# USED CAR VALUE PREDICTION

## Introduction

Car Value Prediction Is Essential for Individuals and Companies

Each day, thousands of pre-owned cars are sold worldwide. Prediction of the second-hand vehicle price provides an important benchmark to both private buyer and the seller as well as business professionals such as car dealers, lenders and insurance companies.

Banks need to know the exact value of second-hand vehicles as they are mostly lienholders or they are transferring the loan from one person or another. Insurance companies alike need to be able to assess the value of the pre-owned vehicles, since they will be calculating premiums when they are making their risk assessment.

The used car market is also a large and strategically important market for car manufacturers since it is closely connected to the new car business. Trading-in used cars in new car retail sales and handling lease returns, repossessions and fleet returns from car rental companies necessitate car manufacturers to engage in the used car market. Therefore, car makers require sophisticated decision support systems to sustain the profitability of the used car business.

The necessity of prediction paved the way for now well-established companies like Edmunds, Kelley Blue Book, NADA Blue Book. These companies utilize statistical models on massive databases and they use machine learning algorithms to effectively predict the value of innumerable car brands and models, answering the market demand.

## Literature Survey

### Existing Problem

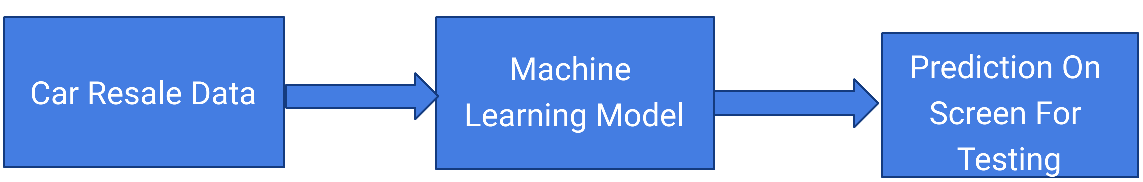
In many developed countries, it is common to lease a car rather than buying it . A lease is a binding contract between a buyer and a seller (or a third party – usually a bank, insurance firm or other financial institutions) in which the buyer must pay fixed installments for a pre-defined number of months/years to the seller/financier. 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 of seller/financiers to be able to predict the salvage value (residual value) of cars with accuracy. If the residual value is under-estimated by the seller/financier at the beginning, the installments will be higher for the clients who will certainly then opt for another seller/financier. If the residual value is over-estimated, the installments will be lower for the clients but then the seller/financier may have much difficulty at selling these high-priced used cars at this over-estimated residual value. Thus, we can see that estimating the price of used cars is of very high commercial importance as well.

## Proposed solution

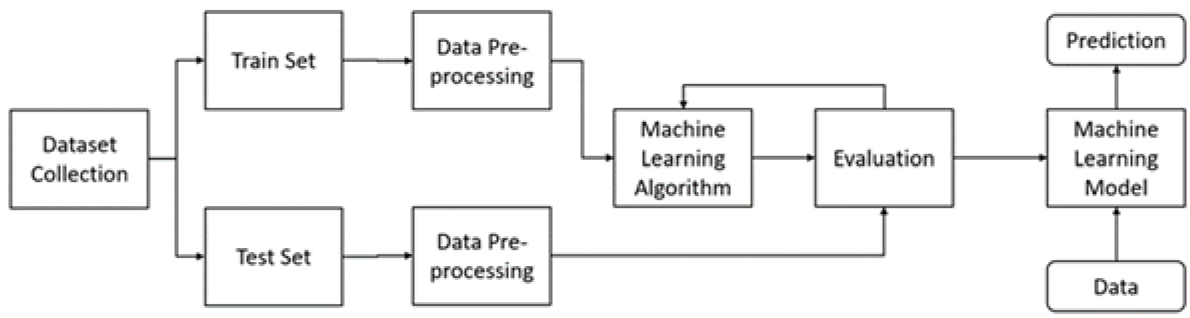
Considering the main factors which would affect the resale value of a vehicle a regression model is to be built that would give the nearest resale value of the vehicle. The main factors are the time in which vehicle got registered, number of KM's it drove, power, type of gear box, model of the car, any damage or repair, fuel type etc. we will be using various regression algorithms and algorithm with the best accuracy will be taken as solution , then it will be integrated to web based application where the user is notified with the status of his product.

# Theoretical Analysis

### Block Diagram:



## Machine Learning Workflow:




## Project Flow :

● User interacts with the UI (User Interface) to enter the current attrition data.

● Entered data are analyzed and predictions are made based on interpretation that whether employee will be attrited or not.

● Predictions are popped onto the UI. Data Collection The given data set is related to Taxi Fares. It was taken from the website kaggle.com. The website provides various datasets from various domains.

## Data Collection

The given data set is related to Taxi Fares. It was taken from the website kaggle.com. The website provides various datasets from various domains.

# Data pre-processing

## Importing required Libraries:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

**Pandas:** It is a python library mainly used for data manipulation.
**NumPy:** This python library is used for numerical analysis.
**Matplotlib and Seaborn:** Both are the data visualization library used for plotting graph which will help us for understanding the data.

## Importing the dataset:

```
[6] from google.colab import drive
    drive.mount('/content/drive')

    Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

[7] data=pd.read_csv(r"/content/drive/MyDrive/autos.csv", encoding='ISO-8859-1')

    data.head()
```

| | dateCrawled | name | seller | offerType | price | abtest | vehicleType | yearOfRegistration | gearbox | powerPS | model | kilometer | monthOfRegistration | fuelType | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 2016-03-24 11:52:17 | Golf_3_1.6 | privat | Angebot | 480 | test | NaN | 1993 | manuell | 0 | golf | 150000 | 0 | benzin | volks |
| 1 | 2016-03-24 10:58:45 | A5_Sportback_2.7_Tdi | privat | Angebot | 18300 | test | coupe | 2011 | manuell | 190 | NaN | 125000 | 5 | diesel | |
| 2 | 2016-03-14 12:52:21 | Jeep_Grand_Cherokee_"Overland" | privat | Angebot | 9800 | test | suv | 2004 | automatik | 163 | grand | 125000 | 8 | diesel | |
| 3 | 2016-03-17 16:54:04 | GOLF_4_1_4__3TÜRER | privat | Angebot | 1500 | test | kleinwagen | 2001 | manuell | 75 | golf | 150000 | 6 | benzin | volks |
| 4 | 2016-03-31 17:25:20 | Skoda_Fabia_1.4_TDI_PD_Classic | privat | Angebot | 3600 | test | kleinwagen | 2008 | manuell | 69 | fabia | 90000 | 7 | diesel | |



- You might have your data in .csv files, .excel files or .tsv files or something else. But the goal is the same in all cases. If you want to analyse that data using pandas, the first step will be to read it into a data structure that's compatible with pandas.

- Let's load a .csv data file into pandas. There is a function for it, called **read_csv().**We will need to locate the directory of the CSV file at first (it's more efficient to keep the dataset in the same directory as your program).

- Path names on Windows tend to have backslashes in them. But we want them to mean actual backslashes, not special characters.

## Taking care of Missing Data:

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously, you could remove the entire line of data but what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common ideas to handle the problem is to take a mean of all the values for continuous and for categorical we make use of mode values and replace the missing data.

1. We will be using isnull().any() method to see which column has missing values.

```
data.isnull().sum()
```

```
dateCrawled                0
name                       0
seller                     0
offerType                  0
price                      0
abtest                     0
vehicleType            37869
yearOfRegistration         0
gearbox                20209
powerPS                    0
model                  20484
kilometer                  0
monthOfRegistration        0
fuelType               33386
brand                      0
notRepairedDamage      72060
dateCreated                0
nrOfPictures               0
postalCode                 0
lastSeen                   0
dtype: int64
```

```
[10] var=['vehicleType','gearbox','model','fuelType','notRepairedDamage']
     for i in var:
         data[i].fillna(data[i].mode()[0],inplace=True)
```

```
[11] data.isnull().sum()
```

```
dateCrawled            0
name                   0
seller                 0
offerType              0
price                  0
abtest                 0
vehicleType            0
yearOfRegistration     0
gearbox                0
powerPS                0
model                  0
kilometer              0
monthOfRegistration    0
fuelType               0
brand                  0
notRepairedDamage      0
dateCreated            0
nrOfPictures           0
postalCode             0
lastSeen               0
dtype: int64
```

### Label encoding

Typically, any structured dataset includes multiple columns with combination of numerical as well as categorical variables. A machine can only understand the numbers. It cannot understand the text. That's essentially the case with [Machine Learning algorithms]() too. We need to convert each text category to numbers in order for the machine to process those using mathematical equations.

How should we handle categorical variables? There are Multiple way to handle, but will see one of it is LabelEncoding.

- **Label Encoding** is a popular encoding technique for handling categorical variables. In this technique, each label is assigned a unique integer based on alphabetical ordering.

Let's see how to implement label encoding in Python using the scikit-learn library.

As we have to convert only the text class category columns, we first select it then we will implement Label Encoding to it.

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
data1['abtest']=le.fit_transform(data1['abtest'])
data1['gearbox']=le.fit_transform(data1['gearbox'])
data1['notRepairedDamage']=le.fit_transform(data1['notRepairedDamage'])
#data1['name']=le.fit_transform(data1['name'])
data1['vehicleType']=le.fit_transform(data1['vehicleType'])
data1['model']=le.fit_transform(data1['model'])
data1['fuelType']=le.fit_transform(data1['fuelType'])
data1['brand']=le.fit_transform(data1['brand'])
```

```
[25] data1.head()
```

| | price | abtest | vehicleType | gearbox | powerPS | model | kilometer | fuelType | brand | notRepairedDamage | postalCode |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 480 | 1 | 6 | 1 | 0 | 118 | 150000 | 1 | 38 | 1 | 70435 |
| **1** | 18300 | 1 | 3 | 1 | 190 | 118 | 125000 | 3 | 1 | 0 | 66954 |
| **2** | 9800 | 1 | 7 | 0 | 163 | 119 | 125000 | 3 | 14 | 1 | 90480 |
| **3** | 1500 | 1 | 4 | 1 | 75 | 118 | 150000 | 1 | 38 | 1 | 91074 |
| **4** | 3600 | 1 | 4 | 1 | 69 | 103 | 90000 | 3 | 31 | 1 | 60437 |

### Feature Scaling

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

```
[78] X_train
```

```
array([[-1.03839956,  0.94770343,  0.        , ...,  0.42020484,
         0.        , -1.01260884],
       [-1.03839956,  0.94770343,  0.        , ..., -0.8192424 ,
         0.        , -0.23166588],
       [ 0.96302044,  0.94770343,  0.        , ..., -1.47542036,
         0.        ,  1.75743101],
       ...,
       [ 0.96302044,  0.94770343,  0.        , ..., -1.4025117 ,
         0.        , -0.64138185],
       [-1.03839956, -1.0902222 ,  0.        , ..., -0.67342508,
         0.        ,  1.3068934 ],
       [-1.03839956,  0.94770343,  0.        , ...,  1.22220012,
         0.        , -1.00161199]])
```

1. **Splitting Data into Train and Test:**

When you are working on a model and you want to train it, you obviously have a dataset. But after training, we have to test the model on some test dataset. For this, you will a dataset which is different from the training set you used earlier. But it might not always be possible to have so much data during the development phase. In such cases, the solution is to split the dataset into two sets, one for training and the other for testing.

But the question is, how do you split the data? You can't possibly manually split the dataset into two sets. And you also have to make sure you split the data in a random manner. To help us with this task, the Scikit library provides a tool, called the Model Selection library. There is a class in the library which is, 'train_test_split.' Using this we can easily split the dataset into the training and the testing datasets in various proportions.

The train-test split is a technique for evaluating the performance of a machine learning algorithm.

- Train Dataset: Used to fit the machine learning model.
- Test Dataset: Used to evaluate the fit machine learning model.

In general you can allocate 80% of the dataset to training set and the remaining 20% to test set.

We will create 4 sets— X_train (training part of the matrix of features), X_test (test part of the matrix of features), Y_train (training part of the dependent variables associated with the X train sets, and therefore also the same indices), Y_test (test part of the dependent variables associated with the X test sets, and therefore also the same indices.

There are a few other parameters that we need to understand before we use the class:

```
[75] 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, random_state=0)
```

```
[76] print(X_train.shape)
     print(X_test.shape)
     print(Y_train.shape)
     print(Y_test.shape)

     (83451, 9)
     (35765, 9)
     (83451, 1)
     (35765, 1)
```

2. **test_size —** this parameter decides the size of the data that has to be split as the test dataset. This is given as a fraction. For example, if you pass 0.5 as the value, the dataset will be split 50% as the test dataset
3. **train_size —** you have to specify this parameter only if you're not specifying the test_size. This is the same as test_size, but instead you tell the class what percent of the dataset you want to split as the training set.
4. **random_state —** here you pass an integer, which will act as the seed for the random number generator during the split. Or, you can also pass an instance of the **Random_state** class, which will become the number generator. If you don't pass anything, the **Random_state** instance used by np.random will be used instead.
5. Now split our dataset into train set and test using **train_test_split** class from scikit learn library.

## Model Building:

### Training and testing the model:

There are several Machine learning algorithms to be used depending on the data you are going to process such as images, sound, text, and numerical values.

## Linear Regression

```
from sklearn.linear_model import LinearRegression
lr=LinearRegression()
lr.fit(X_train,Y_train)
Y_test
```

| | price |
|---|---|
| 202570 | 10500 |
| 305150 | 1200 |
| 119342 | 2500 |
| 55477 | 2999 |
| 150590 | 750 |
| ... | ... |
| 109131 | 9800 |
| 210560 | 800 |
| 47630 | 850 |
| 359881 | 8200 |
| 143147 | 8500 |

35765 rows × 1 columns

```
[81] Y_pred_lr=lr.predict(X_test)
     Y_pred_lr
```

```
array([[3044.0604383 ],
       [2535.13631002],
       [4059.63674863],
       ...,
       [3119.49135875],
       [5279.65084948],
       [4030.7719486 ]])
```

```
[82] from sklearn.metrics import r2_score
     acc_lr=r2_score(Y_test,Y_pred_lr)
     acc_lr
```

```
0.32434078966627333
```

### Decision Tree Regressor

```
[83] from sklearn.tree import DecisionTreeRegressor
     dtr = DecisionTreeRegressor()
     dtr.fit(X_train, Y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=None,
                      max_features=None, 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, presort='deprecated',
                      random_state=None, splitter='best')
```

```
[84] Y_pred_dtr=dtr.predict(sc.fit_transform(X_test))
     Y_pred_dtr
```

```
array([10500.,  2000.,  6950., ...,   199.,  5500.,  9990.])
```

```
[85] from sklearn.metrics import r2_score
     acc_dtr=r2_score(Y_test,Y_pred_dtr)
     acc_dtr
```

```
0.38763499444228977
```

## Random Forest Regressor

```
from sklearn.ensemble import RandomForestRegressor
rfr=RandomForestRegressor()
rfr.fit(X_train,Y_train)
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. Please change the shape of y to (n_sam
  This is separate from the ipykernel package so we can avoid doing imports until
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=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=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

## Predict the values:

Once the model is trained, it's ready to make predictions. We can use the predict method on the model and pass x_test as a parameter to get the output as pred.

Notice that the prediction output is an array of real numbers corresponding to the input array.

```
[87] Y_pred_rfr=rfr.predict(X_test)
     Y_pred_rfr
```

```
array([9310.5  , 3326.5  , 6485.5  , ...,  834.59 , 6608.05 , 7909.785])
```

## Evaluation:

Finally, we need to check to see how well our model is performing on the test data. There are many evaluation techniques are there. For this, we evaluate r2_score produced by the model.

```
[88] from sklearn.metrics import r2_score
     acc_rfr=r2_score(Y_test,Y_pred_rfr)
     acc_rfr
```

```
0.609743438466716
```

## Saving a model:

Model is saved so it can be used in future and no need to train it again.

```
import pickle
pickle.dump(rfr,open('car.pkl','wb'))
```

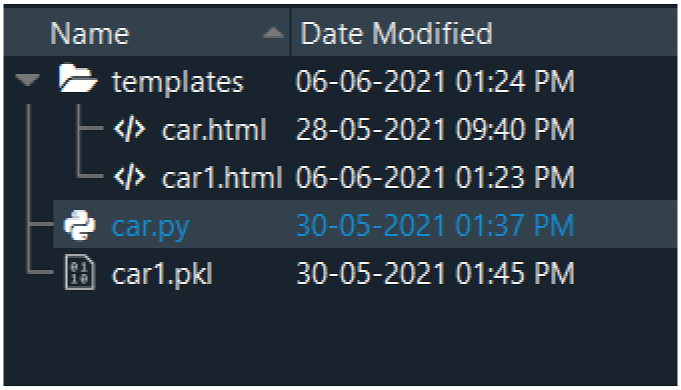
## Application Building:

Creating a HTML File, flask application.

- Build python code
- Importing Libraries
- Routing to the html Page
- Showcasing prediction on UI
- Run The app in local browser.

## Project Structure:

Create a Project folder that contains files as shown below




- We are building a Flask Application that needs HTML pages stored in the templates folder
- Templates folder contains index.html
- Static folder contains CSS and image files.

### Task 1: Importing Libraries

```
from flask import Flask, request, render_template
import pickle
import numpy as np
```

### Task 2: Routing to the html Page :

```python
app = Flask(__name__)
rfr=pickle.load(open('car1.pkl','rb'))
@app.route('/')
def home():
    return render_template('car.html')

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

    at = int(request.form["abtest"])
    vt = int(request.form["vehicleType"])
    gb = int(request.form["gearbox"])
    pps = int(request.form["powerPS"])
    m = int(request.form["model"])
    ft = int(request.form["fuelType"])
    b = int(request.form["brand"])
    nrd = int(request.form["notRepairedDamage"])
    pc = int(request.form["postalCode"])

    a=np.array([[at,vt,gb,pps,m,ft,b,nrd,pc]])
    print(a)
    result=rfr.predict(a)
    return render_template('car.html',x=result)
```

### Task 3: Main Function

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

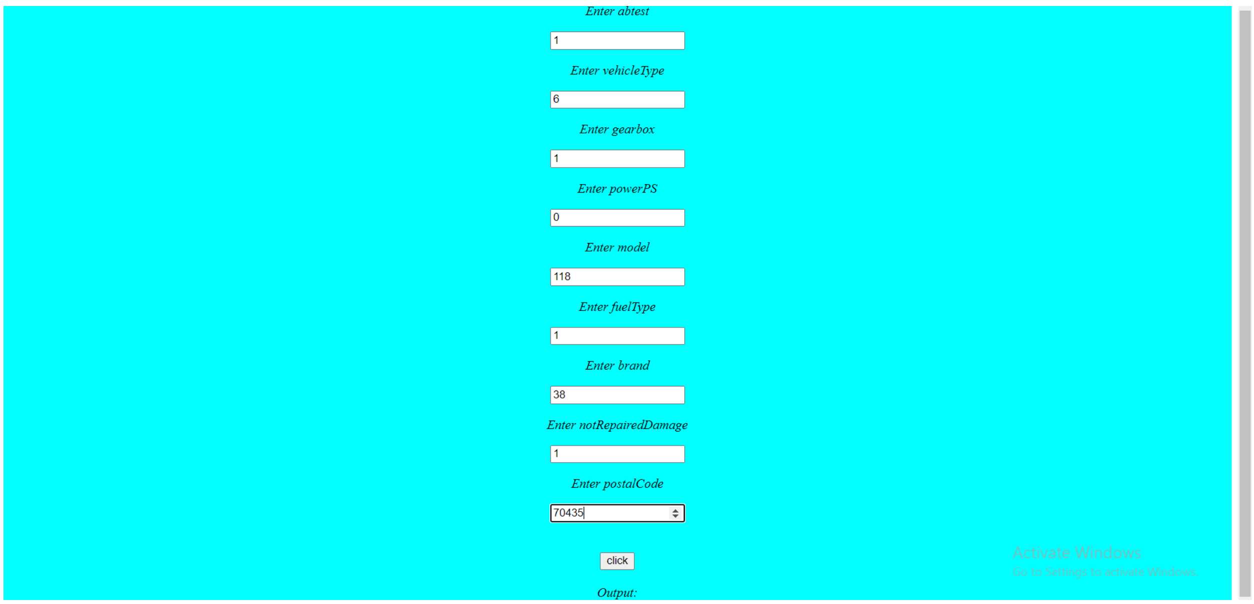
## Activity 3: Run the application

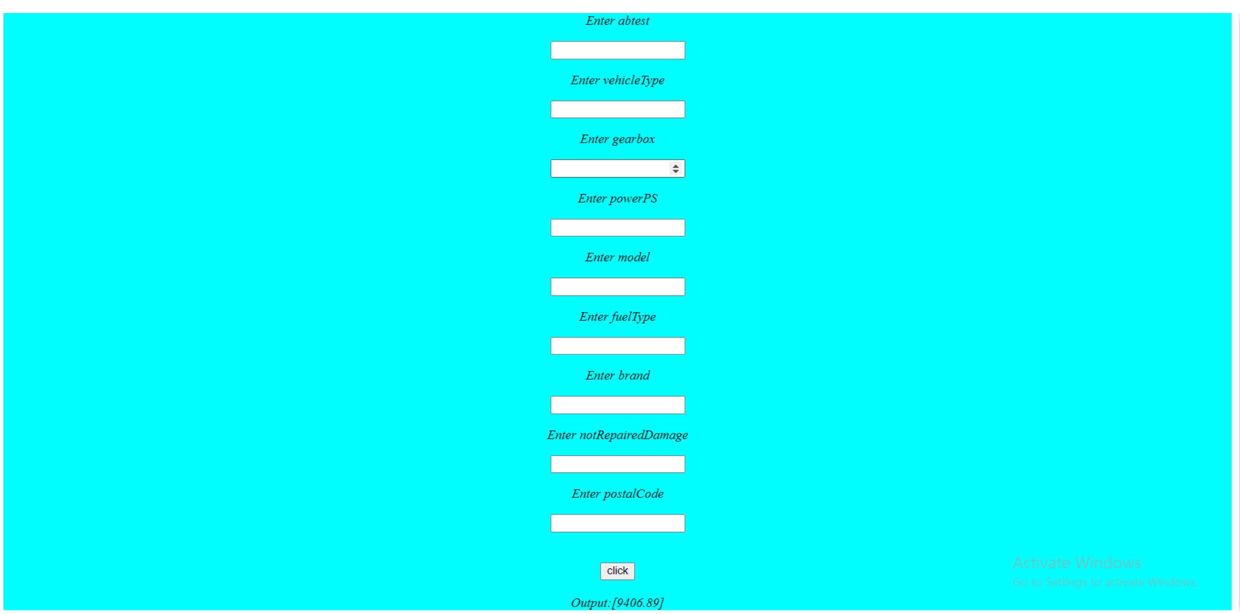
- Open the anaconda prompt from the start menu.
- Navigate to the folder where your app.py resides.
- Now type "python app.py" command.
- It will show the local host where your app is running on [http://127.0.0.1:5000/](http://127.0.0.1:5000/)
- Copy that local host URL and open that URL in the browser. It does navigate me to where you can view your web page.
- Enter the values, click on the predict button and see the result/prediction on the web page

```
Console 1/A

In [2]: runfile('C:/Users/Hi/Downloads/Flask/car.py', wdir='C:/Users/Hi/Downloads/Flask')
 * Serving Flask app "car" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

## ***Output Screen:***



## Findings and Suggestions

Through Exploratory Data Analysis,

- The R square value for training is 60.9% for single randomforest regression.
- The R square value for training is 38.7% for single Decision Tree regression.
- The R square value for training is 32.4% for Linear regression .
- we have predicted the price value by ensemble technique and compared the actual price with predicted price.

## Conclusion

This paper presented a machine learning model developing a stack of regressors predict the Car Resale Price. The model relies on eight predictors: abtest,vehicleType ,gearbox,powerPS,model,fuelType,brand, notRepairedDamage. An experiment was completed using ensemble stacking of the regressors and thereby proving that the combination of models will give high accuracy in the prediction

## Reference


- 1. www.kaggle.com
- 2. www.quora.com
- 3. www.wikkipedia.com