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

Here the problem is Car Acceptability by the customers i.e., based on the some features of a car we must tell that a new car with the same features is acceptable by the customers or not. This can be handled by using classification algorithms like logistic regression, KNN, Random Forests and AdaBoost classifier etc. We

must calculate accuracy and F-score for our training set and check it works fine for our testing data among different models.

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.

$$\text{Accuracy} = (TP+TN)/(TP+TN+FP+FN)$$

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, had cancer. The predicted positives (People predicted as cancerous are TP and FP) and the people having a cancer are TP.

$$\text{Precision} = TP/(TP+FP)$$

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).

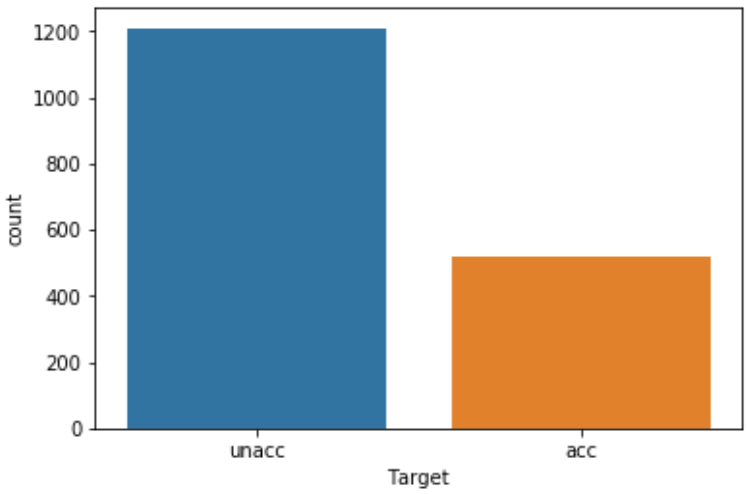
$$\text{Recall} = TP/(TP+FN)$$

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

$$F\beta=(1+\beta 2) \cdot \text{precision} \,.\, \text{recall}/(\beta 2 \cdot \text{precision})+\text{recall}$$

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()

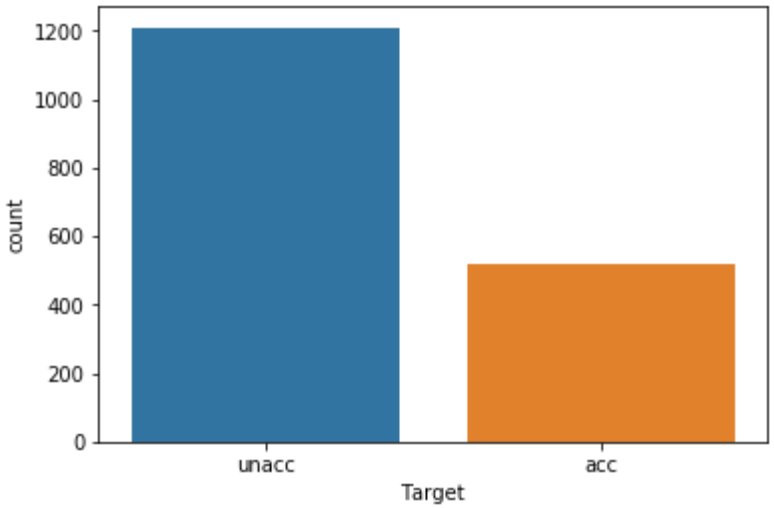
	Sales	Maintenance	Doors	Persons	lug_boot	Safety	Target
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

data.describe()

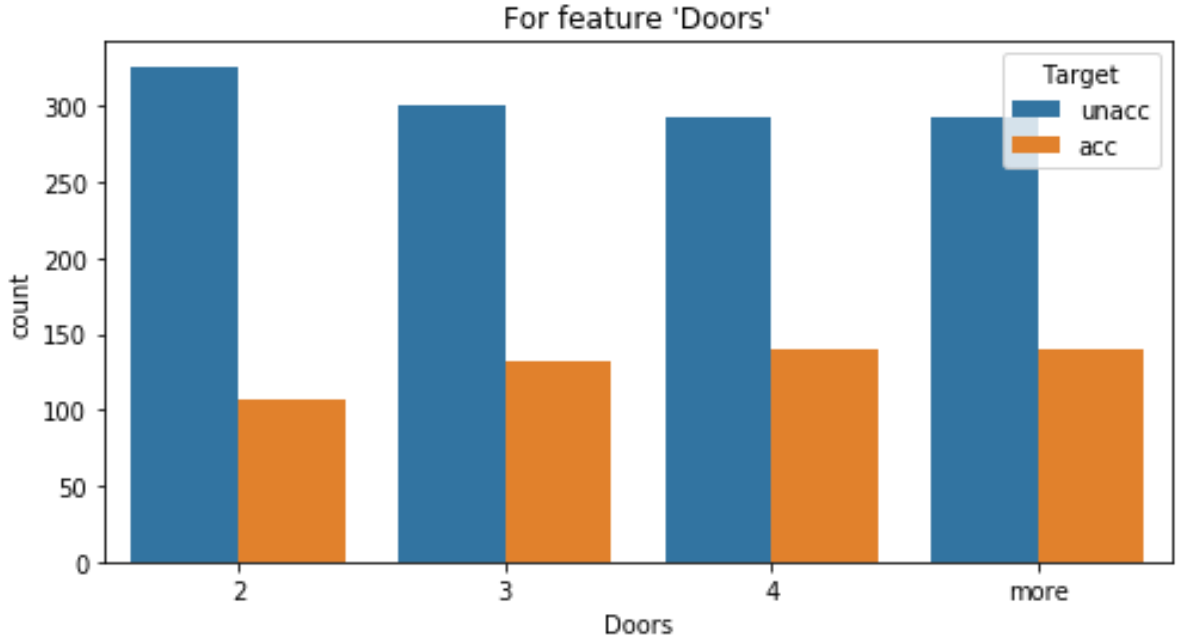
	Sales	Maintenance	Doors	Persons	lug_boot	Safety	Target
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	2
top	med	med	3	more	med	med	unacc
freq	432	432	432	576	576	576	1210

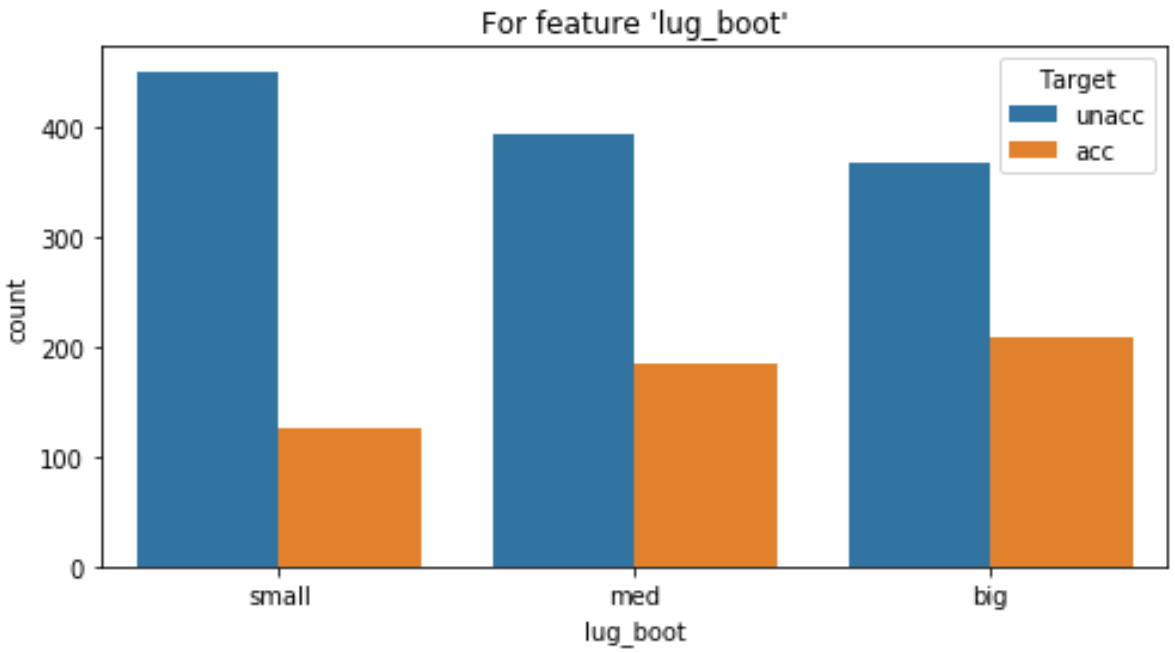
Data Visualization

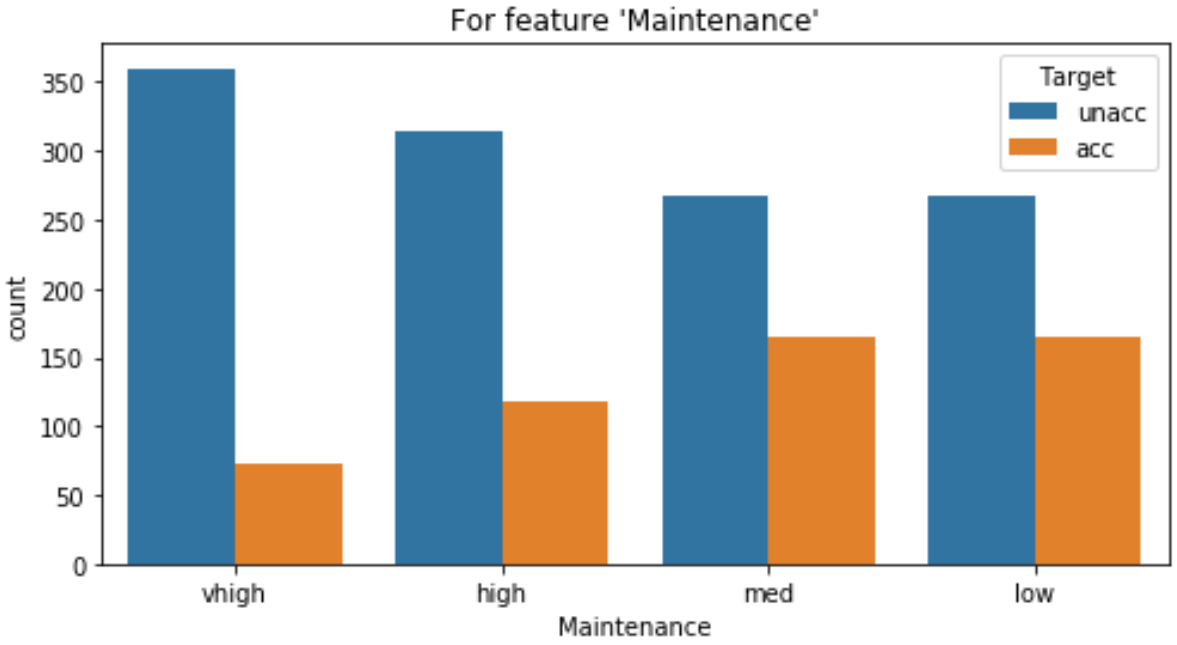
The below graph shows how my class label is distributed in the data: The blue plot is unaccepted one and the orange is accepted one.

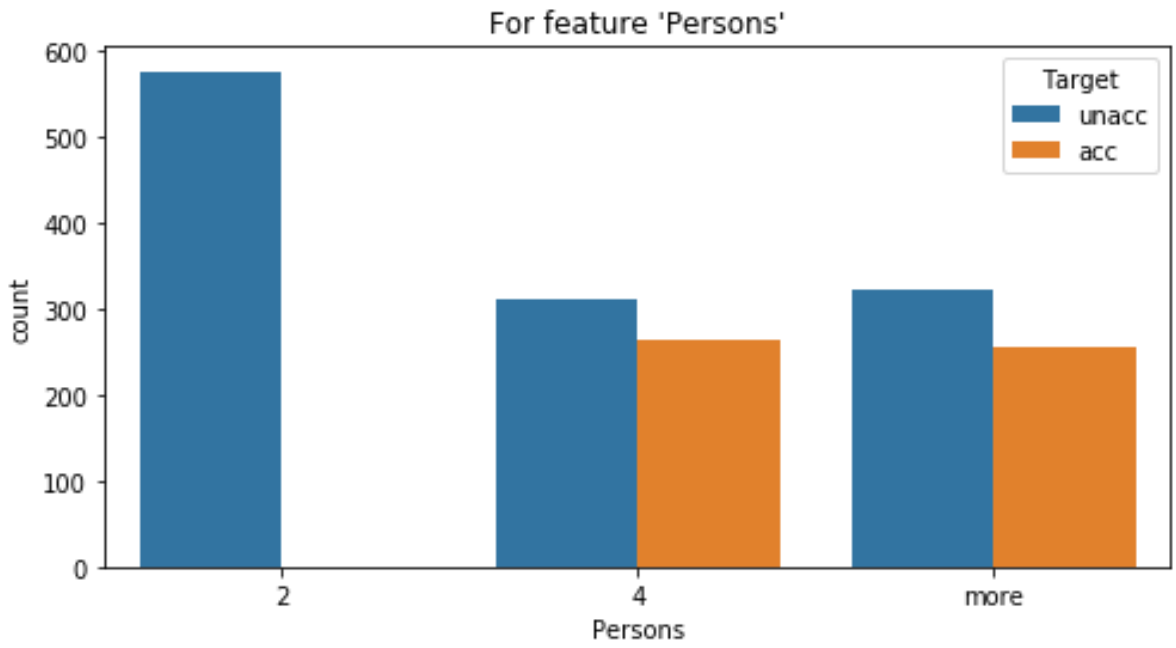


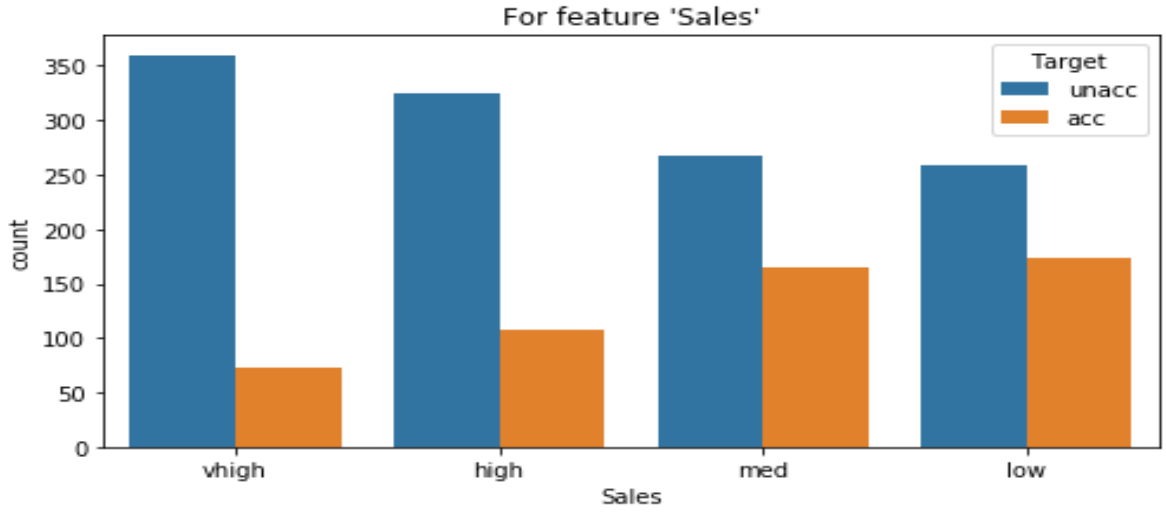
The below plots representing each plot for every attribute individually and plotting how many records are accepted and unaccepted for each different value in those attributes:

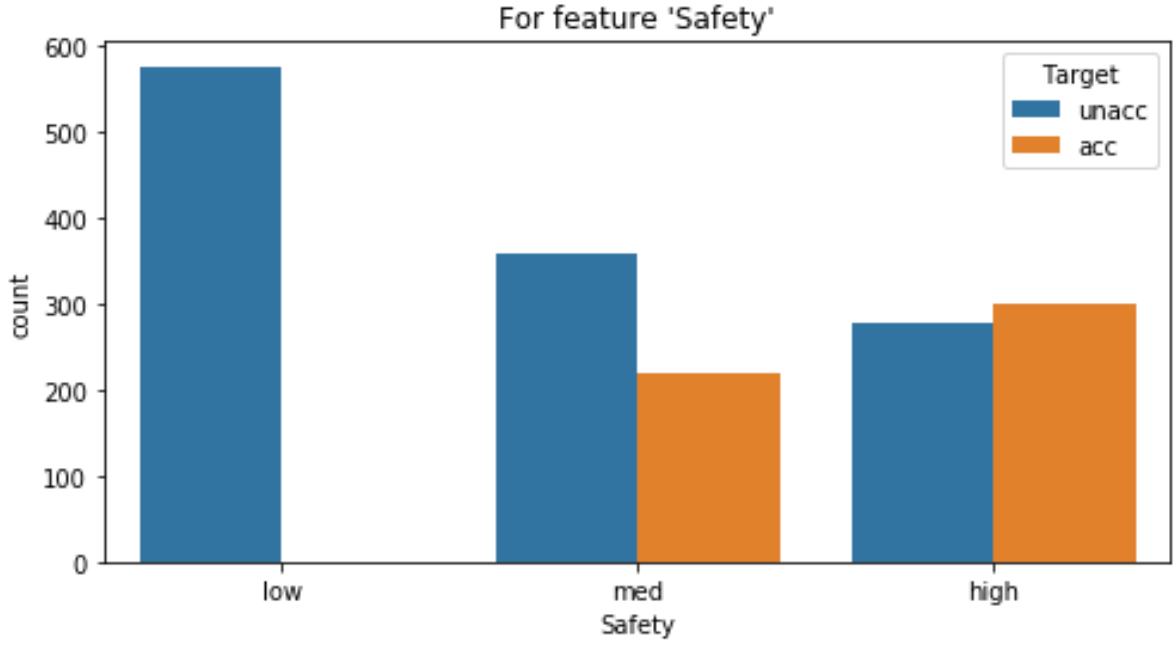



For feature 'lug_boot'
Target
unacc
acc
count
0
100
200
300
400
small
med
big
lug_boot

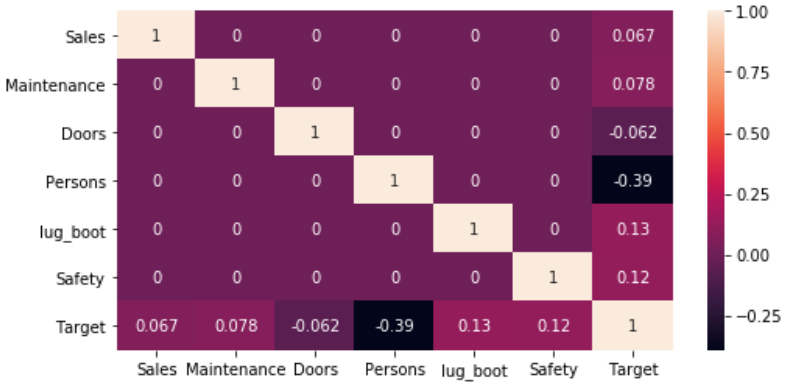

For feature 'Maintenance'
Target
unacc
acc
count
0
50
100
150
200
250
300
350
vhigh
high
med
low
Maintenance


For feature 'Persons'
Target
unacc
acc
count
Persons
2
4
more
0
100
200
300
400
500
600


For feature 'Sales'
Target
unacc
acc
count
Sales
vhigh
high
med
low
0
50
100
150
200
250
300
350


For feature 'Safety'
Target
unacc
acc
count
Safety
low
med
high
0
100
200
300
400
500
600

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.
Logistic regression measures the relationship between the categorical dependent variable and one or more independent variables by estimating probabilities using a logistic function. It uses a black box function to understand the relation between the categorical dependent variable and the independent variables. This black box function is popularly known as the SoftMax function.

The dependent variable is the target class variable we are going to predict. However, the independent variables are the features or attributes we are going to use to predict the target class.

Suppose the shop owner would like to predict the customer who entered the shop will buy the MacBook or Not. To predict whether the customer will buy the MacBook or not. The shop owner will observe the customer features like.

- Gender:
  - Probabilities wise male will high chances of purchasing a MacBook than females.
- Age:
  - Kids won't purchase MacBook.

The shop owner will use the above, similar kind of features to predict the likelihood occurrence of the event (Will buy the MacBook or not.)

In the mathematical side, the logistic regression model will pass the likelihood occurrences through the logistic function to predict the corresponding target class. This popular logistic function is the SoftMax function. We are going to learn about the SoftMax function in the coming sections of this post.

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.

K nearest neighbour algorithm is very simple. It works based on 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 data for KNN algorithm consist of several multivariate attributes name that will be used to classify. The data of KNN can be any measurement scale from ordinal, nominal,
to quantitative scale but for the moment let us deal with only quantitative and binary(nominal).

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.

Random forest algorithm can use both for classification and the regression kind of problems. As the name suggest, this algorithm creates the forest with several trees.

In general, the more trees in the forest the more robust the forest looks like. In the same way in the random forest classifier, the higher the number of trees in the forest gives the high accuracy results.

If you know the decision tree algorithm. You might be thinking are we creating more number of decision trees and how can we create more number of decision trees. As all the calculation of nodes selection will be same for the same dataset.

To perform the prediction using the trained random forest algorithm we need to pass the test features through the rules of each randomly created trees. Suppose let's say we formed 100 random decision trees to from the random forest.

Each random forest will predict different target (outcome) for the same test feature. Then by considering each predicted target votes will be calculated. Suppose the 100 random decision trees are prediction some 3 unique targets x, y, z then the votes of x are nothing but out of 100 random decision trees how many trees prediction is x.

Likewise, for other 2 targets (y, z). If x is getting high votes. Let's say out of 100 random decision trees 60 trees are predicting the target will be x. Then the final random forest returns the x as the predicted target.

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.

AdaBoost can be used to boost the performance of any machine learning algorithm. It is best used with weak learners. AdaBoost was the first really successful boosting algorithm developed for binary classification. It is the best starting point for understanding boosting. The most suited and therefore most common algorithm used with AdaBoost are decision trees with one level. Because these trees are so short and only contain one decision for classification, they are often called decision stumps. Each instance in the training dataset is weighted. The initial weight is set to :

$$weight(xi) = 1/n$$

Where xi is the i'th training instance and n is the number of training instances. A weak classifier (decision stump) is prepared on the training data using the weighted samples. Only binary (two-class) classification problems are supported, so each decision stump makes one decision on one input variable and outputs a +1.0 or -1.0 value for the first or second-class value. The misclassification rate is calculated for the trained model. Traditionally, this is calculated as:

$$error = (correct – N) / N$$

Where error is the misclassification rate, correct are the number of training instance predicted correctly by the model and N is the total number of training instances.
For example, if the model predicted 78 of 100 training instances correctly the error or misclassification rate would be (78-100)/100 or 0.22. This is modified to use the weighting of the training instances:

$$error = sum(w(i) * terror(i)) / sum(w)$$

Which is the weighted sum of the misclassification rate, where w is the weight for training instance i and terror is the prediction error for training instance I which is 1 if misclassified and 0 if correctly classified.
For example, if we had 3 training instances with the weights 0.01, 0.5 and 0.2. The predicted values were -1, -1 and -1, and the actual output variables in the instances were -1, 1 and -1, then the terrors would be 0, 1, and 0. The misclassification rate would be calculated as:

$$error = (0.01*0 + 0.5*1 + 0.2*0) / (0.01 + 0.5 + 0.2)$$

or

$$error = 0.704$$

A stage value is calculated for the trained model which provides a weighting for any predictions that the model makes. The stage value for a trained model is calculated as follows:

$$stage = ln((1-error) / error)$$

Where stage is the stage value used to weight predictions from the model, ln () is the natural logarithm and error is the misclassification error for the model. The effect of the stage weight is that more accurate models have more weight or contribution to the final prediction.

The training weights are updated giving more weight to incorrectly predicted instances, and less weight to correctly predicted instances.

For example, the weight of one training instance (w) is updated using:

$$w = w * exp(stage * terror)$$

Where w is the weight for a specific training instance, exp() is the numerical constant e or Euler's number raised to a power, stage is the misclassification rate for the weak classifier and terror is the error the weak classifier made predicting the output variable for the training instance, evaluated as:

$$terror = 0$$

if (y == p), otherwise 1 Where y is the output variable for the training instance and p is the prediction from the weak learner.

This has the effect of not changing the weight if the training instance was classified correctly and making the weight slightly larger if the weak learner misclassified the instance.

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.

```
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

```
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.

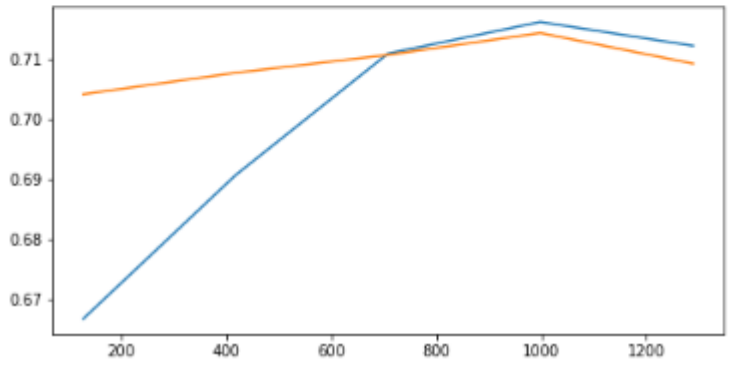
1.Logistic Regression

```
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

As my data is unbalanced and

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

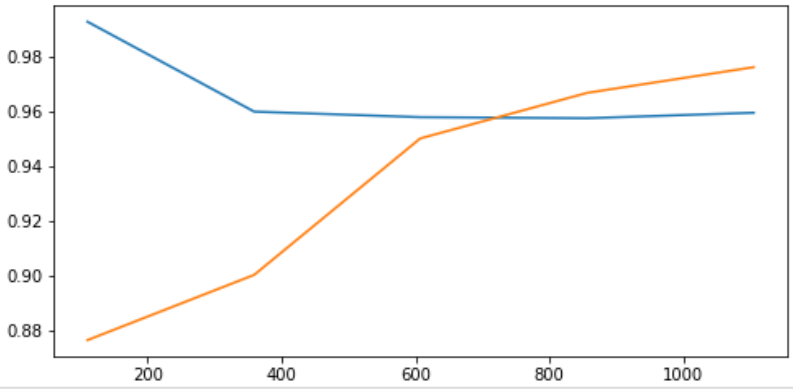
```
0.9393063583815029
```

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>]
```

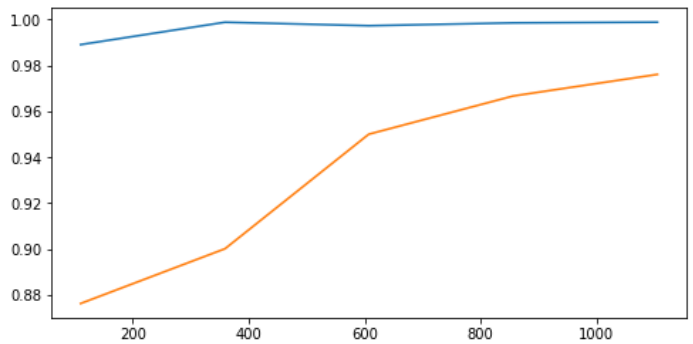


4.Random Forest Classifier

```
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.
After Encoding the data, I got the data values for each record for every attribute either 0,1,2,3 or 4. So from this we can get to know that there are no outliers in the data. Even for unseen data it will work fine because even if we introduce record to predict mostly the new record will also consist of the same data values for all the attributes.

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%)

```
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}
```

```
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.

Our model works well with maximum six features
n_estimators: 50

max_depth: 20

Our model works well with default n_estimator value i.e.,50. That's why our model performance does not change even after parameter tuning. We got same F-score even after applying GridSearchCV technique to our solution model.

```
0.9797687861271677
0.9710982658959537
0.9710982658959537
0.9855491329479769
0.9653179190751445
0.9797687861271677
0.9710982658959537
0.9653179190751445
0.9624277456647399
0.9624277456647399
```

The above values are the scores of RFC with different random_states. By observing them we can say that with the change of random_state the score does not affect much, it stays near to 97.

Justification

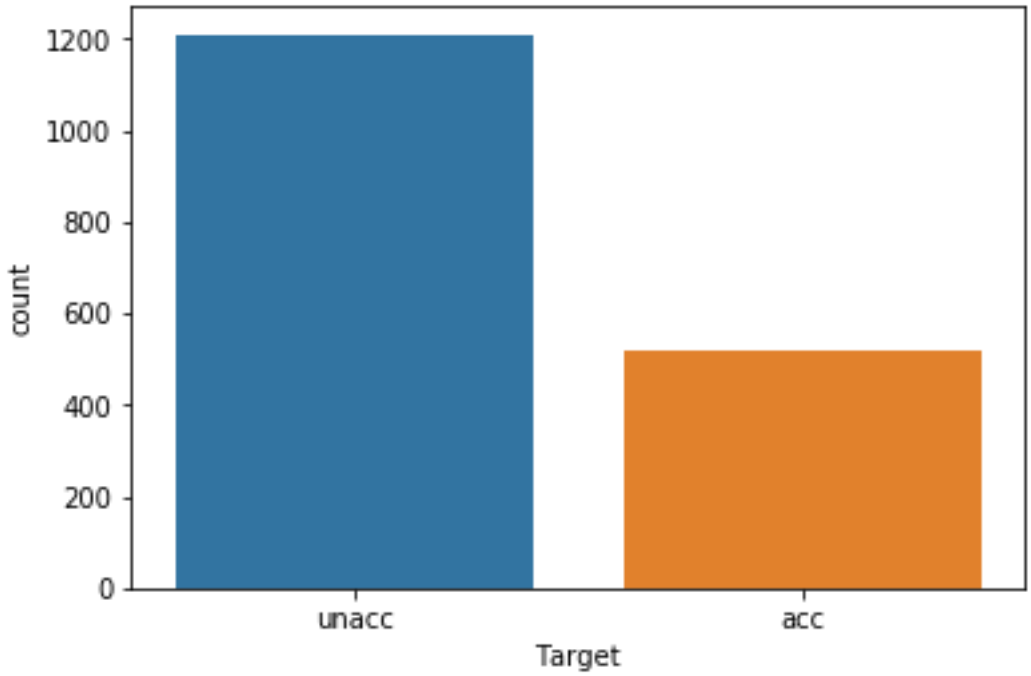
The Logistic regression is used as benchmark model. FBeta score of benchmark model is reference and other model will be judging to perform better if their fbet a score will be greater than Logistic regression model. Accuracy, f-beta score an d confusion matrix and will try to get better results in the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

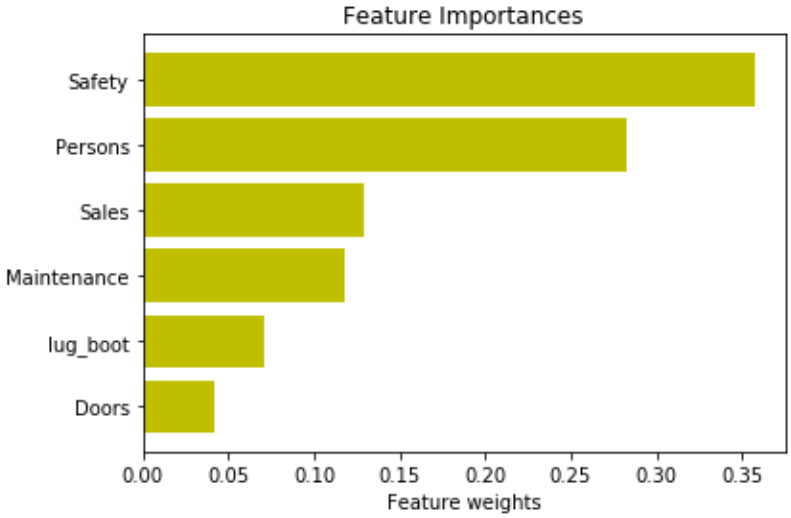
V. Conclusion

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.

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

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

5. http://dataaspirant.com/2017/03/02/how-logistic-regression-model-works/

6. http://people.revoledu.com/kardi/tutorial/KNN/HowTo_KNN.html

7. http://dataaspirant.com/2017/05/22/random-forest-algorithm-machine-learing/

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