0	7	2008.0	1	69.0	102	90000	
5	650.0	4	1995.0	1	102.0	11	150000	

In [25]:

```
x = new_df.drop('price',axis=1)
y = new_df['price']
```

In [26]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3, random_state =
10)
```

In [27]:

```
rf=RandomForestRegressor()
rf.fit(x_train,y_train)
ypred=(rf.predict(x_test))
score=r2_score(y_test,ypred)
print('***Random Forest Regressor Model***')
print('Score for Random Forest Regressor Model is {}'.format(score))
```

```
***Random Forest Regressor Model***
Score for Random Forest Regressor Model is 0.8668253092006806
```

In [57]:

```
#pickle.dump(rf,open("model.pkl","wb"))
```

In [28]:

```
!pip install -U ibm-watson-machine-learning
```

In [29]:

```
from ibm_watson_machine_learning import APIClient
import json
import numpy as np
```

In [30]:

```
wml_credentials = {
    "apikey":"a0FHHGae4F6CFpI_7sydf0vuNpg0PuvgeGkNbuVKb_5v",
    "url":"https://us-south.ml.cloud.ibm.com"
}
```

In [31]:

```
wml_client = APIClient(wml_credentials)
wml_client.spaces.list()
```

```
Python 3.7 and 3.8 frameworks are deprecated and will be removed in a futu
re release. Use Python 3.9 framework instead.
Note: 'limit' is not provided. Only first 50 records will be displayed if
the number of records exceed 50
------------------------------------  ---------------  -------------------
-----
ID                                    NAME             CREATED
72d92135-a643-4139-8a8f-32233b0cff10  USED_CAR RESALE  2022-03-28T06:01:5
9.989Z
------------------------------------  ---------------  -------------------
-----
```

In [32]:

```
SPACE_ID="72d92135-a643-4139-8a8f-32233b0cff10"
```

In [33]:

```
wml_client.set.default_space(SPACE_ID)
```

Out[33]:

```
'SUCCESS'
```

In [34]:

```
wml_client.software_specifications.list()
```

```
----------------------------  ------------------------------------  ----
NAME                          ASSET_ID                              TYPE
default_py3.6                 0062b8c9-8b7d-44a0-a9b9-46c416adcbd9  base
pytorch-onnx_1.3-py3.7-edt    069ea134-3346-5748-b513-49120e15d288  base
scikit-learn_0.20-py3.6       09c5a1d0-9c1e-4473-a344-eb7b665ff687  base
spark-mllib_3.0-scala_2.12    09f4cff0-90a7-5899-b9ed-1ef348aebdee  base
pytorch-onnx_rt22.1-py3.9     0b848dd4-e681-5599-be41-b5f6fccc6471  base
ai-function_0.1-py3.6         0cdb0f1e-5376-4f4d-92dd-da3b69aa9bda  base
shiny-r3.6                    0e6e79df-875e-4f24-8ae9-62dcc2148306  base
tensorflow_2.4-py3.7-horovod  1092590a-307d-563d-9b62-4eb7d64b3f22  base
pytorch_1.1-py3.6             10ac12d6-6b30-4ccd-8392-3e922c096a92  base
tensorflow_1.15-py3.6-ddl     111e41b3-de2d-5422-a4d6-bf776828c4b7  base
runtime-22.1-py3.9            12b83a17-24d8-5082-900f-0ab31fbfd3cb  base
scikit-learn_0.22-py3.6       154010fa-5b3b-4ac1-82af-4d5ee5abbc85  base
default_r3.6                  1b70aec3-ab34-4b87-8aa0-a4a3c8296a36  base
pytorch-onnx_1.3-py3.6        1bc6029a-cc97-56da-b8e0-39c3880dbbe7  base
pytorch-onnx_rt22.1-py3.9-edt 1d362186-7ad5-5b59-8b6c-9d0880bde37f  base
tensorflow_2.1-py3.6          1eb25b84-d6ed-5dde-b6a5-3fbdf1665666  base
tensorflow_2.4-py3.8-horovod  217c16f6-178f-56bf-824a-b19f20564c49  base
runtime-22.1-py3.9-cuda       26215f05-08c3-5a41-a1b0-da66306ce658  base
do_py3.8                      295addb5-9ef9-547e-9bf4-92ae3563e720  base
autoai-ts_3.8-py3.8           2aa0c932-798f-5ae9-abd6-15e0c2402fb5  base
tensorflow_1.15-py3.6         2b73a275-7cbf-420b-a912-eae7f436e0bc  base
pytorch_1.2-py3.6             2c8ef57d-2687-4b7d-acce-01f94976dac1  base
spark-mllib_2.3               2e51f700-bca0-4b0d-88dc-5c6791338875  base
pytorch-onnx_1.1-py3.6-edt    32983cea-3f32-4400-8965-dde874a8d67e  base
spark-mllib_3.0-py37          36507ebe-8770-55ba-ab2a-eafe787600e9  base
spark-mllib_2.4               390d21f8-e58b-4fac-9c55-d7ceda621326  base
xgboost_0.82-py3.6            39e31acd-5f30-41dc-ae44-60233c80306e  base
pytorch-onnx_1.2-py3.6-edt    40589d0e-7019-4e28-8daa-fb03b6f4fe12  base
default_r36py38               41c247d3-45f8-5a71-b065-8580229facf0  base
autoai-ts_rt22.1-py3.9        4269d26e-07ba-5d40-8f66-2d495b0c71f7  base
autoai-obm_3.0                42b92e18-d9ab-567f-988a-4240ba1ed5f7  base
pmml-3.0_4.3                  493bcb95-16f1-5bc5-bee8-81b8af80e9c7  base
spark-mllib_2.4-r_3.6         49403dff-92e9-4c87-a3d7-a42d0021c095  base
xgboost_0.90-py3.6            4ff8d6c2-1343-4c18-85e1-689c965304d3  base
pytorch-onnx_1.1-py3.6        50f95b2a-bc16-43bb-bc94-b0bed208c60b  base
autoai-ts_3.9-py3.8           52c57136-80fa-572e-8728-a5e7cbb42cde  base
spark-mllib_2.4-scala_2.11    55a70f99-7320-4be5-9fb9-9edb5a443af5  base
spark-mllib_3.0               5c1b0ca2-4977-5c2e-9439-ffd44ea8ffe9  base
autoai-obm_2.0                5c2e37fa-80b8-5e77-840f-d912469614ee  base
spss-modeler_18.1             5c3cad7e-507f-4b2a-a9a3-ab53a21dee8b  base
cuda-py3.8                    5d3232bf-c86b-5df4-a2cd-7bb870a1cd4e  base
autoai-kb_3.1-py3.7           632d4b22-10aa-5180-88f0-f52dfb6444d7  base
pytorch-onnx_1.7-py3.8        634d3cdc-b562-5bf9-a2d4-ea90a478456b  base
spark-mllib_2.3-r_3.6         6586b9e3-ccd6-4f92-900f-0f8cb2bd6f0c  base
tensorflow_2.4-py3.7          65e171d7-72d1-55d9-8ebb-f813d620c9bb  base
spss-modeler_18.2             687eddc9-028a-4117-b9dd-e57b36f1efa5  base
pytorch-onnx_1.2-py3.6        692a6a4d-2c4d-45ff-a1ed-b167ee55469a  base
spark-mllib_2.3-scala_2.11    7963efe5-bbec-417e-92cf-0574e21b4e8d  base
spark-mllib_2.4-py37          7abc992b-b685-532b-a122-a396a3cdbaab  base
caffe_1.0-py3.6               7bb3dbe2-da6e-4145-918d-b6d84aa93b6b  base
----------------------------  ------------------------------------  ----
Note: Only first 50 records were displayed. To display more use 'limit' pa
rameter.
```

In [35]:

```
MODEL_NAME = 'UsedCarModel1'
DEPLOYMENT_NAME = 'used_car_deploy_3.8'
Life_MODEL = rf
```

In [36]:

```
# Set Python Version
software_spec_uid = wml_client.software_specifications.get_id_by_name('default_py3.8')

# Setup model meta
model_props = {
    wml_client.repository.ModelMetaNames.NAME: MODEL_NAME,
    wml_client.repository.ModelMetaNames.TYPE: 'scikit-learn_0.23',
    wml_client.repository.ModelMetaNames.SOFTWARE_SPEC_UID: software_spec_uid
}
```

In [37]:

```
#Save model
model_details = wml_client.repository.store_model(
    model=Life_MODEL,
    meta_props=model_props,
    training_data=x_train,
    training_target=y_train
)
```

```
Note: Warnings!! :  Software specification default_py3.8 specified for the
wml_model is deprecated and will be removed in the future. We recommend yo
u use runtime-22.1-py3.9 instead. For details see Supported Frameworks htt
ps://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/pm_service_s
upported_frameworks.html
```

In [38]:

```
model_details
```

Out[38]:


```
{'entity': {'hybrid_pipeline_software_specs': [],
  'label_column': 'price',
  'software_spec': {'id': 'ab9e1b80-f2ce-592c-a7d2-4f2344f77194',
   'name': 'default_py3.8'},
  'training_data_references': [{'connection': {'access_key_id': 'not_appli
cable',
     'endpoint_url': 'not_applicable',
     'secret_access_key': 'not_applicable'},
    'id': '1',
    'location': {},
    'schema': {'fields': [{'name': 'vehicleType', 'type': 'int64'},
      {'name': 'yearOfRegistration', 'type': 'float64'},
      {'name': 'gearbox', 'type': 'int64'},
      {'name': 'powerPS', 'type': 'float64'},
      {'name': 'model', 'type': 'int64'},
      {'name': 'kilometer', 'type': 'object'},
      {'name': 'monthOfRegistration', 'type': 'float64'},
      {'name': 'fuelType', 'type': 'int64'},
      {'name': 'brand', 'type': 'int64'},
      {'name': 'notRepairedDamage', 'type': 'int64'}],
     'id': '1',
     'type': 'DataFrame'},
    'type': 's3'}],
  'type': 'scikit-learn_0.23'},
 'metadata': {'created_at': '2022-03-28T06:13:21.599Z',
  'id': 'd3ddba0b-e838-4033-ba00-2cce61d07947',
  'modified_at': '2022-03-28T06:19:17.526Z',
  'name': 'UsedCarModel1',
  'owner': 'IBMid-661003CYL0',
  'resource_key': '28c6a5ce-443b-4fdc-a1eb-ebd1c4418222',
  'space_id': '72d92135-a643-4139-8a8f-32233b0cff10'},
 'system': {'warnings': [{'message': 'Software specification default_py3.8
specified for the wml_model is deprecated and will be removed in the futur
e. We recommend you use runtime-22.1-py3.9 instead. For details see Suppor
ted Frameworks https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze
-data/pm_service_supported_frameworks.html'}]}}
```


In [39]:

```
model_uid = wml_client.repository.get_model_id(model_details)
```

In [40]:

```
model_uid
```

Out[40]:

```
'd3ddba0b-e838-4033-ba00-2cce61d07947'
```

In [41]:

```
wml_client.connections.list_datasource_types()
```