## ***Car Price Prediction***:

### **Importing the data**:

* **Problem Statement:** To determine the selling price of used cars.
* The dataset name “car data.csv” is downloaded from this site:  [Vehicle dataset from cardekho | Kaggle](https://www.kaggle.com/nehalbirla/vehicle-dataset-from-cardekho)
* This dataset contains information about used cars listed on [www.cardekho.com](http://www.cardekho.com/)

After downloading the dataset we have to create a new environment in anaconda prompt using command conda create –n carprediction python=3.7 (carprediction is new environment name),so that our requirement.txt file will contain all the needful libraries to be install for particular project.

### **Feature Engineering**:

There are 301 records and 9 features, from the dataset I saw that Fuel\_Type, Seller\_Type, Transmission, Owner, this are categorical features .so I found out the unique categories from all of them using ‘print(df[‘Seller\_Type’].unique())’ .

Next I checked is there any missing value present using ‘df.isnull().sum()’ no missing values are present. After this step to see some of the details of the data like mean,std,count of numerical features I used ‘df.describe()’.

There is a feature called as ‘Year’ so it may be 2014,2016 etc.if the year is less at that time depreciation of car will also be going off.by using the feature Year I’m going to create a new feature called “no of year =Year-current year(2020)” .ex-year=2014, means car is 6 years old. The “car\_name” feature is not important as it will not be helpful for predicting car price. so I have created the final dataset using “final\_dataset=df[['Year','Selling\_Price','Present\_Price','Kms\_Driven','Fuel\_Type','Seller\_Type','Transmission','Owner']]”

I’ve created my new feature as “final\_dataset['Current Year']=2020”

“final\_dataset['no\_year']=final\_dataset['Current Year']- final\_dataset['Year']” and drop year, current year column as it not required “final\_dataset.drop(['Year'],axis=1,inplace=True)” “final\_dataset=final\_dataset.drop(['Current Year'],axis=1)”.as there are less no of categories in categorical features I have performed one hot encoding using “final\_dataset=pd.get\_dummies(final\_dataset,drop\_first=True)” So cleaning of the data has done, now I’ve my final dataset.

### **Feature Selection**:

I have performed correlation to see how much every feature is related to each other using “final\_dataset.corr()”also performed “sns.pairplot()” to see the plotting of the same diagrammatically, but this graph does not say much information hence I’m going to plot heatmap using “g=sns.heatmap(final\_dataset[top\_corr\_features].corr(),annot=True,cmap="RdYlGn")from this graph I can analyse that which features are negatively correlated and highly correlated so selling price and present price are highly positively correlated( green colour),Fuel\_type petrol and Fuel\_type diesel are negatively correlated(Red Colour) all this information can be checked out.as here we have less no of feature we are not putting condition according to correlation value for feature selection.(ex-if corr.value >0.8 drop that feature as both are having same job we can drop one)

Now I have taken my independent and dependent feature X,Y. To see which all features are having more important I used ExtraTreeRegressor “model = ExtraTreesRegressor()”. Also plot graph of feature importances for better visualization using “feat\_importances = pd.Series(model.feature\_importances\_, index=X.columns)” “feat\_importances.nlargest(5).plot(kind='barh')”

### **Model Selection & Training**:

Now split X and Y into Train and Test Set for model training, from all regressor algorithm I have selected RandomForestRegressor as it giving better accuracy.

Next I have performed RandomizedSearchCV technique of hyperparameter tunning as this technique is faster than gridsearchCV to estimate best parameters which will give best accuracy.

### **Prediction & Confusion matrix creation**:

now next job is to predict the test set result using “predictions=rf\_random.predict(X\_test)”. But to see how much effectively our model is working, I have compared y-test with predicted result using “sns.distplot(y\_test-predictions)” this gives normally distributed plot means our model is performing well with less error. also used scatter plot to see the same “plt.scatter(y\_test,predictions)”

Now lastly calculated MSE,MAE, RMSE.

## ***Fake News Classifier***:

### **Importing the data**:

* **Problem Statement:** To build a system that will identify unreliable news articles.
* The dataset name “train.csv” is downloaded from this site [https://www.kaggle.com/c/fake-news/data#](https://www.kaggle.com/c/fake-news/data)

Data Description:

* id: unique id for a news article
* title: the title of a news article
* author: author of the news article
* text: the text of the article; could be incomplete
* label: a label that marks the article as potentially unreliable
  + 1: unreliable
  + 0: reliable

### **Feature Engineering**:

Dataset contain 20800 records & 5 Features from this features ‘label’ is our dependent feature and remaining all are independent features. I’m going to consider “text” feature for classification as it’s having more no of words.

first I’m getting my dependent and Independent features using “y=df[‘label’]” “X=df.drop('label',axis=1)” as I’ve text data I’ve to do some text preprocessing.so I’m going to use some of the NLP Algorithms like CountVectorizer(Bag Of Words), TfidfVectorizer, HashingVectorizer for feature Extraction. First I’m going to use Bag Of words For Feature Extraction using “from sklearn.feature\_extraction.text import CountVectorizer, TfidfVectorizer, HashingVectorizer”.

Now our first step in feature engineering is to drop missing values using “df=df.dropna()”

After dropping nan values some of the records are gets deleted hence I have r

resetted index by copying the dataset into Messages variable “messages=df.copy()” “messages.reset\_index(inplace=True)”

* **Data Cleaning :** In data Cleaning step I’m going to replace Punctuation and anything which are not Letter by spaces. I used Natural Language Toolkit to compare our text data with Stopwords (is,the,a I,can etc) and if they are not in stopwords list will perform stemming(convert Past Future tense words into Present tense to Decrease Sparse matrix dimensions.) using following syntax

“review = [ps.stem(word) for word in review if not word in stopwords.words('english')]” “review = ' '.join(review)”

“corpus.append(review)” next task is to take most frequent features from Corpus list which will helpful for predicting news is unreliable or reliable by create bag of words model using “cv = CountVectorizer(max\_features=5000,ngram\_range=(1,3))” here I took 5000 most frequent words will take combination of 1or 2or 3 words. Now I’m ready with data and will perform train test split.To check the combination of features which my model has taken I’ve perform “cv.get\_feature\_names()[:20]” now will see the parameters selected by CV Model using “cv.get\_params()” next I’ve created a dataframe of selected features using

“count\_df =pd.DataFrame(X\_train,columns=cv.get\_feature\_names())”

Now I’m ready with final data.

### **Model Training & Selection& Prediction**:

### I have used Multinomial Naïve Bayes and Passive Aggressive Classifier Algorithm

### Algorithm using

### “from sklearn.naive\_bayes import MultinomialNB “classifier=MultinomialNB()”

### “from sklearn.linear\_model import PassiveAggressiveClassifier”

### “linear\_clf =PassiveAggressiveClassifier(n\_iter=50)” and also plot confusion matrix with classes Fake and Real data. Multinomial Naïve Bayes gives accuracy 90%.and Passive Aggressive Classifier as it works better on text data gives accuracy of 91% . also performed Hyperparameter Tunning to select best Alpha Value which will give best Result.for alpha =0.2 gives Score=90.25%

To see most fakest word and Real Word I performed following steps

“feature\_names =cv.get\_feature\_names()” “sorted(zip(classifier.coef\_[0], feature\_names), reverse=True)[:20]” This will give Most Real data. “sorted(zip(classifier.coef\_[0], feature\_names))[:5000]”. And this will give most fake data.

Feature Extraction Using TFIDF : Here I performed feature extraction using TFIDF by training data on Multinomial Naïve Bayes and Passive Aggressive Classifier Algorithm as did earlier with Bag of words Feature Extraction Technique, I got the accuracy of 88% with Multinomial Naïve Bayes and 91% using Passive Aggressive Classifier also tried HashingVectorizer Feature Extraction Technique With Multinomial naïve Bayes Algorithm Gives accuracy of 87%.