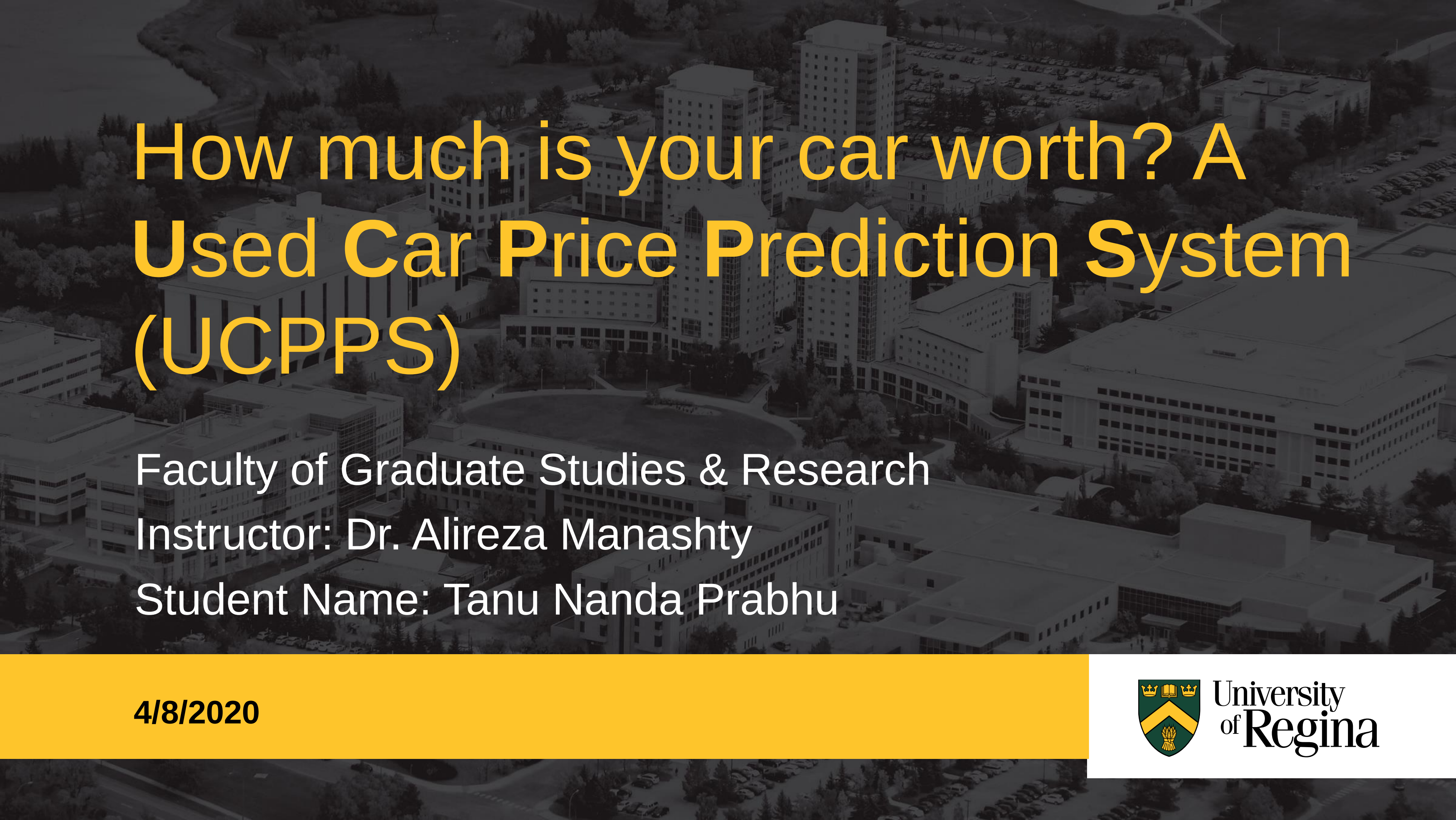


How much is your car worth? A **U**sed **C**ar **P**rice **P**rediction **S**ystem (UCPPS)

Faculty of Graduate Studies & Research

Instructor: Dr. Alireza Manashty

Student Name: Tanu Nanda Prabhu

4/8/2020

University of Regina

Table of Contents

- Introduction
- Problem Statement
- Data
- Model Planning and building
- Solution Overview
- Outcome
- Tools
- Team Roles
- Time line
- References

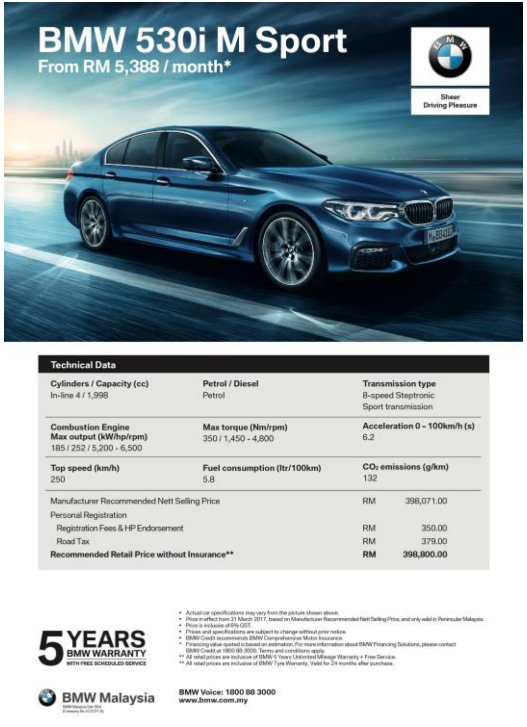
INTRODUCTION

•Vehicle value forecast is a significant errand particularly when the vehicle is used.

•The value of the car depends on several factors:

- Make (brand of the car)
- Power
- Number of kilometers it has been run
- Year of registration, and many more

•Better the features higher the price



Technical Data

Cylinders / Capacity (cc)	Petrol / Diesel	Transmission type
In-line 4 / 1,998	Petrol	8-speed Steptronic Sport transmission
Combustion Engine Max output (kW/hp/rpm)	**Max torque (Nm/rpm)**	**Acceleration 0 - 100km/h (s)**
185 / 252 / 5,200 - 6,500	350 / 1,450 - 4,800	6.2
Top speed (km/h)	**Fuel consumption (ltr/100km)**	**CO_2 emissions (g/km)**
250	5.8	132

Manufacturer Recommended Nett Selling Price	RM	398,071.00
Personal Registration		
Registration Fees & HP Endorsement	RM	350.00
Road Tax	RM	379.00
Recommended Retail Price without Insurance**	**RM**	**398,800.00**

- Actual car specifications may vary from the picture shown above.
- Price in effect from 31 March 2017, based on Manufacturer Recommended Nett Selling Price, and only valid in Peninsular Malaysia.
- Price is inclusive of 6% GST.
- Prices and specifications are subject to change without prior notice.
- BMW Credit recommends BMW Comprehensive Motor Insurance.

* Financing value quoted is based on estimation. For more information about BMW Financing Solutions, please contact BMW Credit at 1800 88 3000. Terms and conditions apply.

** All retail prices are inclusive of BMW 5 Years Unlimited Mileage Warranty + Free Service.

** All retail prices are inclusive of BMW Tyre Warranty. Valid for 24 months after purchase.

BMW Voice: 1800 88 3000
www.bmw.com.my

Image Credits: Manashty, 2020 [2]

PROBLEM STATEMENT

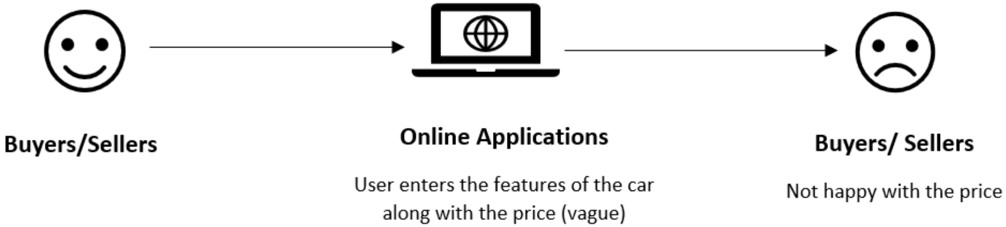
- Used Car Prices are important reflection of the economy and they greatly interest both buyers and sellers.

- A prediction model that estimates resale price based on car’s attributes or features is much more needed today.

- My analysis aims to determine which features of the car that may have the strongest statistical correlation with the price of the car.

Problem Statement

- Current Situation



Buyers/Sellers

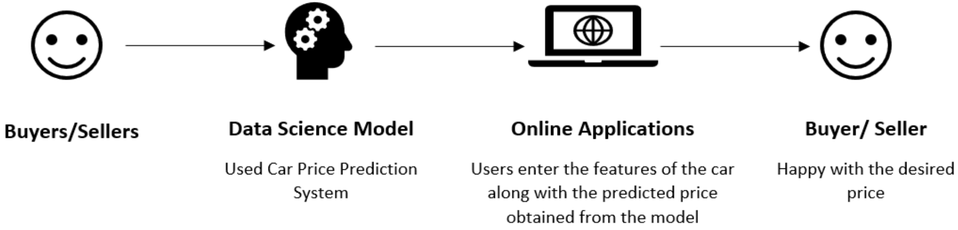
Online Applications

User enters the features of the car along with the price (vague)

Buyers/ Sellers

Not happy with the price

- Desired Situation



Buyers/Sellers

Data Science Model

Used Car Price Prediction System

Online Applications

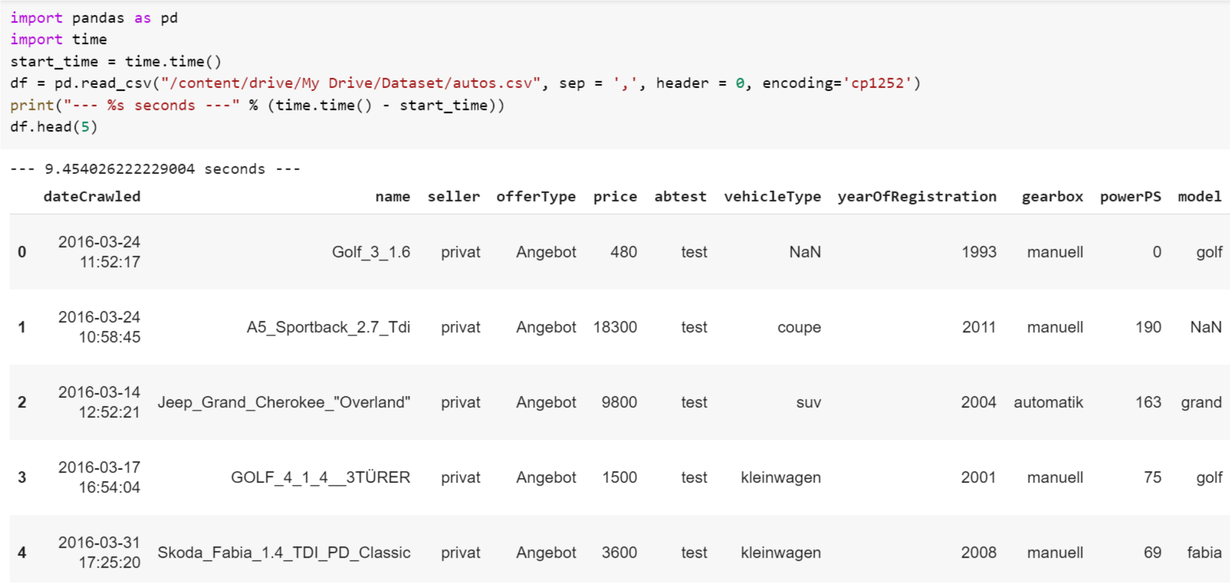
Users enter the features of the car along with the predicted price obtained from the model

Buyer/ Seller

Happy with the desired price

DATA

- The data set was chosen from data.world, data was originally scraped from e-bay [3]

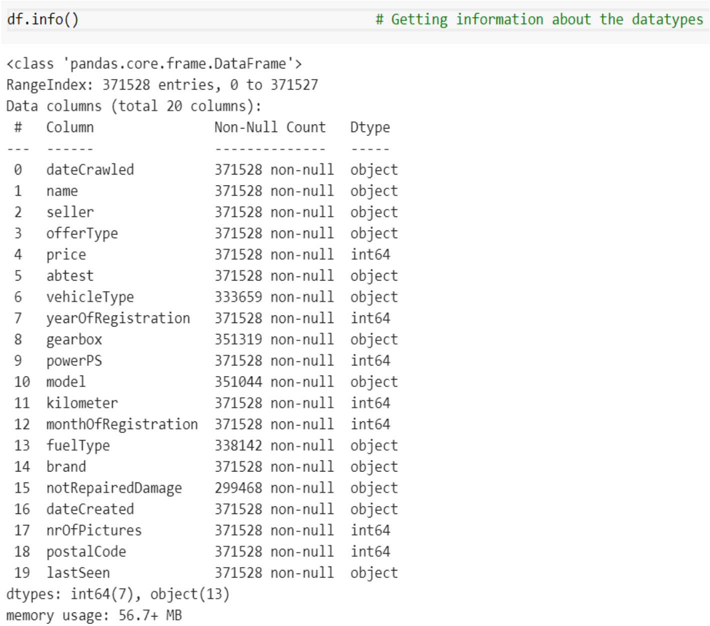


```python
import pandas as pd
import time
start_time = time.time()
df = pd.read_csv("/content/drive/My Drive/Dataset/autos.csv", sep = ',', header = 0, encoding='cp1252')
print("--- %s seconds ---" % (time.time() - start_time))
df.head(5)
```

```
--- 9.454026222229004 seconds ---
```

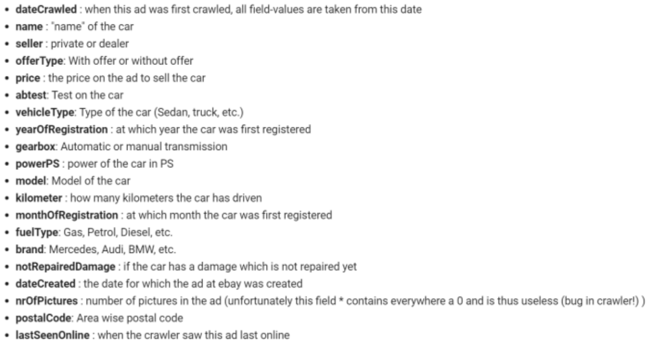
	dateCrawled	name	seller	offerType	price	abtest	vehicleType	yearOfRegistration	gearbox	powerPS	model
0	2016-03-24 11:52:17	Golf_3_1.6	privat	Angebot	480	test	NaN	1993	manuell	0	golf
1	2016-03-24 10:58:45	A5_Sportback_2.7_Tdi	privat	Angebot	18300	test	coupe	2011	manuell	190	NaN
2	2016-03-14 12:52:21	Jeep_Grand_Cherokee_"Overland"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

DATA STATISTICS

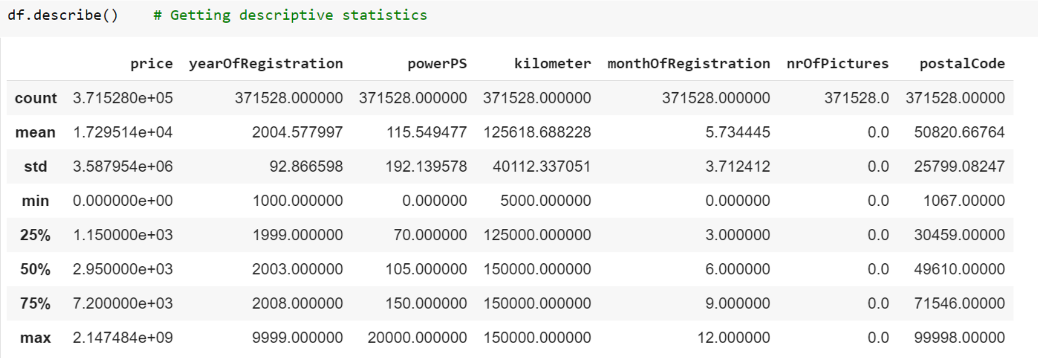


```
df.info()                                    # Getting information about the datatypes
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 371528 entries, 0 to 371527
Data columns (total 20 columns):
 #   Column               Non-Null Count   Dtype 
---  ------               --------------   ----- 
 0   dateCrawled          371528 non-null  object
 1   name                 371528 non-null  object
 2   seller               371528 non-null  object
 3   offerType            371528 non-null  object
 4   price                371528 non-null  int64 
 5   abtest               371528 non-null  object
 6   vehicleType          333659 non-null  object
 7   yearOfRegistration   371528 non-null  int64 
 8   gearbox              351319 non-null  object
 9   powerPS              371528 non-null  int64 
 10  model                351044 non-null  object
 11  kilometer            371528 non-null  int64 
 12  monthOfRegistration  371528 non-null  int64 
 13  fuelType             338142 non-null  object
 14  brand                371528 non-null  object
 15  notRepairedDamage    299468 non-null  object
 16  dateCreated          371528 non-null  object
 17  nrOfPictures         371528 non-null  int64 
 18  postalCode           371528 non-null  int64 
 19  lastSeen             371528 non-null  object
dtypes: int64(7), object(13)
memory usage: 56.7+ MB
```



- **dateCrawled** : when this ad was first crawled, all field-values are taken from this date
- **name** : "name" of the car
- **seller** : private or dealer
- **offerType**: With offer or without offer
- **price** : the price on the ad to sell the car
- **abtest**: Test on the car
- **vehicleType**: Type of the car (Sedan, truck, etc.)
- **yearOfRegistration** : at which year the car was first registered
- **gearbox**: Automatic or manual transmission
- **powerPS** : power of the car in PS
- **model**: Model of the car
- **kilometer** : how many kilometers the car has driven
- **monthOfRegistration** : at which month the car was first registered
- **fuelType**: Gas, Petrol, Diesel, etc.
- **brand**: Mercedes, Audi, BMW, etc.
- **notRepairedDamage** : if the car has a damage which is not repaired yet
- **dateCreated** : the date for which the ad at ebay was created
- **nrOfPictures** : number of pictures in the ad (unfortunately this field * contains everywhere a 0 and is thus useless (bug in crawler!))
- **postalCode**: Area wise postal code
- **lastSeenOnline** : when the crawler saw this ad last online



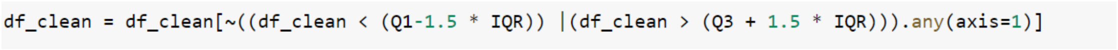
```
df.describe()    # Getting descriptive statistics
```

	price	yearOfRegistration	powerPS	kilometer	monthOfRegistration	nrOfPictures	postalCode
count	3.715280e+05	371528.000000	371528.000000	371528.000000	371528.000000	371528.0	371528.00000
mean	1.729514e+04	2004.577997	115.549477	125618.688228	5.734445	0.0	50820.66764
std	3.587954e+06	92.866598	192.139578	40112.337051	3.712412	0.0	25799.08247
min	0.000000e+00	1000.000000	0.000000	5000.000000	0.000000	0.0	1067.00000
25%	1.150000e+03	1999.000000	70.000000	125000.000000	3.000000	0.0	30459.00000
50%	2.950000e+03	2003.000000	105.000000	150000.000000	6.000000	0.0	49610.00000
75%	7.200000e+03	2008.000000	150.000000	150000.000000	9.000000	0.0	71546.00000
max	2.147484e+09	9999.000000	20000.000000	150000.000000	12.000000	0.0	99998.00000

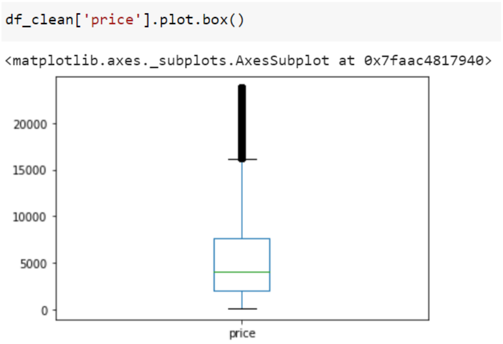
DATA EXPLORATION

Examining distribution of all variables

- Analyzing the outliers using the box-plot
- Using histogram density by plotting a bell curve
- Removing the outliers using IQR and Manual removing technique

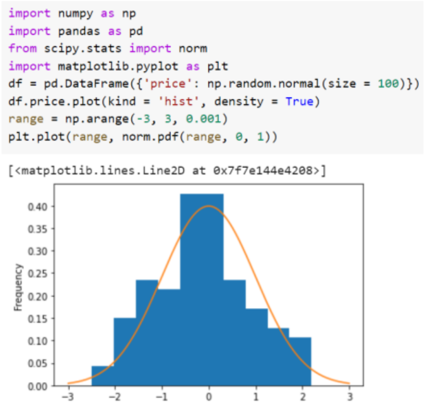


```
df_clean = df_clean[~((df_clean < (Q1-1.5 * IQR)) |(df_clean > (Q3 + 1.5 * IQR))).any(axis=1)]
```



```
df_clean['price'].plot.box()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7faac4817940>
```

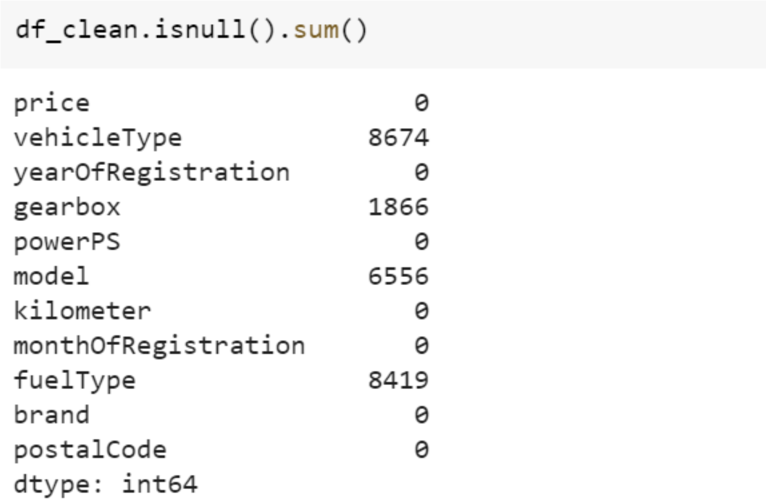


```
import numpy as np
import pandas as pd
from scipy.stats import norm
import matplotlib.pyplot as plt
df = pd.DataFrame({'price': np.random.normal(size = 100)})
df.price.plot(kind = 'hist', density = True)
range = np.arange(-3, 3, 0.001)
plt.plot(range, norm.pdf(range, 0, 1))
```

```
[<matplotlib.lines.Line2D at 0x7f7e144e4208>]
```

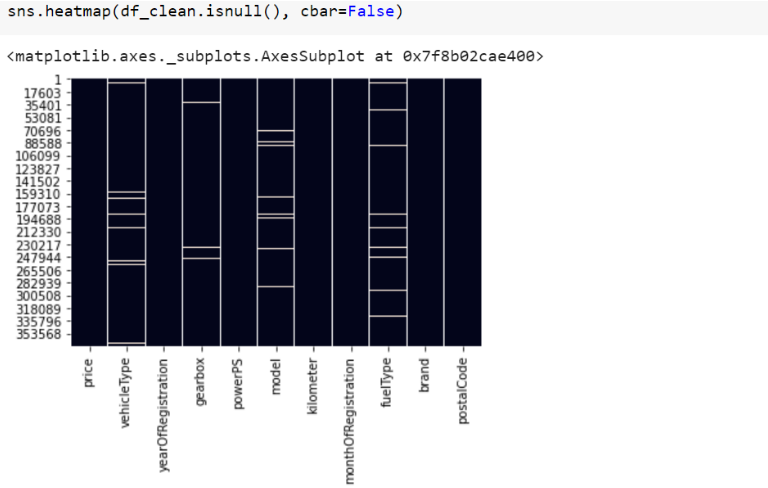
DATA EXPLORATION

Detecting missing values using plot



```
df_clean.isnull().sum()
```

```
price                    0
vehicleType           8674
yearOfRegistration       0
gearbox               1866
powerPS                  0
model                 6556
kilometer                0
monthOfRegistration      0
fuelType              8419
brand                    0
postalCode               0
dtype: int64
```



```
sns.heatmap(df_clean.isnull(), cbar=False)
```

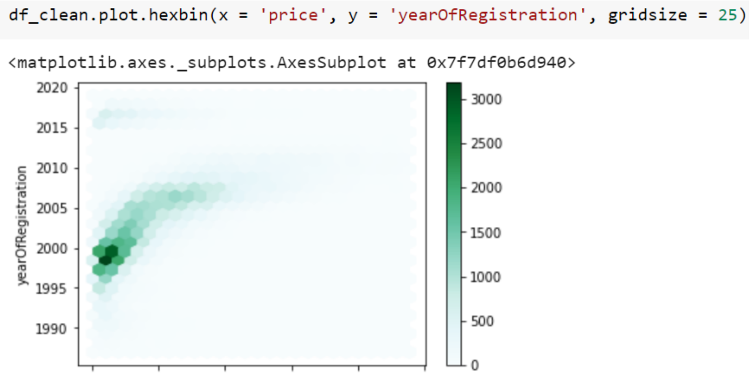
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f8b02cae400>
```

DATA VISUALIZATION

Data was visualized using different plots

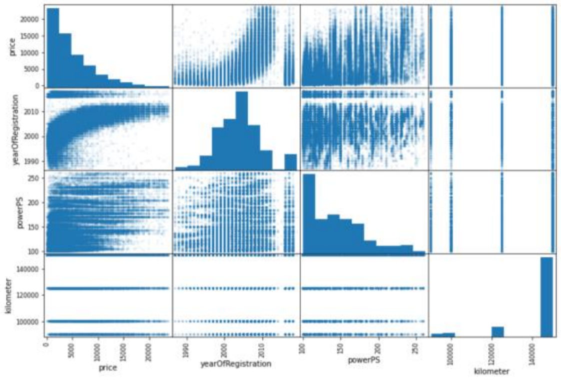
- Hexagonal bin plots

- Scatter matrix plot



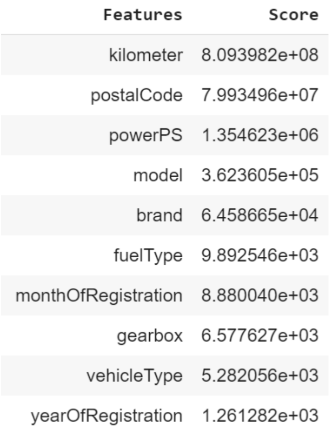
```
df_clean.plot.hexbin(x = 'price', y = 'yearOfRegistration', gridsize = 25)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f7df0b6d940>
```



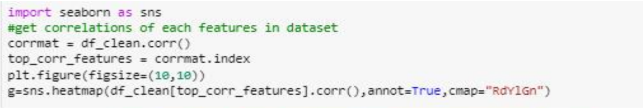
FEATURE SELECTION

- Chi Squared test

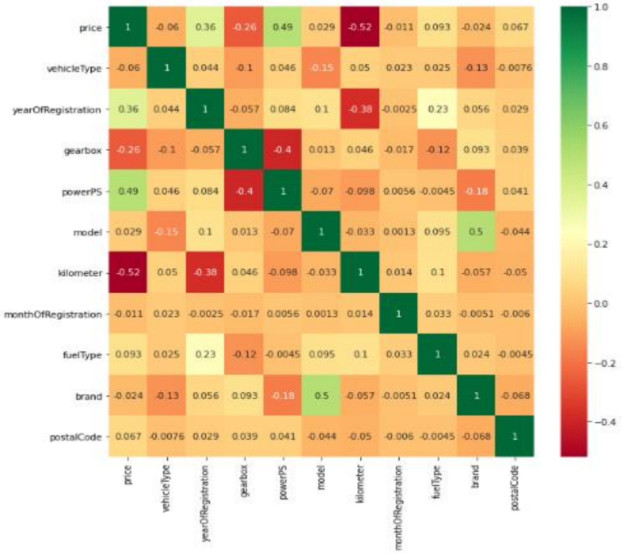


Features	Score
kilometer	8.093982e+08
postalCode	7.993496e+07
powerPS	1.354623e+06
model	3.623605e+05
brand	6.458665e+04
fuelType	9.892546e+03
monthOfRegistration	8.880040e+03
gearbox	6.577627e+03
vehicleType	5.282056e+03
yearOfRegistration	1.261282e+03

- Heat map



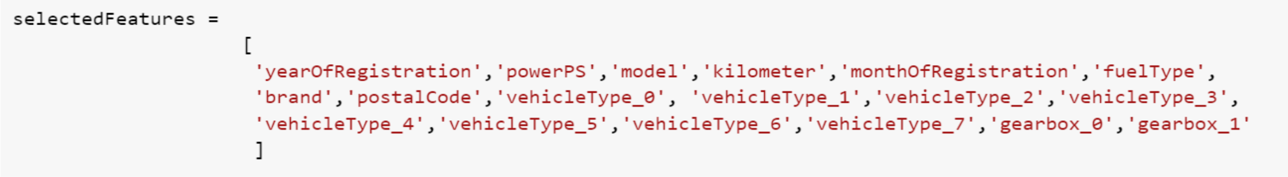
```
import seaborn as sns
#get correlations of each features in dataset
corrmat = df_clean.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(10,10))
g=sns.heatmap(df_clean[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```



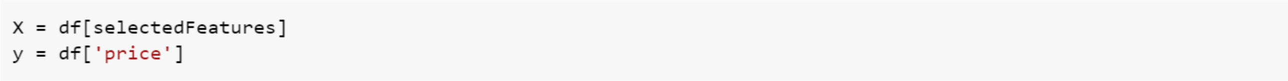
	price	vehicleType	yearOfRegistration	gearbox	powerPS	model	kilometer	monthOfRegistration	fuelType	brand	postalCode
price	1	-0.06	0.36	-0.26	0.49	0.029	-0.52	-0.011	0.093	-0.024	0.067
vehicleType	-0.06	1	0.044	-0.1	0.046	-0.15	0.05	0.023	0.025	-0.13	-0.0076
yearOfRegistration	0.36	0.044	1	-0.057	0.084	0.1	-0.38	-0.0025	0.23	0.056	0.029
gearbox	-0.26	-0.1	-0.057	1	-0.4	0.013	0.046	-0.017	-0.12	0.093	0.039
powerPS	0.49	0.046	0.084	-0.4	1	-0.07	-0.098	0.0056	-0.0045	-0.18	0.041
model	0.029	-0.15	0.1	0.013	-0.07	1	-0.033	0.0013	0.095	0.5	-0.044
kilometer	-0.52	0.05	-0.38	0.046	-0.098	-0.033	1	0.014	0.1	-0.057	-0.05
monthOfRegistration	-0.011	0.023	-0.0025	-0.017	0.0056	0.0013	0.014	1	0.033	-0.0051	-0.006
fuelType	0.093	0.025	0.23	-0.12	-0.0045	0.095	0.1	0.033	1	0.024	-0.0045
brand	-0.024	-0.13	0.056	0.093	-0.18	0.5	-0.057	-0.0051	0.024	1	-0.068
postalCode	0.067	-0.0076	0.029	0.039	0.041	-0.044	-0.05	-0.006	-0.0045	-0.068	1

Model Planning

- Variable selection



```
selectedFeatures = 
                    [
                     'yearOfRegistration','powerPS','model','kilometer','monthOfRegistration','fuelType',
                     'brand','postalCode','vehicleType_0', 'vehicleType_1','vehicleType_2','vehicleType_3',
                     'vehicleType_4','vehicleType_5','vehicleType_6','vehicleType_7','gearbox_0','gearbox_1'
                     ]
```



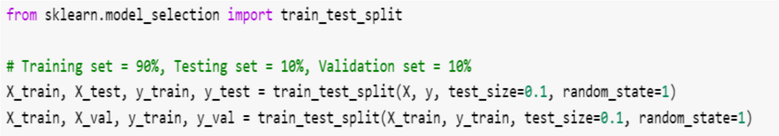
```
X = df[selectedFeatures]
y = df['price']
```

- Model Selection
  - Classification
  - **Regression**
  - Association Rules
  - Text/Image/Video Analysis

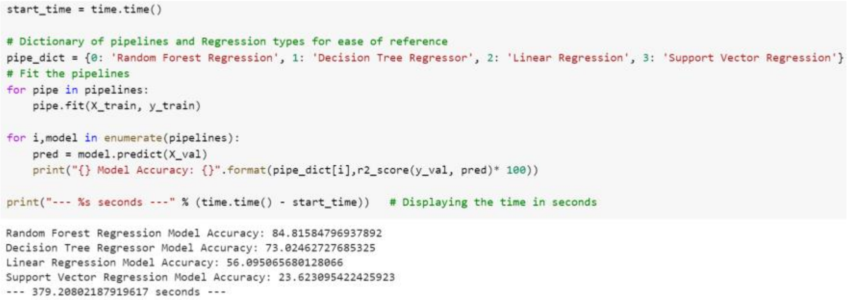
Model Building

- Splitting the dataset into three sets
  - **Training set** - 80%
  - **Validation set** - 10%
  - **Testing set** - 10%

- Choosing the learning algorithm with validation set



```
from sklearn.model_selection import train_test_split

# Training set = 90%, Testing set = 10%, Validation set = 10%
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=1)
X_train, X_val, y_train, y_val = train_test_split(X_train, y_train, test_size=0.1, random_state=1)
```

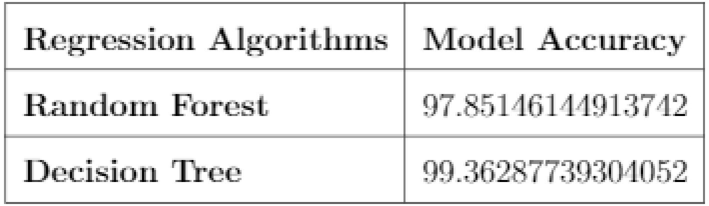


```
start_time = time.time()

# Dictionary of pipelines and Regression types for ease of reference
pipe_dict = {0: 'Random Forest Regression', 1: 'Decision Tree Regressor', 2: 'Linear Regression', 3: 'Support Vector Regression'}
# Fit the pipelines
for pipe in pipelines:
    pipe.fit(X_train, y_train)

for i,model in enumerate(pipelines):
    pred = model.predict(X_val)
    print("{} Model Accuracy: {}".format(pipe_dict[i],r2_score(y_val, pred)* 100))

print("--- %s seconds ---" % (time.time() - start_time))   # Displaying the time in seconds
```

```
Random Forest Regression Model Accuracy: 84.81584796937892
Decision Tree Regressor Model Accuracy: 73.02462727685325
Linear Regression Model Accuracy: 56.095065680128066
Support Vector Regression Model Accuracy: 23.623095422425923
--- 379.20802187919617 seconds ---
```

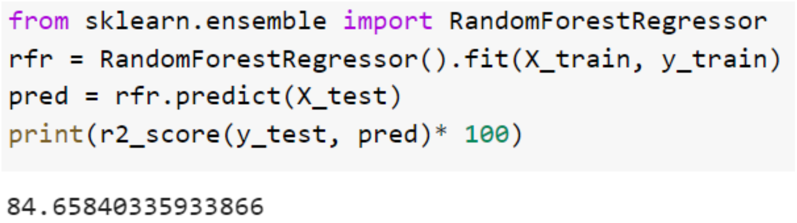
Model Building

- **Underfitting**
  - No underfitting



Regression Algorithms	Model Accuracy
Random Forest	97.85146144913742
Decision Tree	99.36287739304052

- **Overfitting**
  - Slightly overfitting

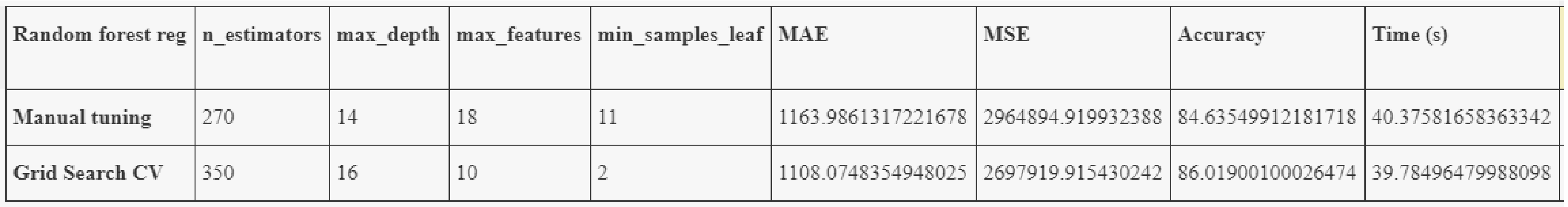


```
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor().fit(X_train, y_train)
pred = rfr.predict(X_test)
print(r2_score(y_test, pred)* 100)
```

```
84.65840335933866
```

Model Building

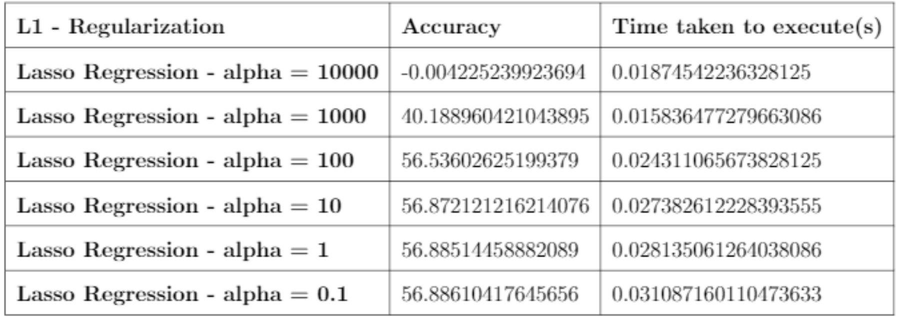
- **Hyper parameter tuning**
  - Manually
  - Grid search CV



Random forest reg	**n_estimators**	**max_depth**	**max_features**	**min_samples_leaf**	**MAE**	**MSE**	**Accuracy**	**Time (s)**
Manual tuning	270	14	18	11	1163.9861317221678	2964894.919932388	84.63549912181718	40.37581658363342
Grid Search CV	350	16	10	2	1108.0748354948025	2697919.915430242	86.01900100026474	39.78496479988098

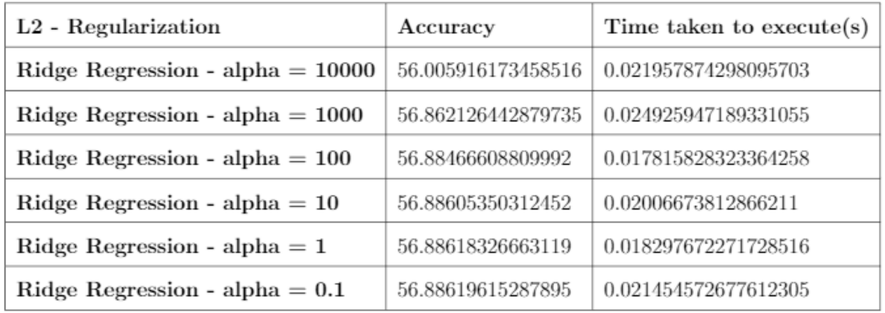
Model Building

- **L1 – Regularization**
  - **Lasso**



L1 - Regularization	**Accuracy**	**Time taken to execute(s)**
Lasso Regression - alpha = 10000	-0.004225239923694	0.01874542236328125
Lasso Regression - alpha = 1000	40.188960421043895	0.015836477279663086
Lasso Regression - alpha = 100	56.53602625199379	0.024311065673828125
Lasso Regression - alpha = 10	56.872121216214076	0.027382612228393555
Lasso Regression - alpha = 1	56.88514458882089	0.028135061264038086
Lasso Regression - alpha = 0.1	56.88610417645656	0.031087160110473633

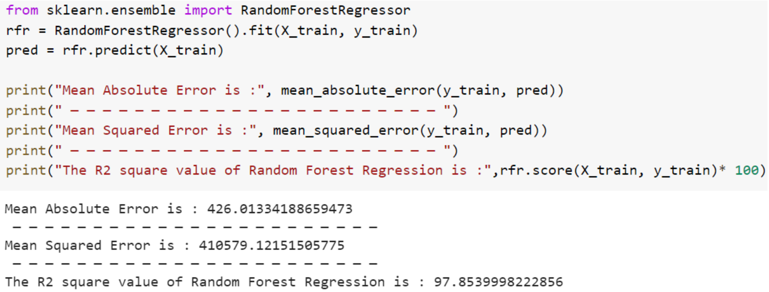
- **L2 - Regularization**
  - **Ridge**



L2 - Regularization	**Accuracy**	**Time taken to execute(s)**
Ridge Regression - alpha = 10000	56.005916173458516	0.021957874298095703
Ridge Regression - alpha = 1000	56.862126442879735	0.024925947189331055
Ridge Regression - alpha = 100	56.88466608809992	0.017815828323364258
Ridge Regression - alpha = 10	56.88605350312452	0.02006673812866211
Ridge Regression - alpha = 1	56.88618326663119	0.018297672271728516
Ridge Regression - alpha = 0.1	56.88619615287895	0.021454572677612305

Model Building

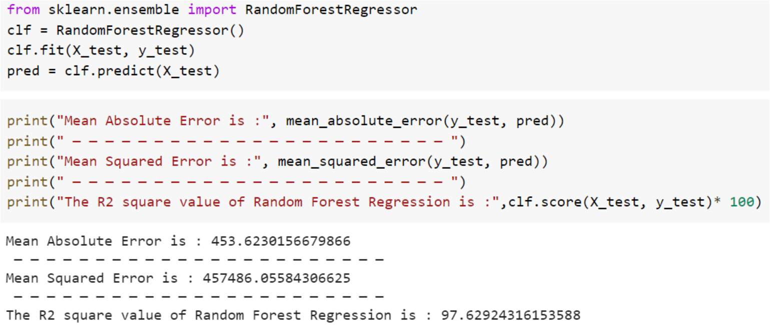
- MSE train



```python
from sklearn.ensemble import RandomForestRegressor
rfr = RandomForestRegressor().fit(X_train, y_train)
pred = rfr.predict(X_train)

print("Mean Absolute Error is :", mean_absolute_error(y_train, pred))
print(" - - - - - - - - - - - - - - - - - - - - - - - - ")
print("Mean Squared Error is :", mean_squared_error(y_train, pred))
print(" - - - - - - - - - - - - - - - - - - - - - - - - ")
print("The R2 square value of Random Forest Regression is :",rfr.score(X_train, y_train)* 100)
```

```
Mean Absolute Error is : 426.01334188659473
 - - - - - - - - - - - - - - - - - - - - - - - - 
Mean Squared Error is : 410579.12151505775
 - - - - - - - - - - - - - - - - - - - - - - - - 
The R2 square value of Random Forest Regression is : 97.8539998222856
```

- MSE test



```python
from sklearn.ensemble import RandomForestRegressor
clf = RandomForestRegressor()
clf.fit(X_test, y_test)
pred = clf.predict(X_test)
```

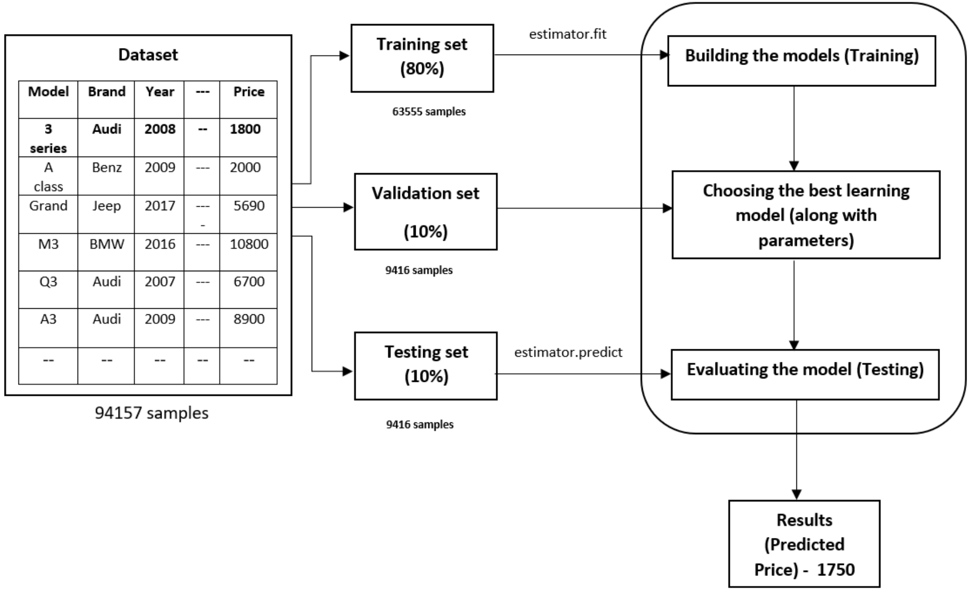
```python
print("Mean Absolute Error is :", mean_absolute_error(y_test, pred))
print(" - - - - - - - - - - - - - - - - - - - - - - - - ")
print("Mean Squared Error is :", mean_squared_error(y_test, pred))
print(" - - - - - - - - - - - - - - - - - - - - - - - - ")
print("The R2 square value of Random Forest Regression is :",clf.score(X_test, y_test)* 100)
```

```
Mean Absolute Error is : 453.6230156679866
 - - - - - - - - - - - - - - - - - - - - - - - - 
Mean Squared Error is : 457486.05584306625
 - - - - - - - - - - - - - - - - - - - - - - - - 
The R2 square value of Random Forest Regression is : 97.62924316153588
```

- T-test
  - t = -124.323, p = 0.02

SOLUTION OVERVIEW

- Category of data falls under supervised machine learning: Regression Estimator



Dataset

Model	Brand	Year	---	Price
3 series	**Audi**	**2008**	**--**	**1800**
A class	Benz	2009	---	2000
Grand	Jeep	2017	--- -	5690
M3	BMW	2016	---	10800
Q3	Audi	2007	---	6700
A3	Audi	2009	---	8900
--	--	--	--	--

94157 samples

Training set (80%)

63555 samples

Validation set (10%)

9416 samples

Testing set (10%)

9416 samples

estimator.fit

estimator.predict

Building the models (Training)

Choosing the best learning model (along with parameters)

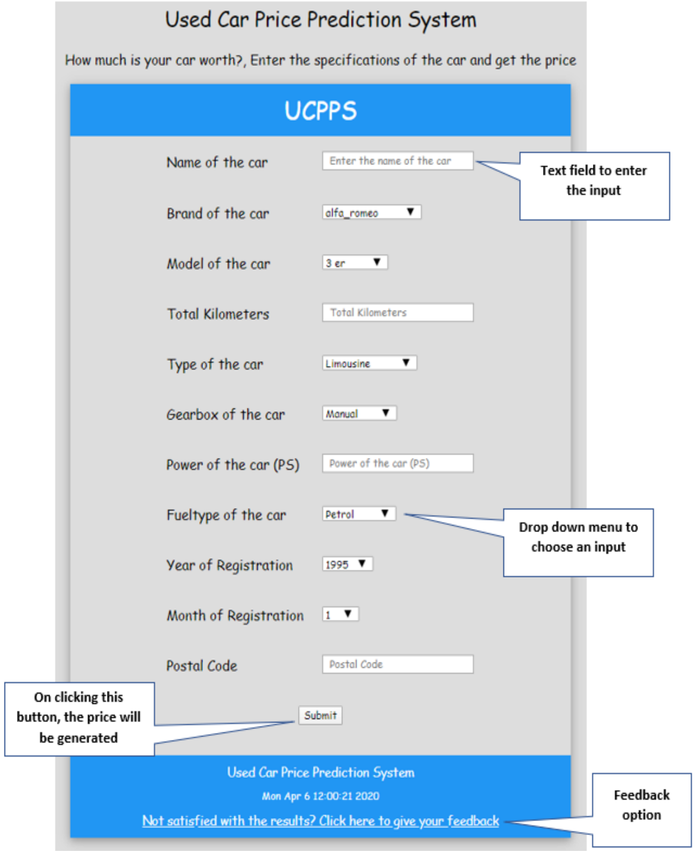
Evaluating the model (Testing)

Results (Predicted Price) - 1750

OUTCOME

- Component Of Project
  - Fully developed and Operational Cloud-based published website. (Copied from UR Courses)

- Link: https://ucpps.herokuapp.com/ucpps_app/home.html



Used Car Price Prediction System

How much is your car worth?, Enter the specifications of the car and get the price

UCPPS

Name of the car	Enter the name of the car
Brand of the car	alfa_romeo ▼
Model of the car	3 er ▼
Total Kilometers	Total Kilometers
Type of the car	Limousine ▼
Gearbox of the car	Manual ▼
Power of the car (PS)	Power of the car (PS)
Fueltype of the car	Petrol ▼
Year of Registration	1995 ▼
Month of Registration	1 ▼
Postal Code	Postal Code

Used Car Price Prediction System

Mon Apr 6 12:00:21 2020

Not satisfied with the results? Click here to give your feedback

Text field to enter the input

Drop down menu to choose an input

On clicking this button, the price will be generated

Feedback option

TOOLS

- Google Colab
- Anaconda Jupyter Notebook
- Python Programming (Obviously)
- Python Libraries (Pandas, NumPy, Matplotlib, scikit learn)
- Django Web framework (HTML, CSS, Bootstrap)
- GitHub
- Visual Studio Code
- Heroku

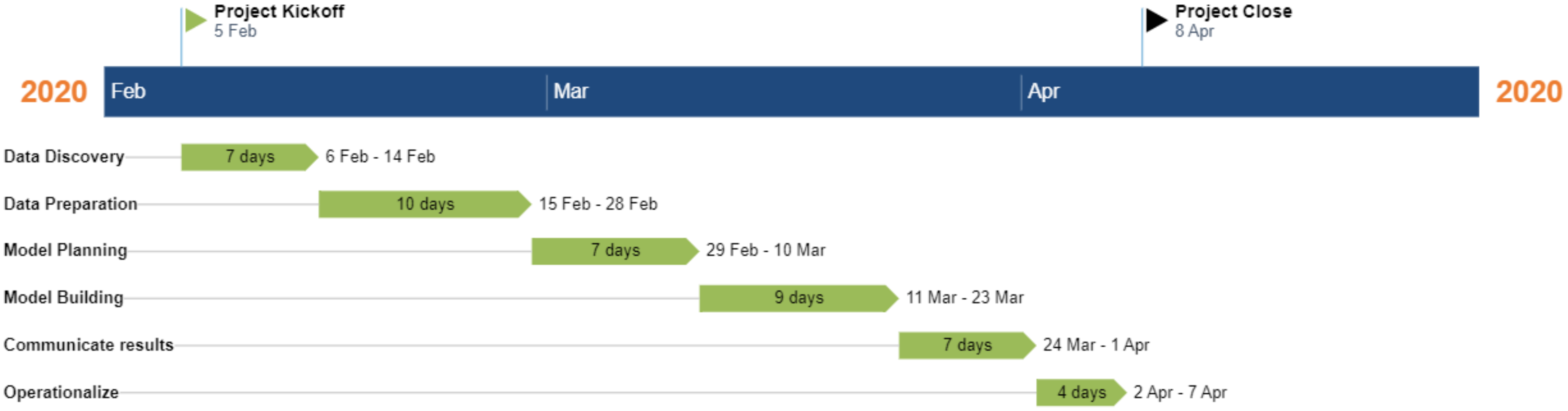
TEAM ROLES

- Data Collection, data understanding
- Model Design, model evaluation
- Code Documentation
- Deployment and building a functional website.

- 2 new things that took place recently
  - Uploaded my first YouTube video (Live coding) about code deployment
  - Deployed my first project on to the cloud

Image Credits: Medium

TIMELINE


Project Kickoff
5 Feb
Project Close
8 Apr
2020
Feb
Mar
Apr
2020
Data Discovery
7 days
6 Feb - 14 Feb
Data Preparation
10 days
15 Feb - 28 Feb
Model Planning
7 days
29 Feb - 10 Mar
Model Building
9 days
11 Mar - 23 Mar
Communicate results
7 days
24 Mar - 1 Apr
Operationalize
4 days
2 Apr - 7 Apr

REFERENCES

[1] Manashty, D. (2020). *Data Science Fundamentals - Chapter 1*. Presentation, University of Regina, Canada.

[2] *The all-new BMW 5-series (G30) launched – All You Need to Know*. (2020). [Image]. Retrieved 8 April 2020, from http://kensomuse.com/blog/2017/03/30/new-bmw-5-series-g30-launched-need-know/

[3] Leka, O. Used Cars Data - dataset by data-society. Retrieved 8 April 2020, from https://data.world/data-society/used-cars-data

Thank you