# Machine Learning Engineer Nanodegree

## Capstone Project Report

**A Customer Predicting whether Purchasing a car is worth enough or not using supervised learning.**

**Sushma Ponakala**

**June 29th 2018**

## Definition

### Project Overview

In today's world we are spending so much of our expenses on purchasing motor bikes and cars.Based on their brands some of them are so expensive.so,when we are affording too much on the vehicles,it is necessary to check and examine all the features of a vehicle.we should check our safety as well.Now-a-days people are very passionate about cars.I took a dataset that represents the features of cars.so we should check if  a car with specific features and specifications is worthy enough  to buy and as well as safe or not and it really matters.Supervised learning deals with two important concepts called Regression and Classification.These concepts helps for this prediction.

### Problem Statement

we make predictions if it is worth enough spending such huge amount on car ,so that customer saves his money from purchasing a less worthy car for huge amount.suppose we purchase a car for 20 lakhs and if all its features are not so good and if there is no guarantee for safety,then purchasing that car is useless.So,before spending such huge amount on it ,if we predict whether purchasing it gives comfort and safety  then we can save money.So Im going to predict whether purchasing a particular branded car is worth or not looking at its features.

This prediction can be done using supervised learning.  Classification and Regression are two powerful techniques that can be used . Algorithms like RandomTreeClassifier, KNN, Logistic Regression, SVM can be used.F score is calculated for each of these algorithms and the algorithm with highest F score is used.we split the data into testing and training sets and evaluate F score on them for all the above Algorithms.

### Features and Description

- overall price
- buying price
- price of the maintenance
- comfort
- number of doors
- capacity in terms of persons to carry
- the size of luggage boot
- estimated safety of the car and
- Target variable which we should predict.

**Dataset and Inputs**

The dataset has 1729 instances and 6 attributes.They are:

Sales: Level of Sales

Maintenance: Level of maintenance

Doors: Number of doors for the car

Persons: Capacity of the car (Number of persons in a car)

lug_boot: Size of the luggage boot

safety: Safety level of customers

By considering all the above features we predict whether a car is worth or not.

Target: target variable that tells whether a car is worth enough or not.

Overall 6 attributes are considered and dataset description can be viewed here: https://archive.ics.uci.edu/ml/machine-learning-databases/car/car.names

There are no missing values in the dataset.

Class Distribution (number of instances per class)

| class | N | N[%] |
|---|---|---|
| unacc | 1210 | (70.023 %) |
| acc | 384 | (22.222 %) |
| good | 69 | ( 3.993 %) |
| v-good | 65 | ( 3.762 %) |

## Metrics

For this Project I choose fbeta_score .

Precision (also called positive predictive value) is the fraction of relevant instances among the retrieved instances, while recall(also known as sensitivity) is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

Precision=TP/(TP+FP)

Recall=TP/(TP+FN)

Where TP=True Positives and FN=False Negatives.

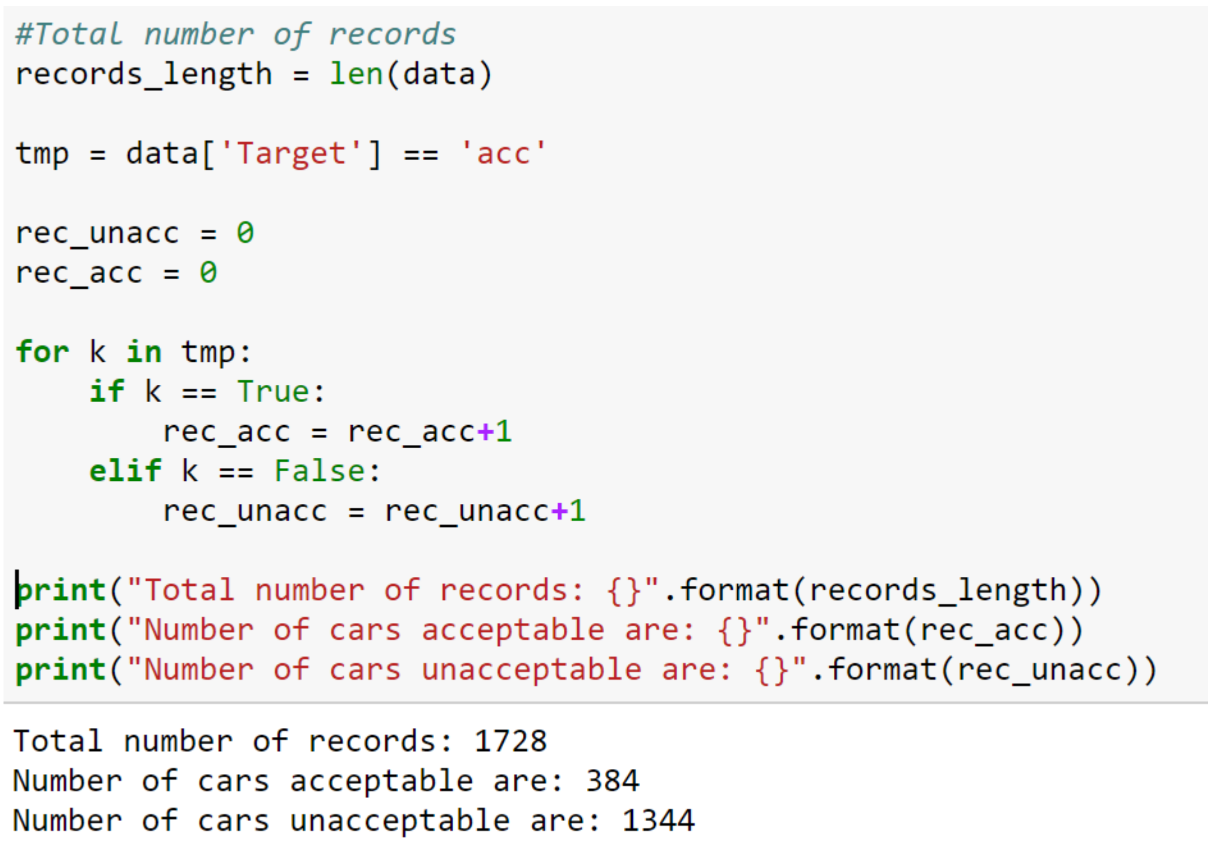
$$F\beta=(1+\beta2)\cdot precision\cdot recall/(\beta2\cdot precision)+recall$$

Accuracy can also be used but when data is unbalanced using Fbeta is much much better.

Analysis

Data Exploration

In this section I calculated the total records and the number of accepted and Unaccepted cars.



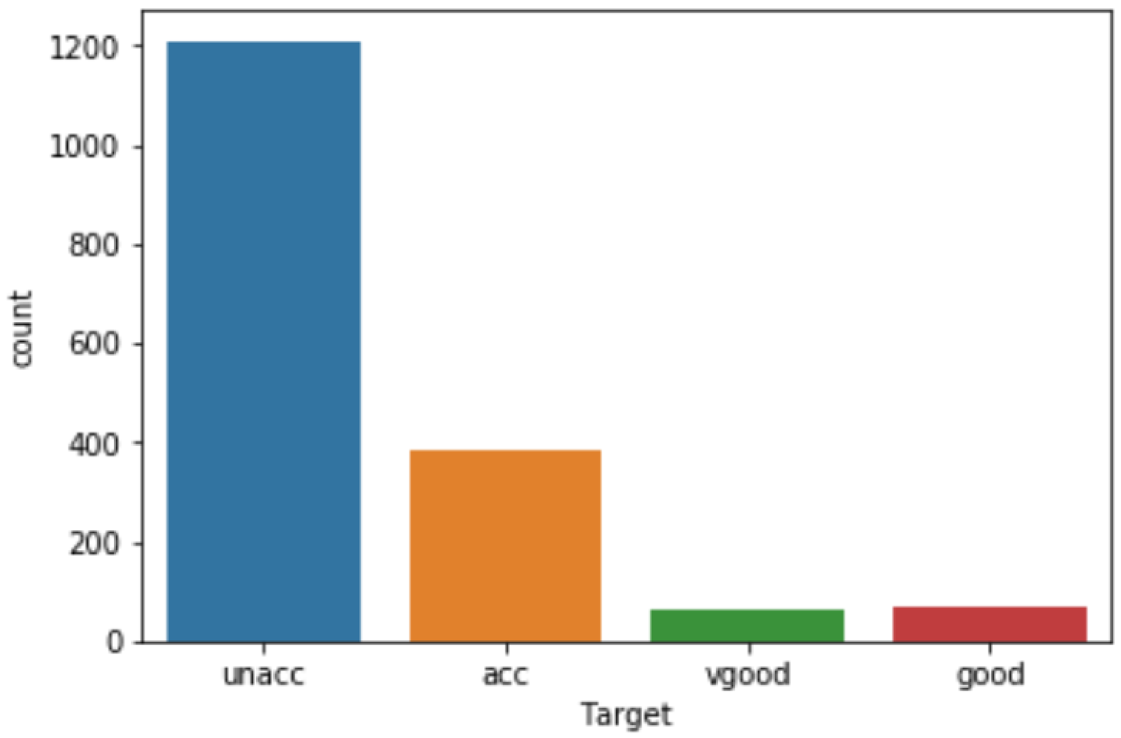
```python
#Total number of records
records_length = len(data)

tmp = data['Target'] == 'acc'

rec_unacc = 0
rec_acc = 0

for k in tmp:
    if k == True:
        rec_acc = rec_acc+1
    elif k == False:
        rec_unacc = rec_unacc+1

print("Total number of records: {}".format(records_length))
print("Number of cars acceptable are: {}".format(rec_acc))
print("Number of cars unacceptable are: {}".format(rec_unacc))
```

```
Total number of records: 1728
Number of cars acceptable are: 384
Number of cars unacceptable are: 1344
```

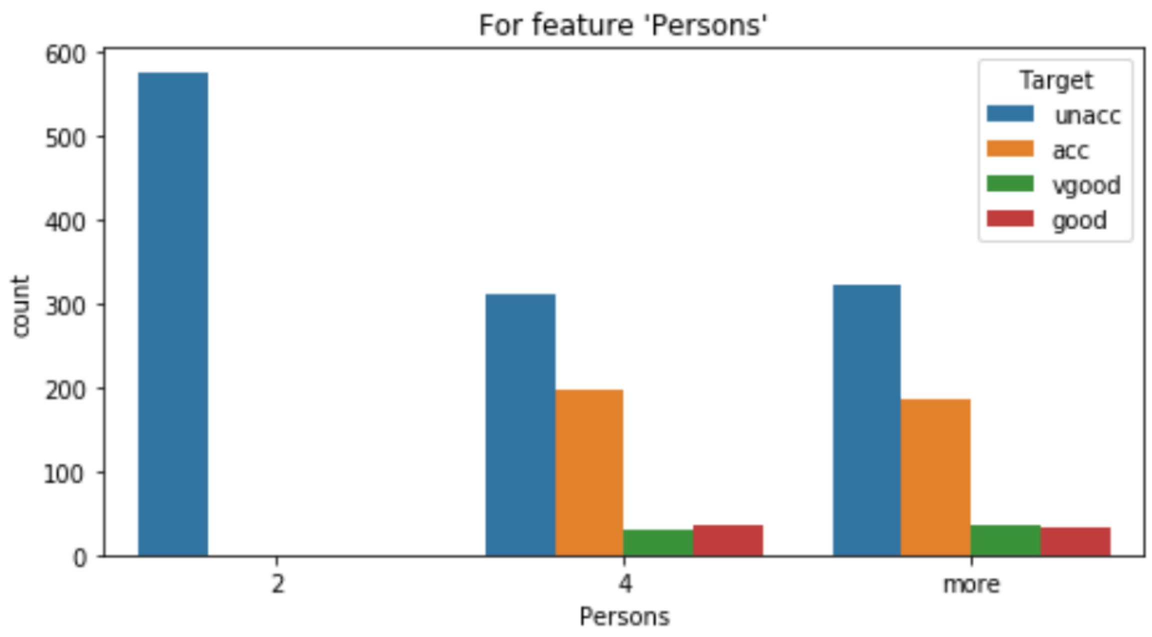
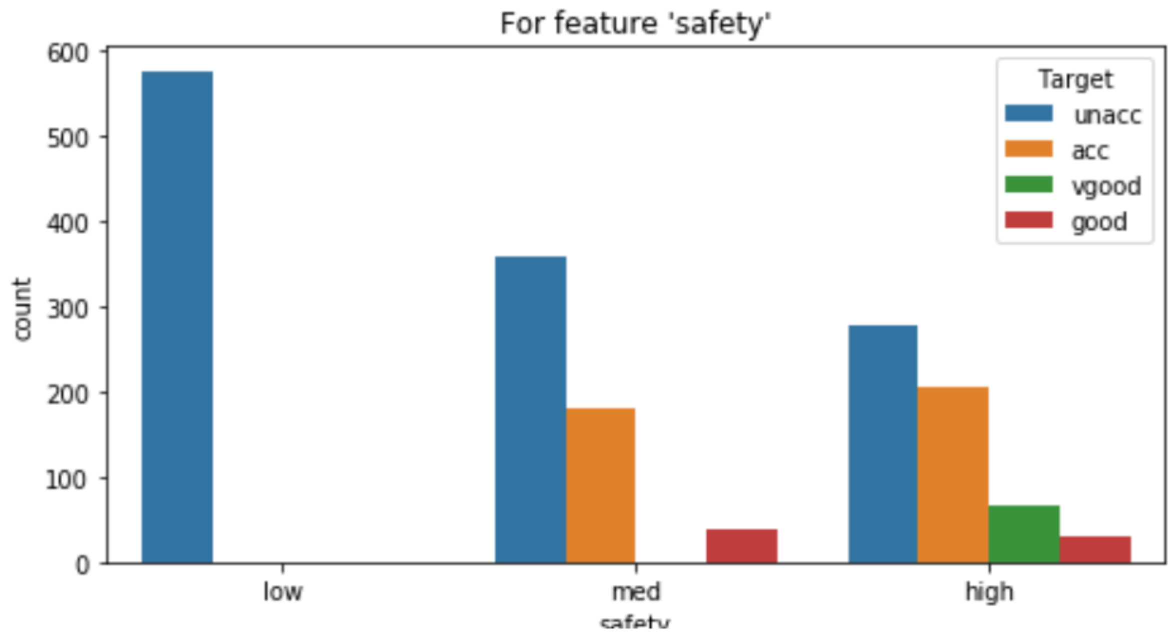
| | Sales | Maintenance | Doors | Persons | lug_boot | safety | Target |
|---|---|---|---|---|---|---|---|
| **0** | vhigh | vhigh | 2 | 2 | small | low | unacc |
| **1** | vhigh | vhigh | 2 | 2 | small | med | unacc |
| **2** | vhigh | vhigh | 2 | 2 | small | high | unacc |
| **3** | vhigh | vhigh | 2 | 2 | med | low | unacc |
| **4** | vhigh | vhigh | 2 | 2 | med | med | unacc |

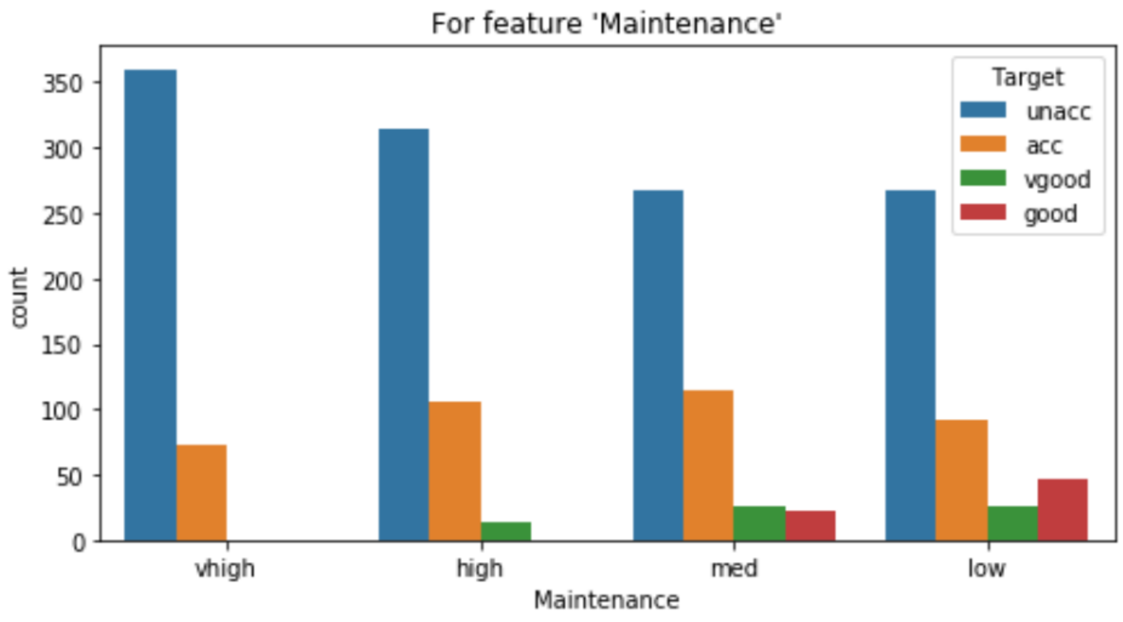
## Data visualisation

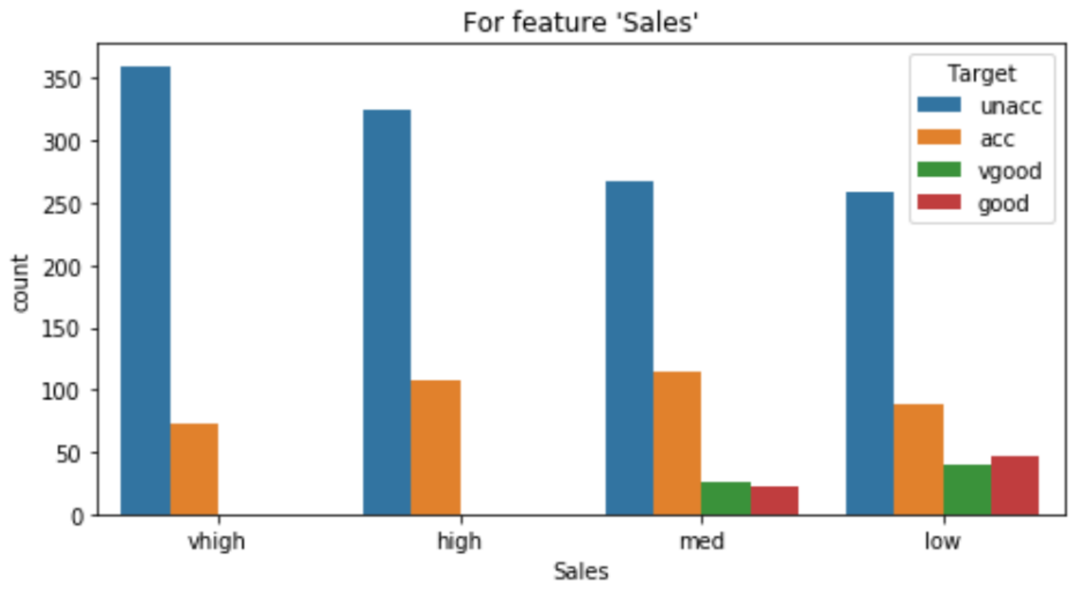
**Observe blue and orange bars and see the accepted and unaccepted ratio.**



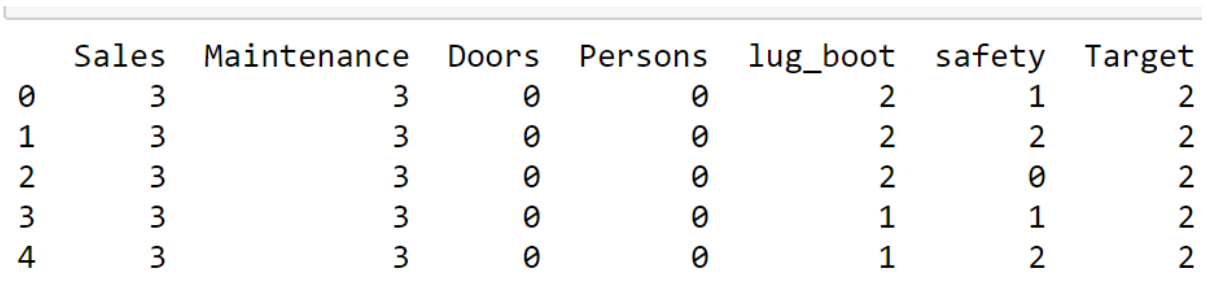
**The below plots will show you each feature's corresponding 'unacc' and 'acc'. Do observe only "blue" and "orange" bars for all the below charts.**


For feature 'Persons'
count
Persons
2
4
more
Target
unacc
acc
vgood
good

For feature 'safety'
count
safety
low
med
high
Target
unacc
acc
vgood
good



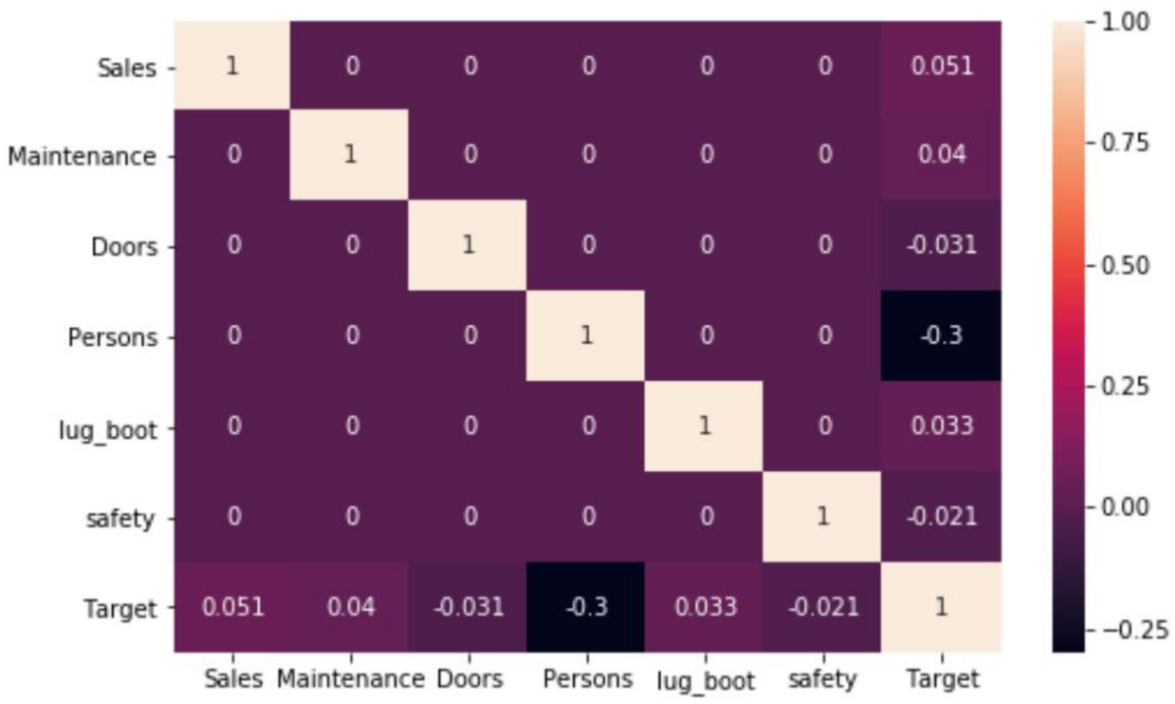


**These are acceptance rates for some features.acc .Observe blue and orange bars .blue stands for unaccepted and green stands for accepted.**



```
   Sales  Maintenance  Doors  Persons  lug_boot  safety  Target
0      3            3      0        0         2       1       2
1      3            3      0        0         2       2       2
2      3            3      0        0         2       0       2
3      3            3      0        0         1       1       2
4      3            3      0        0         1       2       2
```

**Categorial data is converted into Numerical data in the above image.**



**The above plot shows the correlation between all the features which is 0.Also observe the correlation between attributes and the class label .**

## Algorithms and Techniques

### Logistic Regression

Logistic Regression is a very simple classification Algorithm.It doesn't require any linear relationships between the target variable and the predictors.But the problem with this is it is more prone to overfitting.

### RandomForestClassifier

Random Forests and other ensemble methods are excellent models for some classification tasks. They don't require as much preprocessing as some other methods and can take both categorical and numerical variables as input. A simple decision tree isn't very robust, so ensemble methods run many decision trees and aggregate their outputs for prediction. This process controls for overfitting and can often produce a very robust, high-performing model. Random forests and some other boosting methods are easy, out-of-the-box models that I often use early in the modeling process to see how they perform with my data. Also, they have a super helpful feature called feature importances.

### KNN

KNN can be used for both classification and regression predictive problems. However, it is more widely used in classification problems in the industry. To evaluate any technique we generally look at 3 important aspects:

1. Ease to interpret output

2. Calculation time

3. Predictive Power

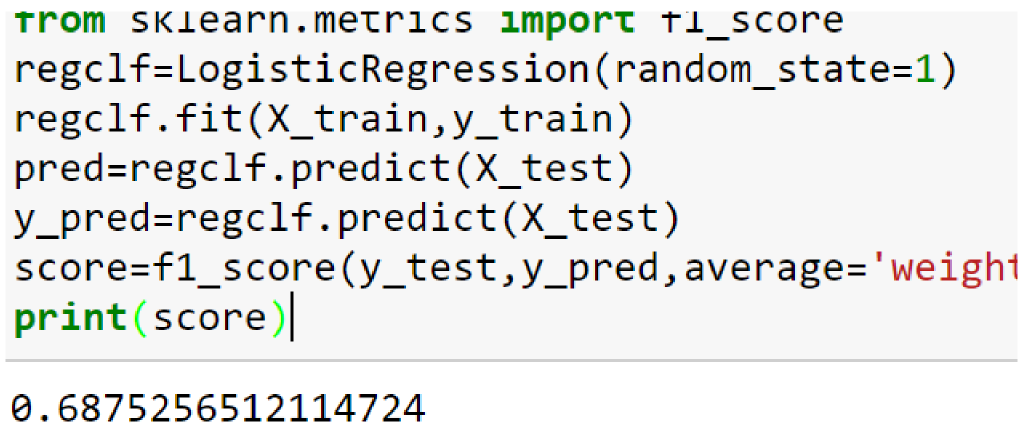
KNN is very simple .The purpose of KNN is to classify based on attributes and training samples.It works based on the minimum distance from the query instance to the training samples to determine the k-nearest neighbours.After we gather k nearest neighbours ,we take simple majority of these k nearest neighbours to be the prediction of the query instance.The classification is using the majority of votes among the classification of the k objects.

## Techniques

GridsearchCV is the technique I used to tune the hyper parameters.A great combination of features can be selected using GridSearchCV.

## BenchMark Model

I used Logistic Regression as my Benchmark model and compared its results with KNN and RandomForestClassifier .



```
from sklearn.metrics import f1_score
regclf=LogisticRegression(random_state=1)
regclf.fit(X_train,y_train)
pred=regclf.predict(X_test)
y_pred=regclf.predict(X_test)
score=f1_score(y_test,y_pred,average='weight
print(score)
```

```
0.6875256512114724
```

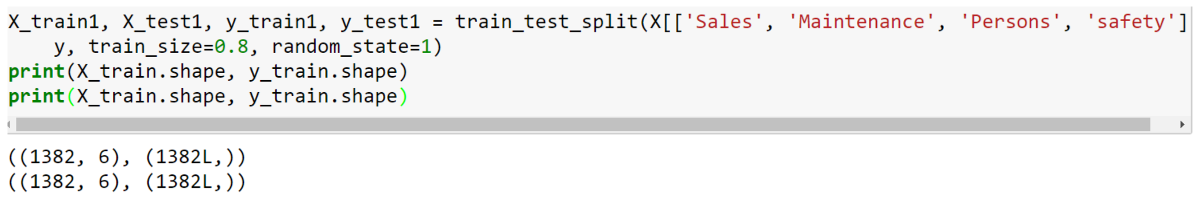
I got a f score of 0.68.

## Methodology

### Data Preprocessing

There are no null values in my dataset.There are two kinds of data i.e Categorial and Numerical data.I converted the categorial data into Numerical dat by Hot encoding.

Then I splitted my data into training and testing sets and calculated F score for all the Algorithms.

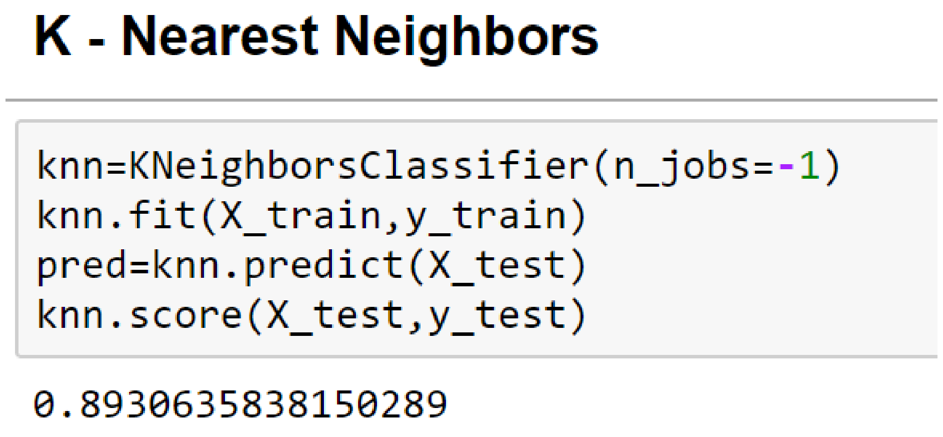


```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X[['Sales', 'Maintenance', 'Persons', 'safety']
    y, train_size=0.8, random_state=1)
print(X_train.shape, y_train.shape)
print(X_train.shape, y_train.shape)
```

```
((1382, 6), (1382L,))
((1382, 6), (1382L,))
```

## Testing other Algorithms

### KNN

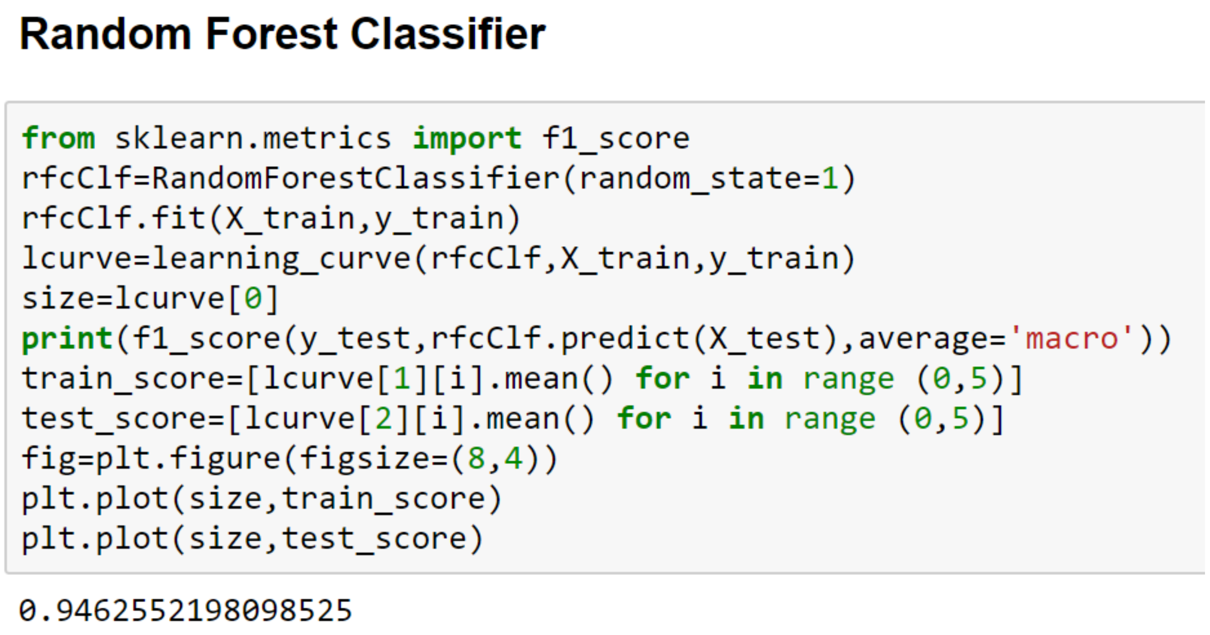


## K - Nearest Neighbors

```python
knn=KNeighborsClassifier(n_jobs=-1)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
knn.score(X_test,y_test)
```

```
0.8930635838150289
```

**RandomForest Classifier**



## Random Forest Classifier

```python
from sklearn.metrics import f1_score
rfcClf=RandomForestClassifier(random_state=1)
rfcClf.fit(X_train,y_train)
lcurve=learning_curve(rfcClf,X_train,y_train)
size=lcurve[0]
print(f1_score(y_test,rfcClf.predict(X_test),average='macro'))
train_score=[lcurve[1][i].mean() for i in range (0,5)]
test_score=[lcurve[2][i].mean() for i in range (0,5)]
fig=plt.figure(figsize=(8,4))
plt.plot(size,train_score)
plt.plot(size,test_score)
```

```
0.9462552198098525
```

**Classifier :KNN**

**Fscore:0.893**

**Classifier:RandomForestClassifier**

**Fscore:0.9462**

**Fscore increased for RandomForestClassifier .so Im going to pick it and apply GridSearchCv on it to tune the parameters.**

**Refinement**

```
from sklearn.model_selection import GridSearchCV
param_grid={'max_depth':[5,10,20],
           'max_features':[4,6,'auto'],
           'max_leaf_nodes':[2,3,None],}
grid=GridSearchCV(estimator=RandomForestClassifier(n_estimators=50,n_jobs=-1,random_state=30),
                  param_grid=param_grid,cv=10,n_jobs=-1)
grid.fit(X_train,y_train)
print(grid.best_params_)
print(grid)
```

```
from sklearn.metrics import f1_score
rfcClf=RandomForestClassifier(max_depth=10,random_state=10,max_features=6,max_leaf_nodes=None)
rfcClf.fit(X_train,y_train)
print(f1_score(y_test,rfcClf.predict(X_test),average='macro'))
```

```
0.9562371285672256
```

**I optimised using Gridsearch and therefore fscore has increased from 0.94 to 0.95.**

## Results

### Justification

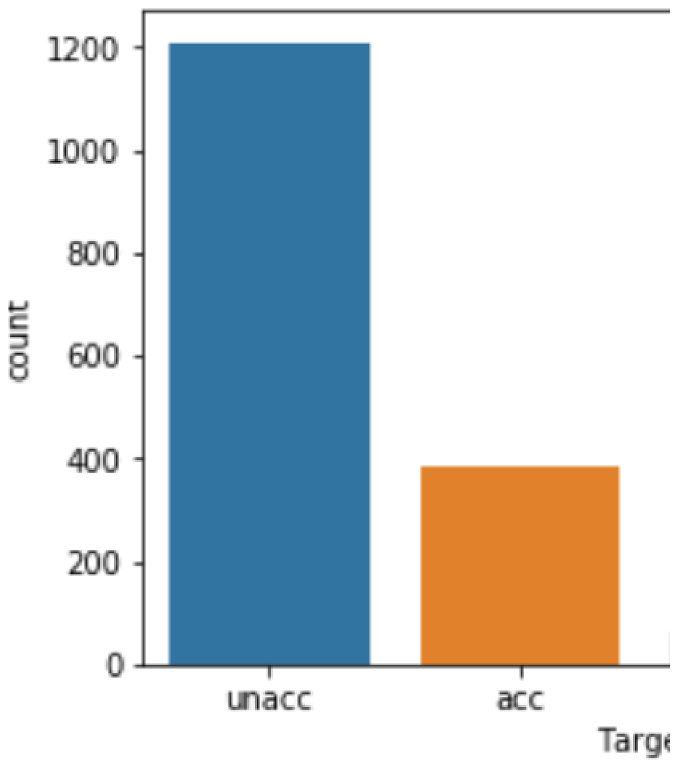
**I used Logistic Regression as the Benchmark model and got an fscore of 0.64 . With KNN fscore has increased to 0.89 and with RandomForestClassifier fscore increased to 0.94.Also hyperparameters are best selected and reduced.**

### Benchmark model :0.687

### Unoptimised model :0.893

### Optimised model :0.9462

### 5.Free Form Visualisation



The data is imbalanced hence we'll get most of the results as 'unacc'.I believe if we have balanced data ,results will be more appropriate.

## Reflection

**First I read the data using pandas Dataframe.**

**Then I found the no of instances and attributes.**

**Then I converted the categorial data into Numerical data.**

**Then I used heatmap and checked how correlated are my attributes.**

**Then I used Logistic Regression as my benchmark Algorithm.**

**Then I compared my benchmark model with other models like KNN and RandomForestClassifier.**

**Then I foung the best model(RFC) and tuned it using GridSearchCV.**

**I didn't find anything difficult in doing all the above things.I did them Interestingly.**

## Improvements

I think if I used balanced data ,I could have got much better results.

Conclusion

For the problem I picked,I applied logistic regression and then applied KNN and RFC and compared with the benchmark model.Finally  I picked RandomForestClassifier because among all the models it produced best fscore of 0.94.

## REFERENCES

http://people.revoledu.com/kardi/tutorial/KNN/

http://scikit-learn.org/stable/auto_examples/ensemble/plot_forest_importances.html

https://machinelearningmastery.com/logistic-regression-for-machine-learning/

https://beckernick.github.io/logistic-regression-from-scratch/

https://www.geeksforgeeks.org/understanding-logistic-regression/