Machine Learning Engineer Nanodegree

CAPSTONE PROJECT REPORT

CAR EVALUATION

Geetha Harika Darimireddy

June 27th,2018

I. Definition

Project Overview
A car with specific features and specifications is worth or not is very important to a customer. It is not a surprise that the car with particular features is very profitable for a customer. The car is not worth means that may not be safe or may be the maintenance is high and comfort is low. However, this issue become ultra-costly when the sales become less because of its unacceptability. Supervised learning deals with Regression and Classification. Classification is being implemented in different fields Ex: Medical diagnosis and this model has gained a moment to understand the change in behaviour of the consumer, so this helps business works on the data.

Every customer invests a lot to buy a new car, its fair for him to know whether the car is worth or not for him to buy by analysing its features. If a customer buys any car, mainly that car should have more safety level and high comfort level. I am searching for a dataset which contains features of a car which decide whether a car is worth or not.

Link: https://archive.ics.uci.edu/ml/datasets/Automobile

Link: https://archive.ics.uci.edu/ml/datasets/Wine+Quality

Link: https://www.kaggle.com/dmi3kno/newcarsalesnorway

Problem Statement

Buying a car by investing lot of money on it and there is a chance that the customer may feel that the car he bought is not worth for its price because of its low safety level or low comfort level. Due to low safety level there may be a chance of accidents. If a customer wants to buy any particular car it would be

better if he get to know whether the car is worth for him to buy or not by knowing the features of that car.

Features and Description

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

Target: This is our target variable

Buying a car is not easy for everyone since more money is needed. Once if we buy a car and later if we feel its unsafe and comfort level is less it is not easy to replace it or sell it, since the customer gets loss. So, it would be better for the customer to know its features first and know whether it is worthy or not and then buy it. So, I am using classification model of supervised learning by passing the current employee data to the model and predicting whether the car is worthy or not. There are no null values. And I will do hot encoding for categorical column and there are exactly 10 features in the dataset. I used Label encoder and converted the categorical data into numerical data. I will perform ensemble learning model to understand why the car is unworthy and will finalize only one model base on the fscore. If the model is performing better, we will perform tuning and the main theme of the model is to find whether the car is worth for buying or not. Total number of records are 1728 and 7 columns. And the number of cars that are acceptable(worthy) are 518 and number of cars that are unacceptable(unworthy) are 1210. Here Target is our target variable which tells whether it is worth or not. I took this data set from https://archive.ics.uci.edu and made little changes to the data set.

Link: https://archive.ics.uci.edu/ml/datasets/Car+Evaluation

Metrics

Accuracy in classification problems is the number of correct predictions made by the model over all kinds predictions made.

$$Accuracy = (TP+TN)/(TP+TN+FP+FN)$$

In the Numerator, are our correct predictions (True positives and True Negatives) (Marked as red in the fig above) and in the denominator, are the kind of all predictions made by the algorithm (Right as well as wrong ones).

Precision is a measure that tells us what proportion of patients that we diagnosed as having cancer, actually had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people having a cancer are TP.

$$Precision = TP/(TP+FP)$$

Recall is a measure that tells us what proportion of patients that actually had cancer was diagnosed by the algorithm as having cancer. The actual positives (People having cancer are TP and FN) and the people diagnosed by the model having a cancer are TP. (Note: FN is included because the Person actually had a cancer even though the model predicted otherwise).

$$Recall = TP/(TP+FN)$$

Fβ score – measures the effectiveness of retrieval with respect to a user who attaches beta times as much importance to recall as precision.

$$F\beta=(1+\beta2) \cdot \text{precision . recall}/(\beta2 \cdot \text{precision})+\text{recall}$$

When beta=0.5 more emphasis is placed on precision. This is called f 0.5 score.

I choose fbeta score as a metric because my data is imbalanced. Accuracy will not give better results when the data is imbalanced. The histogram for the target variable left is shown below. Accuracy is not a good metric when the data is imbalanced.
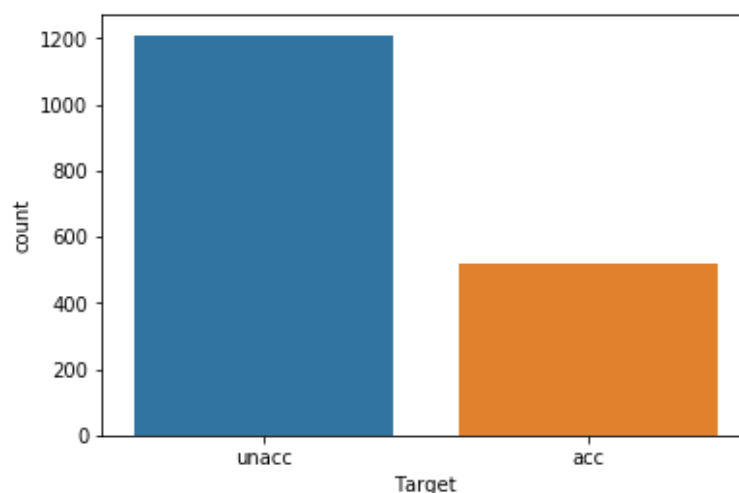
II. Analysis

Data Exploration

In this data exploration section, I calculated the total number of records and the number of cars which are acceptable and number of cars which are unacceptable and their percentages.

```
Total number of records: 1728
Number of cars acceptable are: 518
Number of cars unacceptable are: 1210
Percentage of cars that acceptable are: 29.9768518519%
Percentage of cars that unacceptable are: 70.0231481481%
```

data.head()

	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.describe()

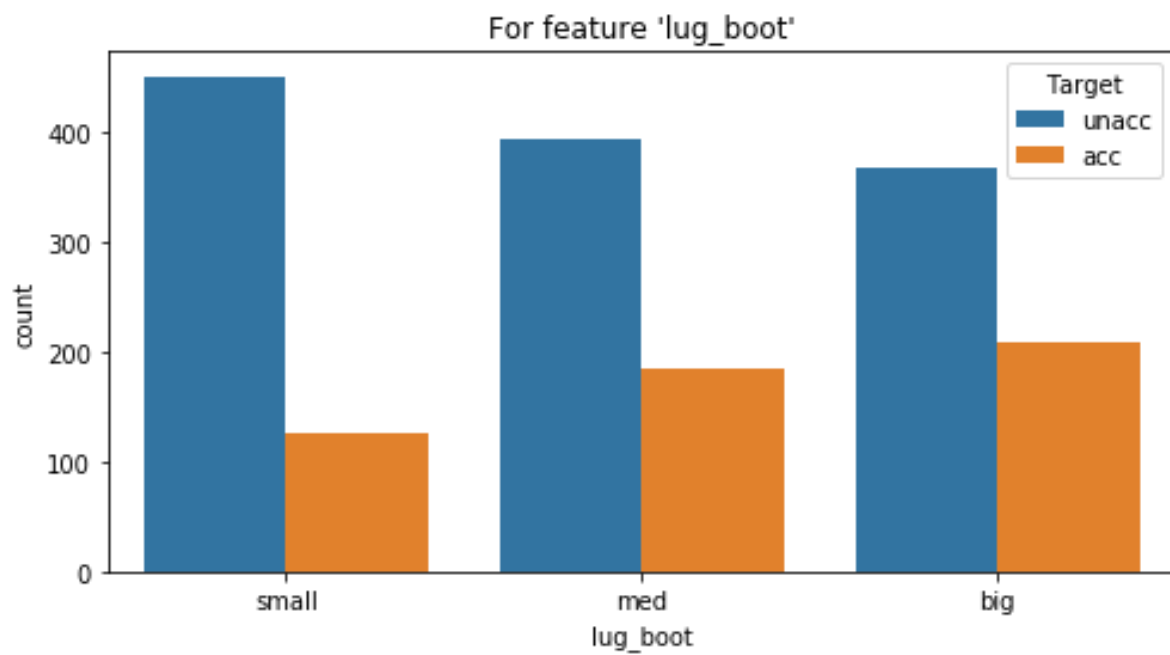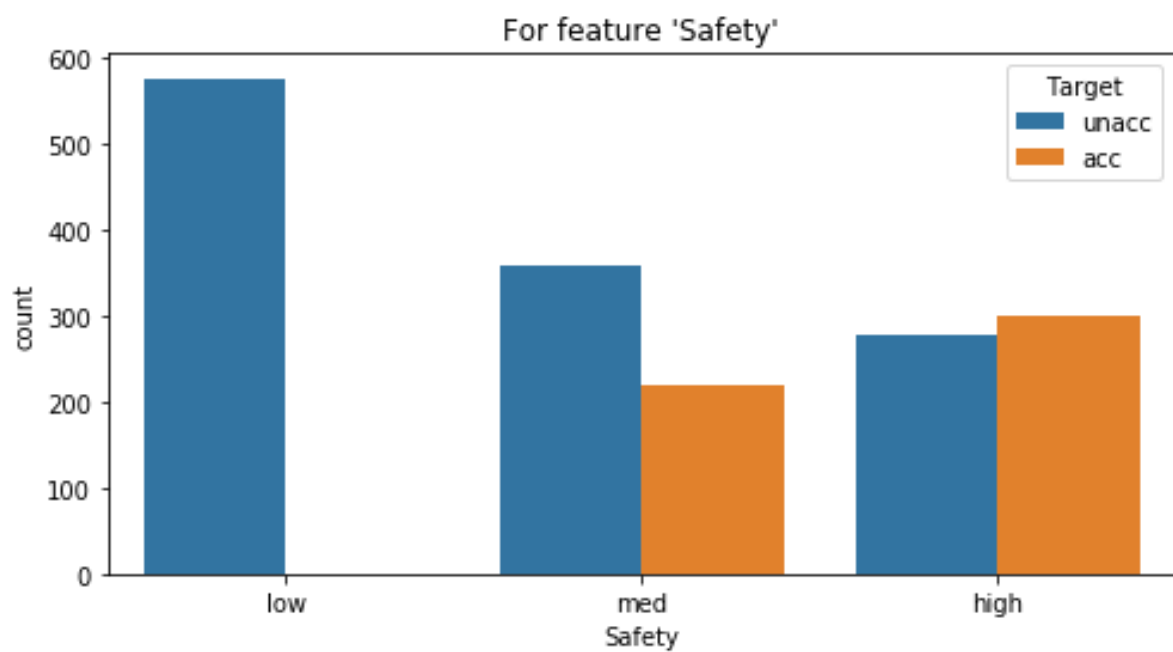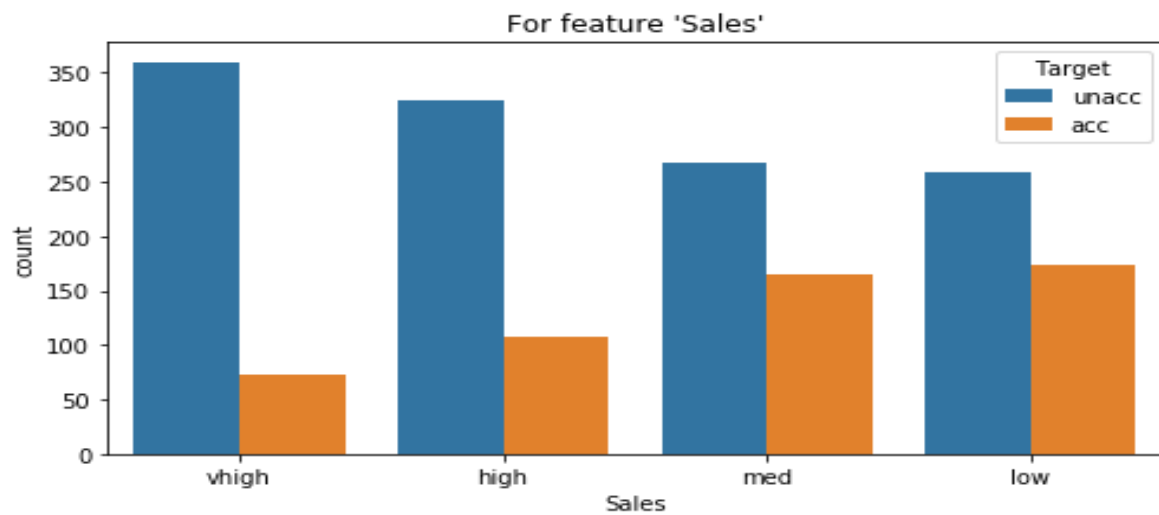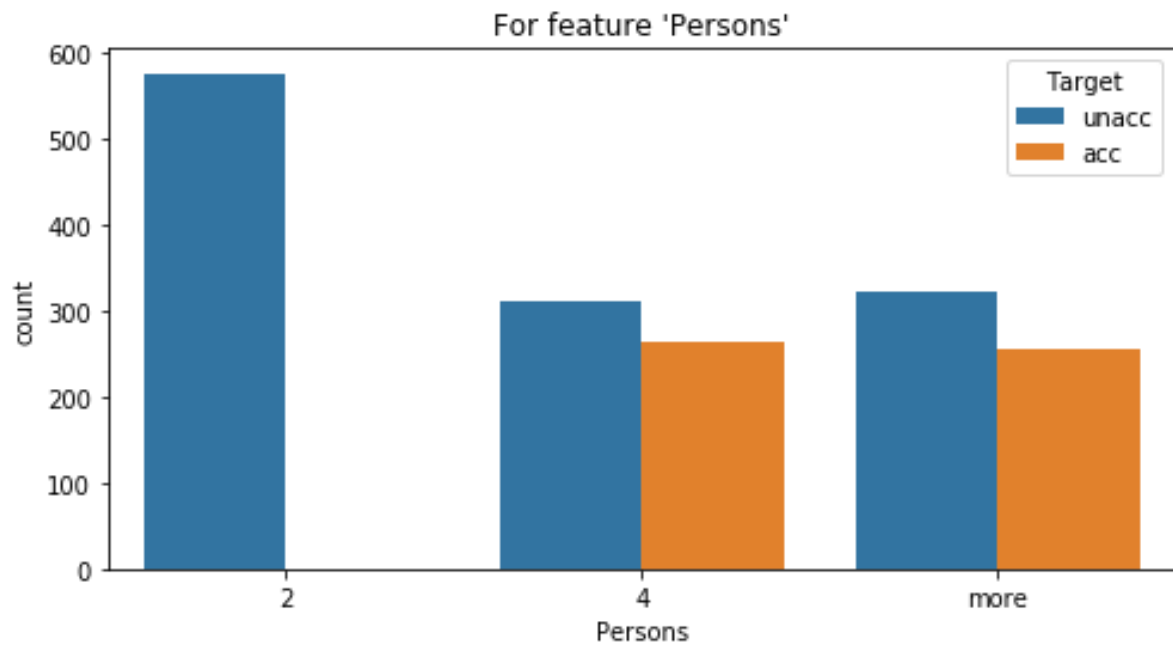|	Sales	Maintenance	Doors	Persons	lug_boot	Safety	Target
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	2
top	med	med	3	more	med	med	unacc
freq	432	432	432	576	576	576	1210

Data Visualization

The below graph shows how my class label is distributed in the data: The blue plot is unaccepted one and the orange is accepted one.
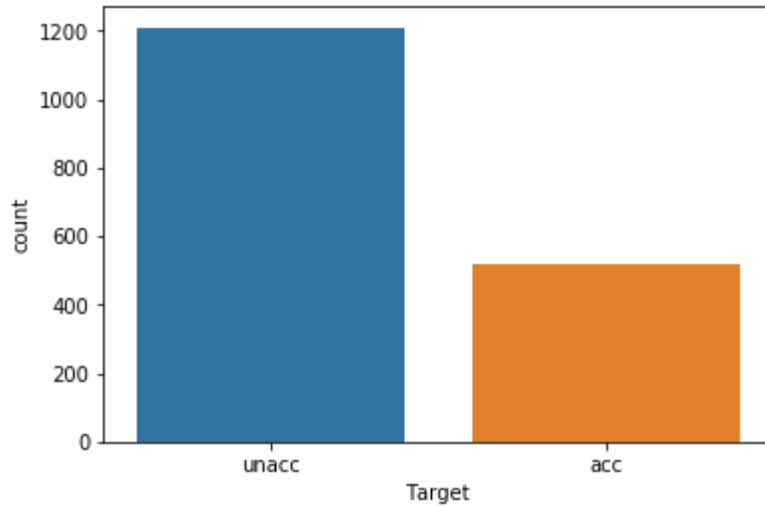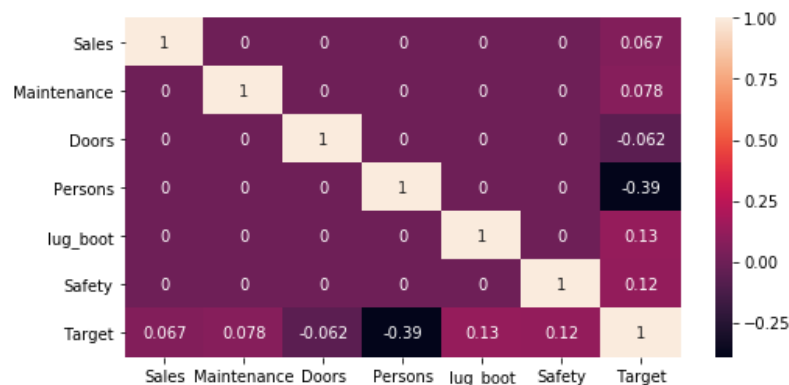
The below plots representing each plot for every attribute individually and plotting how many records are accepted and unaccepted for each different value in those attributes:

For feature 'lug_boot'

For feature 'Maintenance'

For feature 'Persons'

For feature 'Sales'

For feature 'Safety'
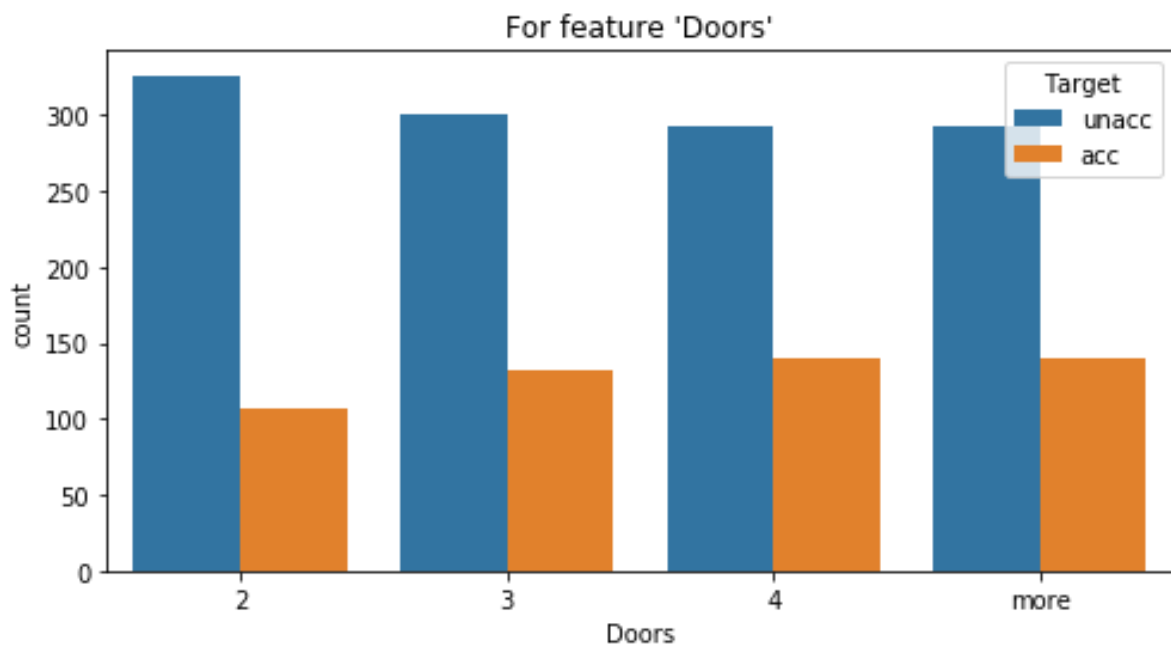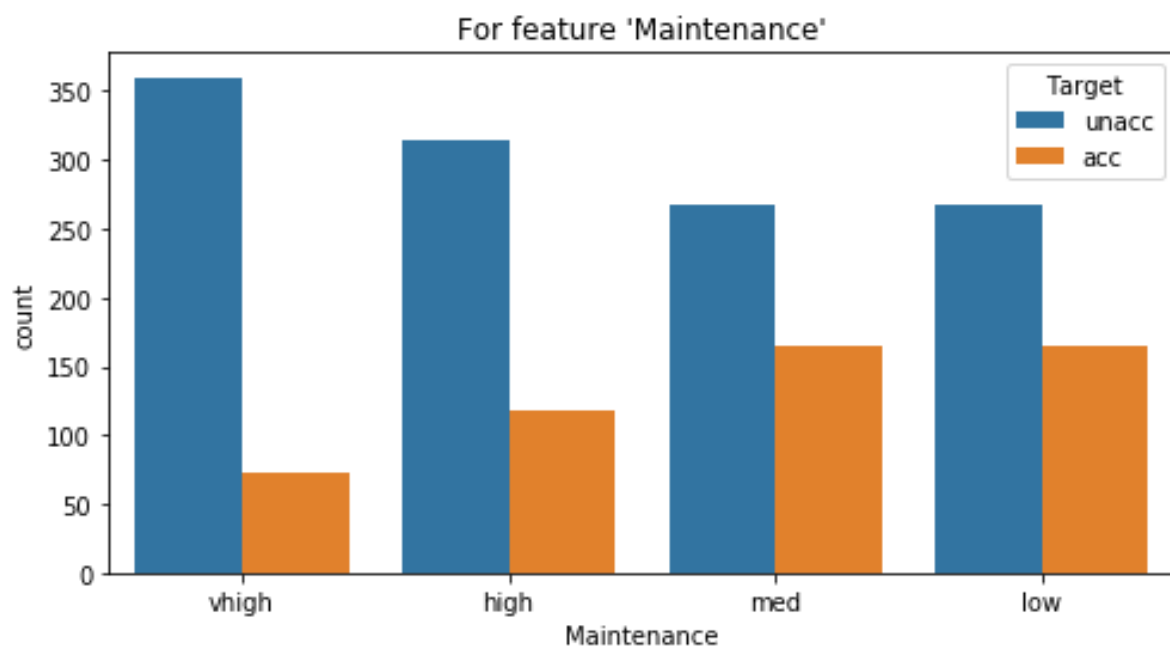
The below heat map describes how my attributes are correlated with each other:

By observing the heat map, we can say that there is no correlation between all attributes. But every attribute is correlated with class label.

Algorithms and Techniques

Algorithms

The problem I am solving is multiclass classification. I am going to select three classification algorithms out of which I am going to select one that performs well.

The three algorithms I choose are:

1.Logistic Regression

I choose this algorithm because it is a simple classification algorithm. It is more robust. The main disadvantage of this algorithm is that it is more prone to overfitting. However, adding more and more variables to the model can result in overfitting, which reduces the generalizability of the model beyond the data on which the model is fit.

2.K-Nearest Neighbours

I choose this algorithm because of its easy of interpretation and low calculation time. This algorithm is faster compared to the above algorithm. k-NN is a type of instance-based learning, or lazy learning, where the function is only approximated locally, and all computation is deferred until classification. The

KNNalgorithm is among the simplest of all machine learning algorithms

3.RandomForestClassifier

I choose this algorithm because it gives more accurate values and It can overcome overfitting in many cases. It is also simple to implement as it has less hyperparameters. Random decision forests are an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

4.AdaBoost Classifier

I choose this algorithm because it is adaptive in the sense that subsequent weak learners are tweaked in favour of those instances misclassified by previous classifiers. AdaBoost is sensitive to noisy data and outliers. In some problems it can be less susceptible to the overfitting problem than other learning algorithms. The individual learners can be weak, but if the performance of each one is slightly better than random guessing, the final model can be proven to converge to a strong learner.

Techniques

The technique I am going to use to improve performance is GridSearchCV because tuning hyperparameters is a very important thing in improving the performance of the model. Finding best parameters is a very difficult task and gridsearchcv is a technique which takes parameters to be tuned and returns best parameter combination among all the combinations very easily.

Benchmark model

Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbeta score will be greater than Logistic regression model. Accuracy, f-beta score and confusion matrix and will try to get better results in the ensemble learning models.

I got 0.72 f-score value using bench mark model.

III. Methodology

Data Pre-processing

The pre-processing steps I took are:

1. Convert the categorical data into numerical data using LabelEncoder.

```python
from sklearn.preprocessing import LabelEncoder
encoder = LabelEncoder()
for i in data.columns:
    data[i]=encoder.fit_transform(data[i])
print(data.head())
```

	Sales	Maintenance	Doors	Persons	lug_boot	Safety	Target
0	3	3	0	0	2	1	1
1	3	3	0	0	2	2	1
2	3	3	0	0	2	0	1
3	3	3	0	0	1	1	1
4	3	3	0	0	1	2	1

2. Split the data into training and testing data using train_test_split function in sklearn.

Data Split

```python
from sklearn.model_selection import train_test_split
X=data[data.columns[:-1]]
y=data['Target']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

For the models that I mentioned below I calculated the F-scores for that models to check their performance by fitting the data into the classifiers and calculated the F1-score of the that models by using F1-score function from sklearn.metrics.

1.Logistic Regression

Logistic regression is another technique borrowed by machine learning from the field of statistics. It is the go-to method for binary classification problems (problems with two class values).

I will import logistic regression from sklearn.linear_model. And will implement the classifier. In this no parameter is passed into the classifier.

And my parameters are random_state.

```
regclf=LogisticRegression(random_state=1)
regclf.fit(X_train,y_train)
pred=regclf.predict(X_test)
y_pred=regclf.predict(X_test)
f1score=f1_score(y_test,y_pred,average='weighted')
print(f1score)
```

0.7202992921123794

```
lcurve=learning_curve(regclf,X_train,y_train,cv=15)
size=lcurve[0]
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)
```

[<matplotlib.lines.Line2D at 0x216af438>]
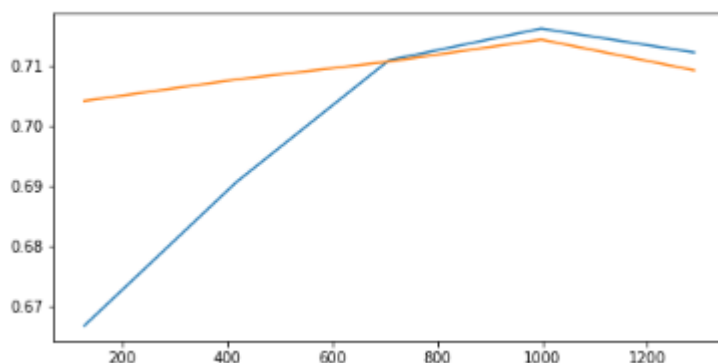
2.K-Nearest Neighbours

```
knn=KNeighborsClassifier(n_jobs=-1)
knn.fit(X_train,y_train)
pred=knn.predict(X_test)
knn.score(X_test,y_test)
```
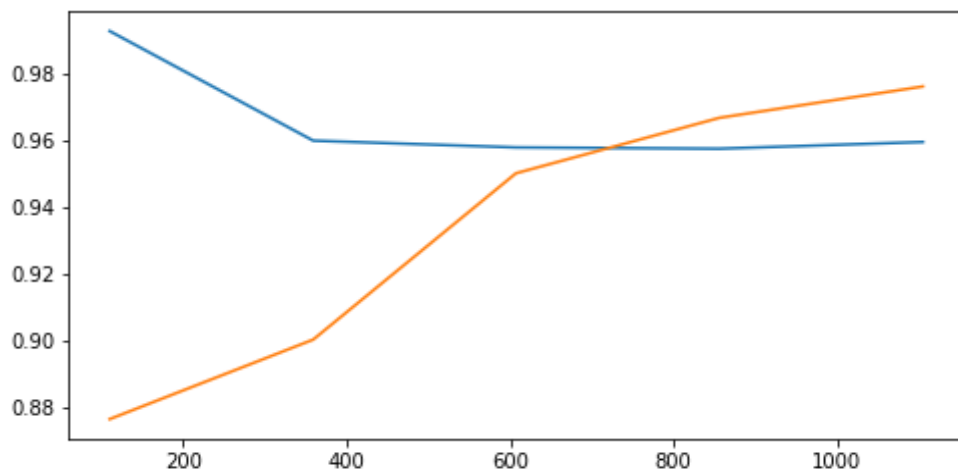
0.9393063583815029

3.AdaBoost Classifier

```
AdaClf=AdaBoostClassifier(random_state=1)
AdaClf.fit(X_train,y_train)
lcurve=learning_curve(AdaClf,X_train,y_train)
size=lc[0]
print(f1_score(y_test,AdaClf.predict(X_test),average='macro'))
train_score=[lcurve[1][i].mean() for i in range (0,5)]
test_score=[lc[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)
```

0.9430640118479514

[<matplotlib.lines.Line2D at 0x20b76710>]

4.Random Forest Classifier

```
rfcClf=RandomForestClassifier(random_state=1)
rfcClf.fit(X_train,y_train)
lcurve=learning_curve(rfcClf,X_train,y_train)
size=lc[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=[lc[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)
```

0.9546189830804879

[<matplotlib.lines.Line2D at 0x1fa61748>]
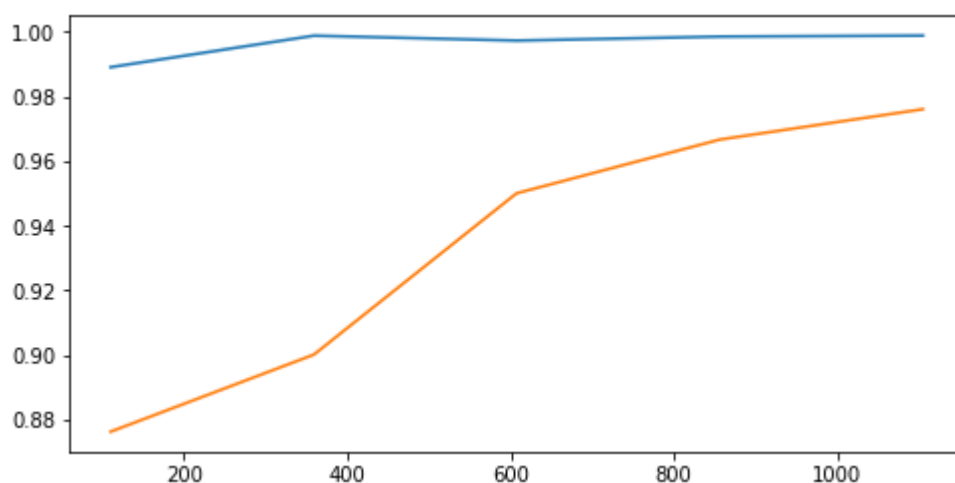
I choose RandomForestClassifier as my solution model as it has high F-score compared to remaining three models. The initial model has a f1 score value of 72% whereas KNNclassifier has a f1 score of 93 % which is quite well compared

to Logistic Regression, and for AdaBoost classifier I got 94%. Finally, I got an f1 score of around 95.46% by using Random Forest Classifier. After taking Random Forest Classifier as my solution model I tuned the parameters to increase the performance of the model using GridSearchCV.

Refinement

I used Grid search technique to improve my solution model. And the f1 score is increased 2%(i.e.,95.4% to 97.7%)

```python
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=29),
                  param_grid=param_grid,cv=10,n_jobs=-1)
grid.fit(X_train,y_train)
print(grid.best_params_)
```

```
{'max_features': 6, 'max_leaf_nodes': None, 'max_depth': 10}
```

```python
rfcClf=RandomForestClassifier(max_depth=20,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'))
```

```
0.9773094915402438
```

IV.RESULTS

Model Evaluation and Validation

As we know that the fbeta score on unoptimized testing data is 0.95 and for optimised model. fbeta score is 0.97.

Random Forest Classifier is chosen to do parameter tuning by GridSearchCV.

RandomForestClassifier(max_depth=20,random_state=10,max_features=6, max_leaf_nodes=None)

GridSearchCV(cv=10, error_score='raise', estimator=RandomForestClassifier(
            bootstrap=True, class_weight=None, criterion='gini',
            max_depth=None, max_features='auto',
            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, n_estimators=50,
 n_jobs=-1,oob_score=False, random_state=29,
verbose=0, warm_start=False),fit_params=None,
iid=True, n_jobs=-1,
param_grid={'max_features': [4, 6, 'auto'],
'max_leaf_nodes': [2, 3, None],
 'max_depth': [5, 10, 20]},
 pre_dispatch='2*n_jobs', refit=True,
return_train_score='warn',
scoring=None, verbose=0)
```

Justification

The Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbet a score will be greater than Logistic regression model. Accuracy, f-beta score an d confusion matrix and will try to get better results in the ensemble learning mod els. With choice of different classifier in hand such as Ensemble methods and I d ecided to go with ensemble learning because this method increase the f-score of the models. Performance of ensemble learning methods will not be slow, and it u ses weak learner to identify the features and label it with result this was best cho ice. I felt difficult in choosing which method should I choose to give best f-score value. And if I choose Gradient boosting I may fell into overfitting, so I decided to choose Random Forest classifier. And with the help of kfold cross validation and grid search I avoided overfitting.

Fbeta score for benchmark model is 0.72

By seeing the F-score values of all the four models we can say that our solution model works well than the benchmark model and the other model. It's Fscore value is high compared to remaining two models and it also takes less hyper parameters to solve the problem.

V. Conclusion

Free-Form Visualization
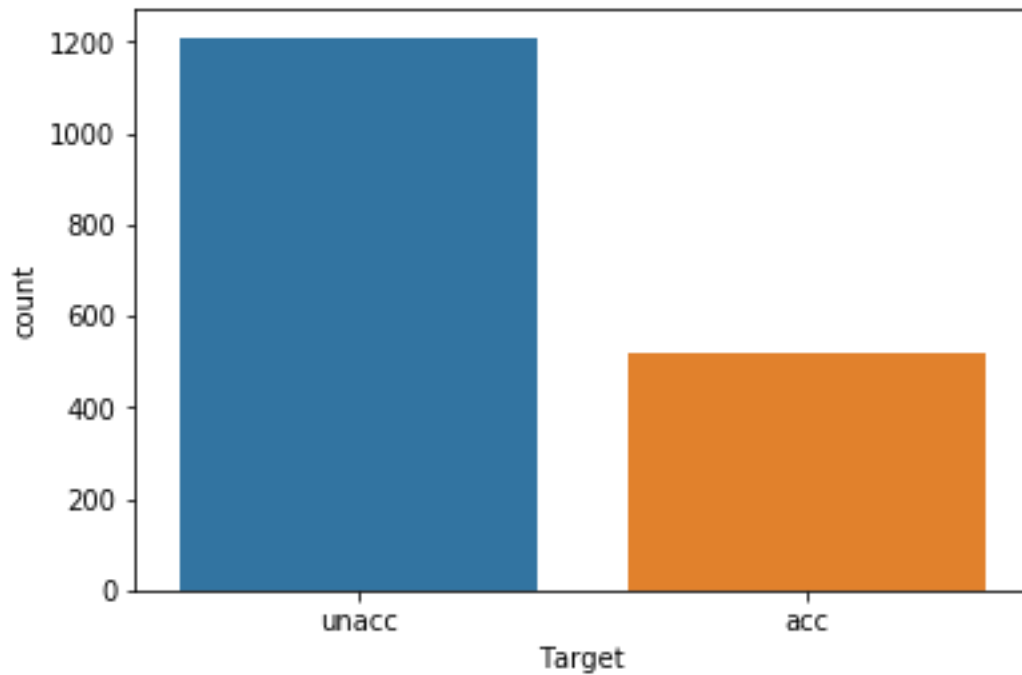
We can see in the above graph that class label is highly unbalanced hence we will get most of the results as 'unacc'. Hence If we have more data or data is balanced we can get better results compared to the current results.

```python
import numpy as np
rfc=RandomForestClassifier()
m=rfc.fit(X_train,y_train)
importances = m.feature_importances_
columns = X_train.columns.values
columns = [x for _,x in sorted(zip(importances,columns))]
indices = np.argsort(importances)
plt.title('Feature Importances')
plt.barh(range(len(indices)), importances[indices], color='y', align='center')
plt.yticks(range(len(indices)), columns)
plt.xlabel('Feature weights')
plt.show()
```
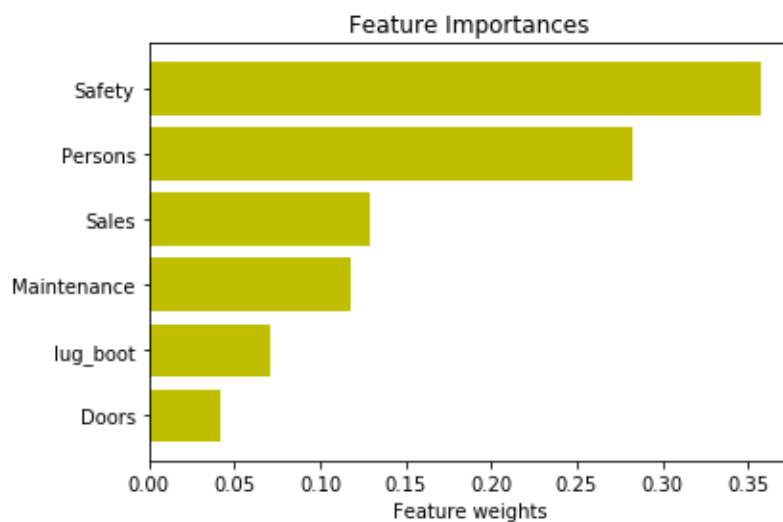
[0.14758967 0.11028315 0.04450488 0.28945125 0.06694605 0.34122499]

By observing above plots, we can say that our target value will not much depend on Doors attribute.

Reflection

1. First I selected my dataset from UCI machine learning repository and imported necessary libraries and read the dataset into a pandas dataframe.

2. Then I defined my problem statement i.e., domain knowledge and to predict car acceptability using the features.

3. After I described my data using describe () and have shown no of instances of each attribute in my data.

4. I converted my categorical data into numerical data is using label encoder.

5. Then I plotted heat map for my data set to know the correlation between each attribute.

6. Then I selected four different models for my problem and selected LogisticRegression as my benchmark model and I got very less performance.

7. Then I compared the performance of my bench mark model with other models. Random Forest classifier got high f-score when compared to other models.

8. Then I performed Parameter tuning using Grid Search CV for Random Forest Classifier and my f-score value has increased 2%.

9. I find it difficult when I am using GridsearchCV technique because I already got nearly nearly accurate result I thought there would be no improvement but then I found my f-score has increased 2%. Overall, I really loved doing the project.

Improvement

1. I think the results can be improved if I have more balanced data, then automatically the performance of models will increase after parameter tuning.

2. I think using SVM model or Naïve Bayes or other ensemble methods also we can achieve better results which is also same as RandomForests.

3. I don't think that I can get better results If I used my final solution as my new benchmark model because the data is highly unbalanced.

REFERENCES:

1. https://machinelearningmastery.com/boosting-and-adaboost-for-machine-learning/

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

3. https://towardsdatascience.com/cross-validation-in-machine-learning-72924a69872f

4. http://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_curve.html

5. http://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html

6. http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html