

Machine Learning Engineer Nanodegree

Capstone Project Report

Car Evaluation

Ramakrishna

June 25th, 2018

I. Definition

Project Overview

In this project I am going to take a car evaluation dataset which is from Automobile Marketing Domain. In today's Automobile marketing domain every company or seller wants to sell more and more products to customers. In order to make that happen the sellers should know what type of automobiles the customers prefer based on some features like safety, maintainence etc. The Automobile marketing domain has drastically changed from the past few decades with the invention of more and more new models. In order to survive in this domain the company should know the customer requirements and design their models according to them.

Some other machine learning prediction problems similar to mine are:

1. <https://archive.ics.uci.edu/ml/datasets/Wine+Quality>
2. <https://archive.ics.uci.edu/ml/datasets.html>
3. <https://archive.ics.uci.edu/ml/datasets/Contraceptive+Method+Choice>

Problem Statement

Here the problem is Car Acceptability by the customers i.e., based on the some features of a car we have to 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 etc. We have to calculate accuracy and F-score for our training set and check it works fine for our testing data among different models.

Metrics

The performance metric I used is f1-score because f1-score includes precision and recall. We can also take accuracy as my performance metric but my data is unbalanced so I cannot go with accuracy as my metric because it measures the ratio of correct predictions to the total number of cases evaluated hence I used f1-score as my performance metric.

$$F1 = 2 * (\text{precision} * \text{recall}) / (\text{precision} + \text{recall})$$

II. Analysis

Data Exploration

The dataset I have taken is from UCI machine learning repository (<https://archive.ics.uci.edu/ml/datasets/Car+Evaluation>) which contains 1728 records and 6 attributes. The attributes with different values are as follows:

Buying(buying price): vhigh, high, med, low.

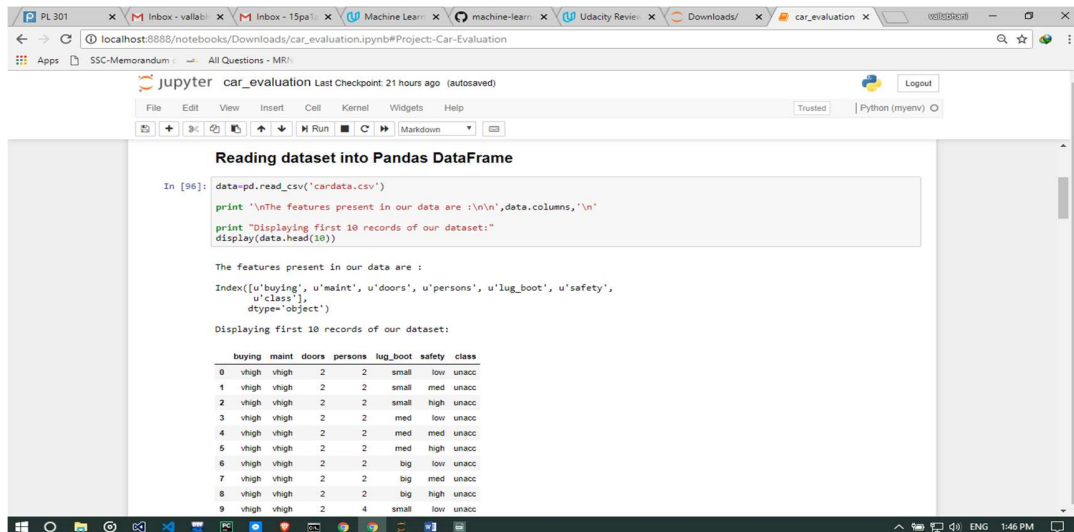
maint(maintainence price): vhigh, high, med, low.

doors(No of Doors the car have): 2, 3, 4, 5more.

persons(No of persons that car can fit): 2, 4, more.

lug_boot(luggage boot size): small, med, big.

safety(safety level): low, med, high.



```
In [96]: data=pd.read_csv('cardata.csv')
print '\n\nThe features present in our data are :\n\n',data.columns,'\n\n'
print "Displaying first 10 records of our dataset:"
display(data.head(10))

The features present in our data are :
Index([u'buying', u'maint', u'doors', u'persons', u'lug_boot', u'safety',
       u'class'],
      dtype='object')

Displaying first 10 records of our dataset:
```

	buying	maint	doors	persons	lug_boot	safety	class
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
5	vhigh	vhigh	2	2	med	high	unacc
6	vhigh	vhigh	2	2	big	low	unacc
7	vhigh	vhigh	2	2	big	med	unacc
8	vhigh	vhigh	2	2	big	high	unacc
9	vhigh	vhigh	2	4	small	low	unacc

Based on the above attributes we have to give car acceptability labels as unacc, acc, good, vgood. I considered these attributes because the customers first see the buying price of the car and how many members can fit in the car and the maintenance cost it will take and mainly safety levels of the car which are very important compared to other features.

```
] : data.describe()

]:
```

	buying	maint	doors	persons	lug_boot	safety	class
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	med	med	3	more	med	med	unacc
freq	432	432	432	576	576	576	1210

By seeing the above data all the column attributes are equally dist.

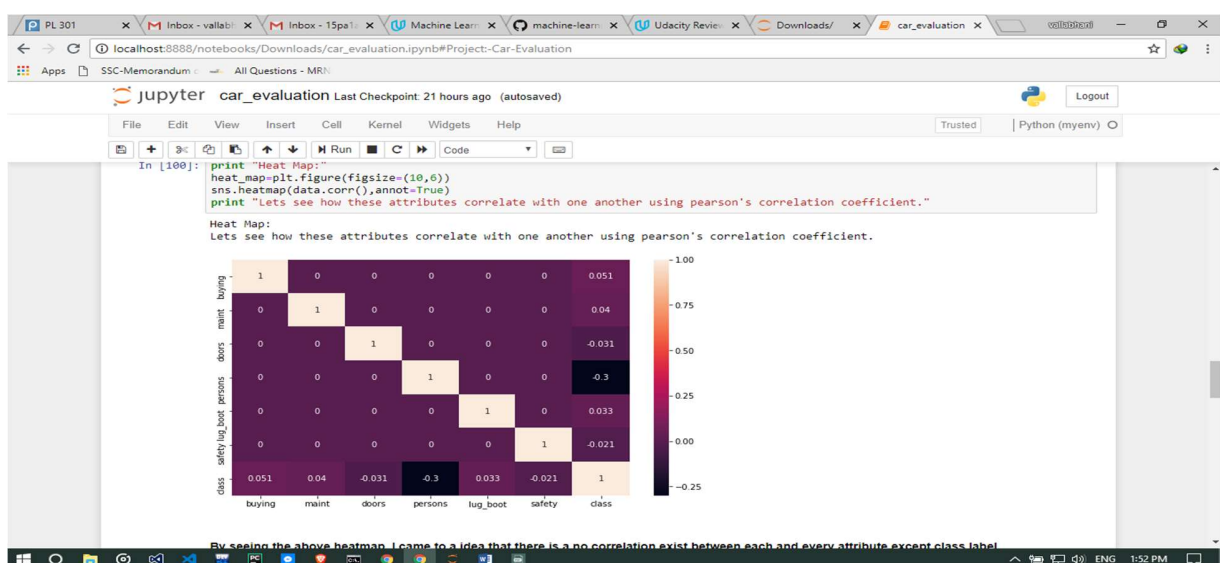
By seeing the above stats we can say that that all the features are equally distributed except the class label. Which says that our data is unbalanced.

Exploratory Visualization

The below graph shows how my class label is distributed in the data:



The below heat map describes how my attributes are correlated with each other :



By above heatmap the attributes are not at all correlating with each other and with persons attribute is less correlating with class label. I choose heatmap because it is the most easiest way to analyze correlation between attributes.

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.KNNClassifier

I choose this algorithm because of its easy of interpretation and low calculation time. This algorithm is more 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 k-NN algorithm 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 hyper parameters. 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.

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

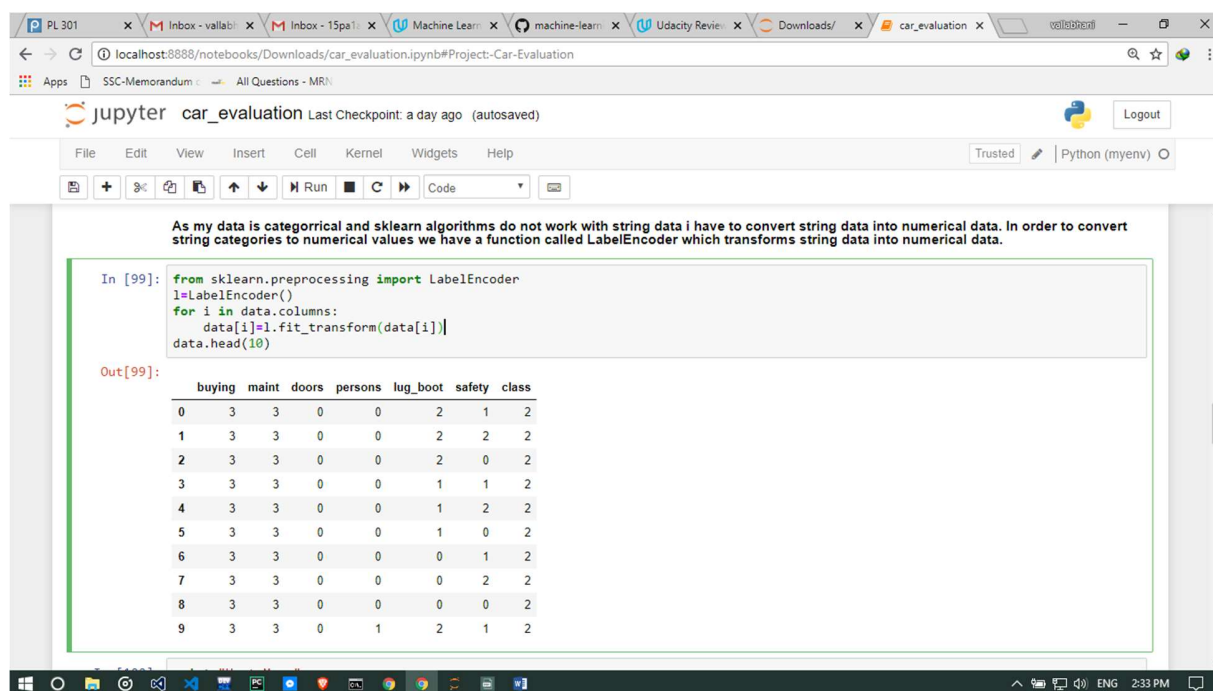
First I will use logistic regression as my benchmark model and calculate it's F-score to test the performance of the model. I got around 61% F-score which is less hence I implement another model and compare its F-score to test its performance compared to Benchmark model. So by implementing other models we will get a better F-score than Benchmark model.

III. Methodology

Data Preprocessing

The preprocessing steps I took are:

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



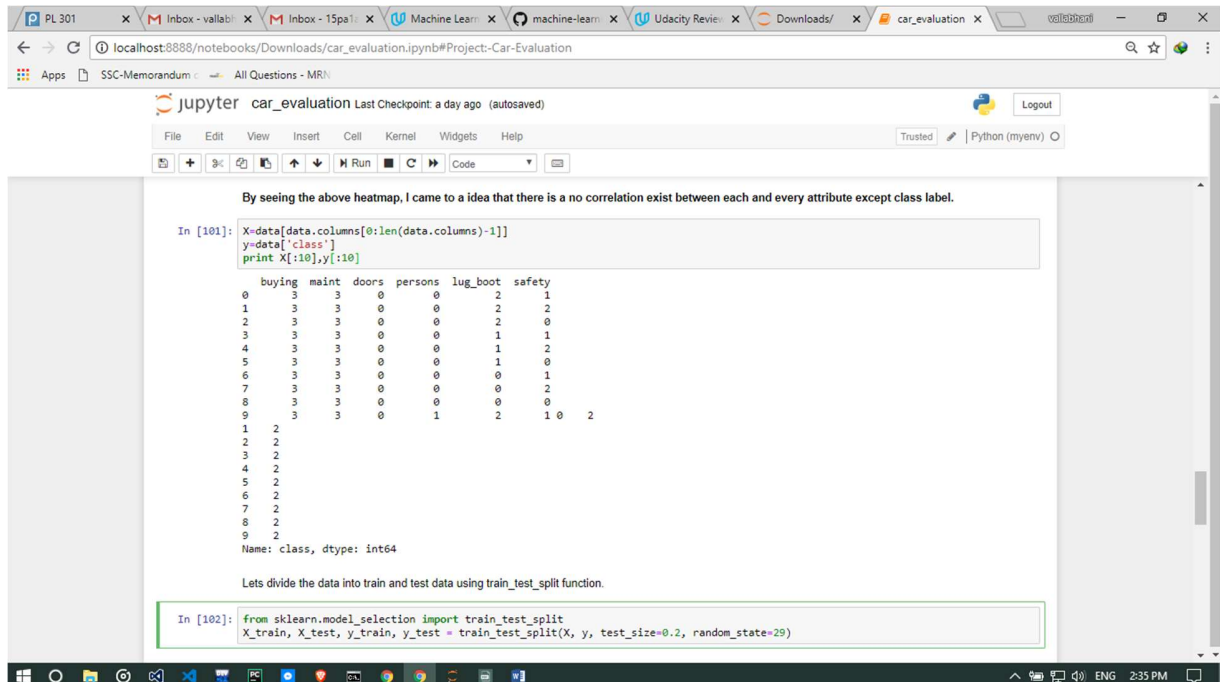
The screenshot shows a Jupyter Notebook interface with a browser window at the top. The notebook is titled 'car_evaluation' and shows a code cell with the following Python code:

```
In [99]: from sklearn.preprocessing import LabelEncoder
         le=LabelEncoder()
         for i in data.columns:
             data[i]=le.fit_transform(data[i])
         data.head(10)
```

The output of the code cell is a table showing the first 10 rows of the data after conversion:

	buying	maint	doors	persons	lug_boot	safety	class
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
5	3	3	0	0	1	0	2
6	3	3	0	0	0	1	2
7	3	3	0	0	0	2	2
8	3	3	0	0	0	0	2
9	3	3	0	1	2	1	2

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



```
In [101]: X=data[data.columns[0:len(data.columns)-1]]
y=data['class']
print X[:10],y[:10]
```

	buying	maint	doors	persons	lug_boot	safety
0	3	3	0	0	2	1
1	3	3	0	0	2	2
2	3	3	0	0	2	0
3	3	3	0	0	1	1
4	3	3	0	0	1	2
5	3	3	0	0	1	0
6	3	3	0	0	0	1
7	3	3	0	0	0	2
8	3	3	0	0	0	0
9	3	3	0	1	2	1 0 2

```
Name: class, dtype: int64
```

Lets divide the data into train and test data using `train_test_split` function.

```
In [102]: from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X, y, test_size=0.2, random_state=29)
```

Implementation

I choose implement the following three algorithms:

1. Logistic Regression
2. KNNClassifier
3. RandomForestClassifier

After I calculated the F-scores of these three models to check their performance by fitting the data into the classifiers and calculated the F1-score of the three models using F1-score function from sklearn.metrics.

```
In [103]: from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import f1_score

As my data is unbalanced using accuracy as my metric is not a good option hence i will go with f1_score which is best among all metrics.

In [114]: for i in [LogisticRegression, KNeighborsClassifier, RandomForestClassifier]:
mod=i()
mod=mod.fit(X_train,y_train)
y_pred=mod.predict(X_test)
f1score=f1_score(y_test,y_pred,average='weighted')
print "using",i,"i got an f1_score of",f1score

using <class 'sklearn.linear_model.logistic.LogisticRegression'> i got an f1_score of 0.6100884895666437
using <class 'sklearn.neighbors.classification.KNeighborsClassifier'> i got an f1_score of 0.8896111598243296
using <class 'sklearn.ensemble.forest.RandomForestClassifier'> i got an f1_score of 0.9578446519387169

C:\Users\valia\Anaconda2\lib\site-packages\sklearn\metrics\classification.py:1135: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
'precision', 'predicted', average, warn_for)

From the above f1_score values of three classifiers we can say that RandomForestClassifier is giving better F1_score of 95.78, Hence we can take RandomForestClassifier as our solution model.

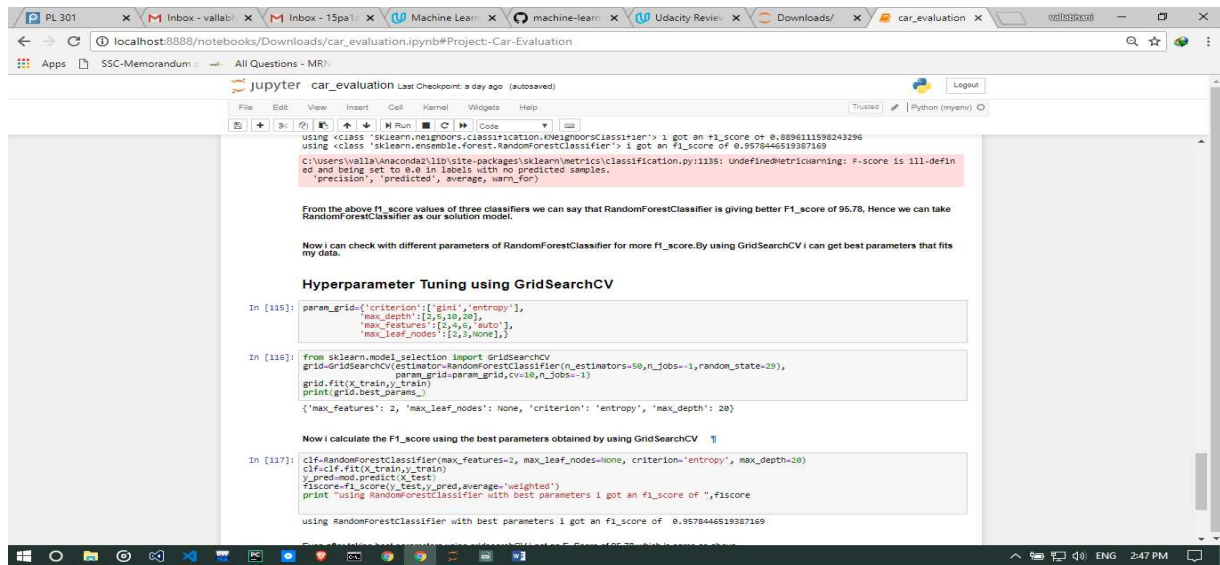
Now i can check with different parameters of RandomForestClassifier for more f1_score.By using GridSearchCV i can get best parameters that fits my data.

Hyperparameter Tuning using GridSearchCV
```

I choose RandomForestClassifier as my solution model as it has high F-score compared to remaining two models. The initial model has a f1 score value of 61% whereas KNN classifier has a f1 score of 88.9% which is quite well compared to Logistic Regression. Finally I got an f1 score of around 95.7 by using RFC. After taking RFC as my solution model I tuned the parameters to increase the performance of the model using GridSearchCV. While implementing GridSearchCV it got little complicated because I don't know which parameters to choose for paramGrid which is passed to GridSearchCV.

Refinement

I used Grid search technique to refine my solution model. But the initial (before tuning) and final solutions (after tuning) are same. i.e., I got an F-score of 95.78% before tuning and after tuning also I got the same result 95.78%.



```
using <class 'sklearn.neighbors.classification.kneighborsclassifier'> I got an f1_score of 0.8896111598243296
using <class 'sklearn.ensembleforest.randomforestclassifier'> I got an f1_score of 0.9578446519387169
c:\users\vallab\anaconda2\lib\site-packages\sklearn\metrics\classification.py:1195: UndefinedMetricWarning: F-score is ill-defined and being set to 0.0 in labels with no predicted samples.
  'precision', 'predicted', average, warn_for)

From the above f1_score values of three classifiers we can say that RandomForestClassifier is giving better F1_score of 95.78, Hence we can take RandomForestClassifier as our solution model.

Now I can check with different parameters of RandomForestClassifier for more f1_score. By using GridSearchCV I can get best parameters that fits my data.

Hyperparameter Tuning using GridSearchCV

In [115]: param_grid={'criterion':['gini','entropy'],
                    'max_depth':[0,10,20],
                    'max_features':['0.5','auto'],
                    'max_leaf_nodes':[2,2,None],}

In [116]: from sklearn.model_selection import GridSearchCV
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': 2, 'max_leaf_nodes': None, 'criterion': 'entropy', 'max_depth': 20}

Now I calculate the F1_score using the best parameters obtained by using GridSearchCV

In [117]: clf=RandomForestClassifier(max_features=2, max_leaf_nodes=None, criterion='entropy', max_depth=20)
clf=clf.fit(X_train,y_train)
y_pred=clf.predict(X_test)
f1score=f1_score(y_test,y_pred,average='weighted')
print "using RandomForestClassifier with best parameters I got an f1_score of ",f1score

using RandomForestClassifier with best parameters I got an f1_score of 0.9578446519387169
```

IV. Results

Model Evaluation and Validation

Finally our solution got an F-score of 95.78 which is pretty good compared to our benchmark model. The solution model parameters are:

max_features: 2

Our model works well with maximum two features.

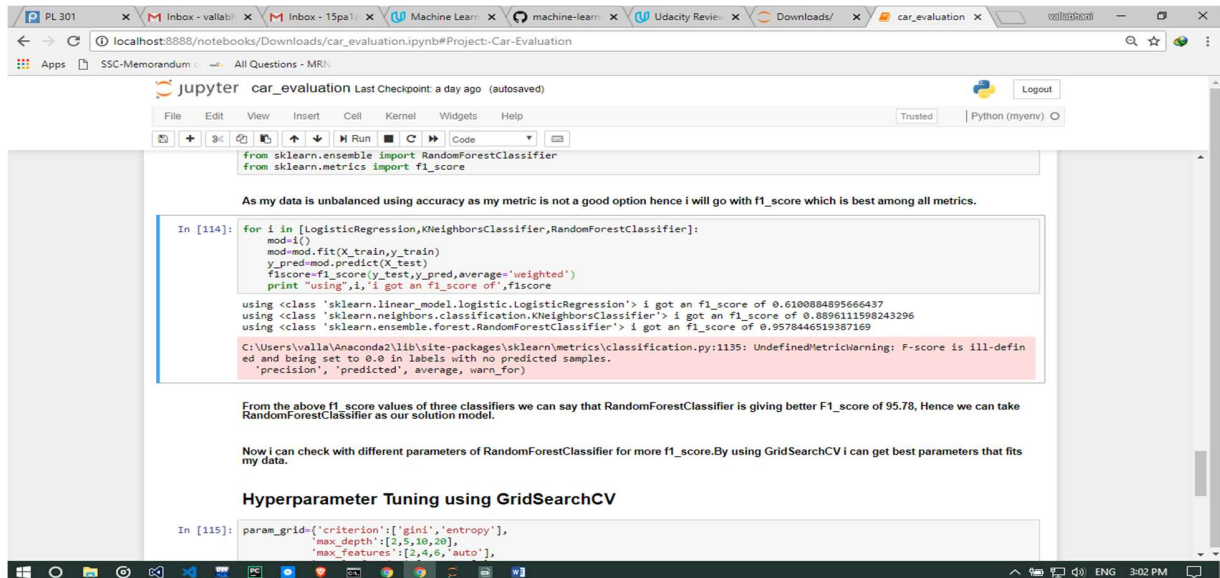
n_estimators: 50

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.

```
0.9653179190751445
0.9508670520231214
0.9566473988439307
0.953757225433526
0.9508670520231214
0.9479768786127167
0.953757225433526
0.953757225433526
0.9421965317919075
0.9450867052023122
```

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 effect much, it stays near to 95.

Justification



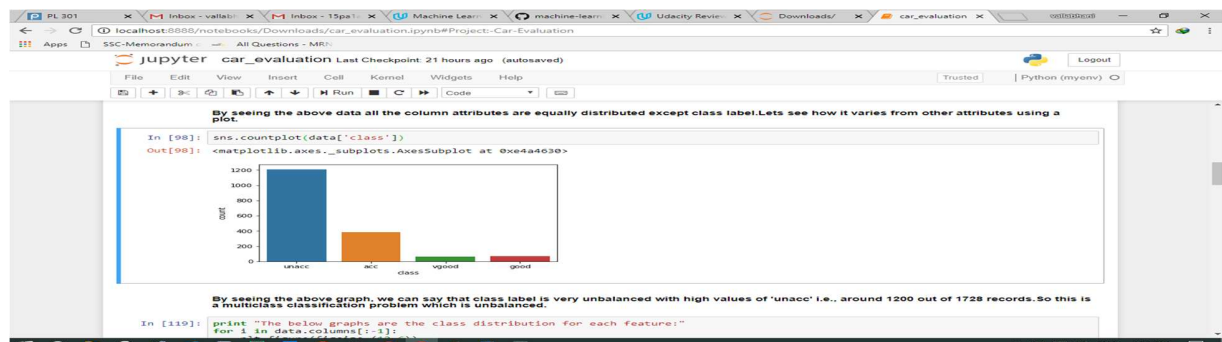
The screenshot shows a Jupyter Notebook interface with the following content:

- Imports: `from sklearn.ensemble import RandomForestClassifier` and `from sklearn.metrics import f1_score`.
- Text: "As my data is unbalanced using accuracy as my metric is not a good option hence i will go with f1_score which is best among all metrics."
- Code cell [114]: A loop testing three models: LogisticRegression, KNeighborsClassifier, and RandomForestClassifier. It prints the f1_score for each. The output shows f1_scores of approximately 0.61, 0.89, and 0.96 respectively.
- Text: "From the above f1_score values of three classifiers we can say that RandomForestClassifier is giving better F1_score of 95.78, Hence we can take RandomForestClassifier as our solution model."
- Text: "Now i can check with different parameters of RandomForestClassifier for more f1_score. By using GridSearchCV i can get best parameters that fits my data."
- Section Header: "Hyperparameter Tuning using GridSearchCV"
- Code cell [115]: `param_grid = {'criterion': ['gini', 'entropy'], 'max_depth': [2, 5, 10, 20], 'max_features': [2, 4, 6, 'auto']}`

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

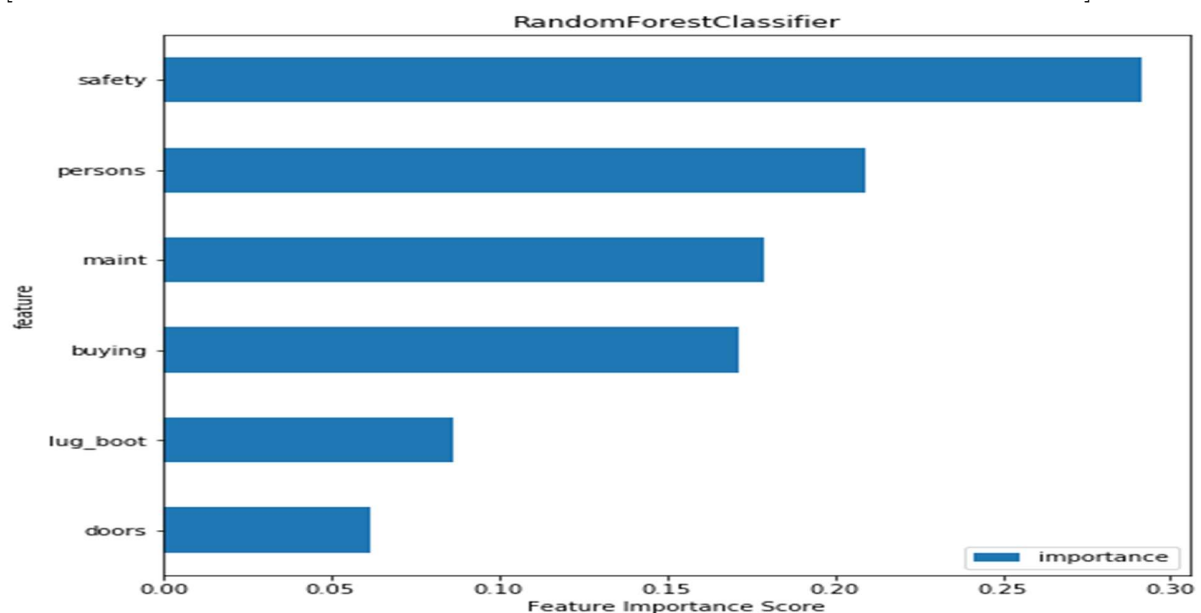
V. Conclusion

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.

```
[0.17142423 0.17908856 0.06181634 0.20947376 0.08664885 0.29154826]
```



Above the feature importance ranks and graph. By seeing them I can say that doors attribute is less important compared to other attributes.

Reflection

1. First I take 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. Then as my data is categorical I converted them into numerical values using LabelEncoder.
5. After I plot a heat map to see how my attributes are correlated with each other and with class label.

6. Then I selected three different models for my problem and selected LogisticRegression as my benchmark model and I got very less performance.
7. I check the performance with other two models as well by calculating F-score. Among three models RandomForestClassifier performs well with a F-score of 95.78 and I choose it as my solution model.
8. Later I use GridSearchCV technique to improve the performance by selecting best combination of parameters for my model but I got same F-score value as before tuning.
9. I find difficult when I am using GridsearchCV technique because it gave me same result after parameter tuning, I thought it will increase but it didn't. Overall I enjoyed doing the project very much.

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 Adaboost ensemble model 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.