**Data Mining Final Project**

**Name:** Shang-Yung Hsu

**ID:** sh555

**Tool**

Sklearn

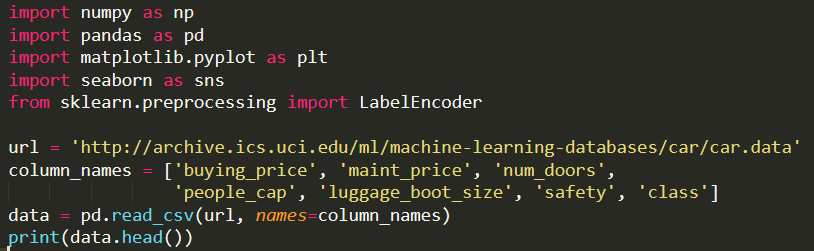
Matplotlib

panda

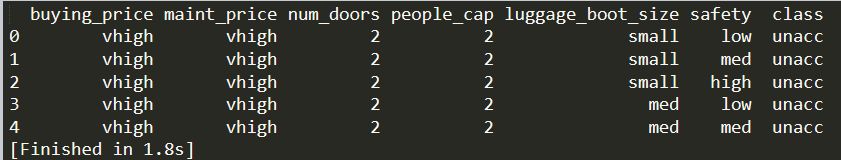
**Check and Modify the Dataset**

The car evaluation dataset is from <https://archive.ics.uci.edu/ml/datasets/Car+Evaluation>.

I give the column name to each column.

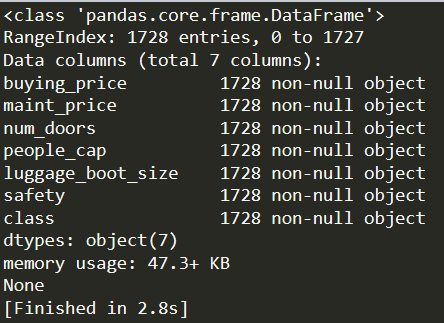




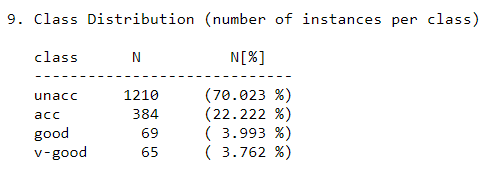


Check the dataset if there is any null value.

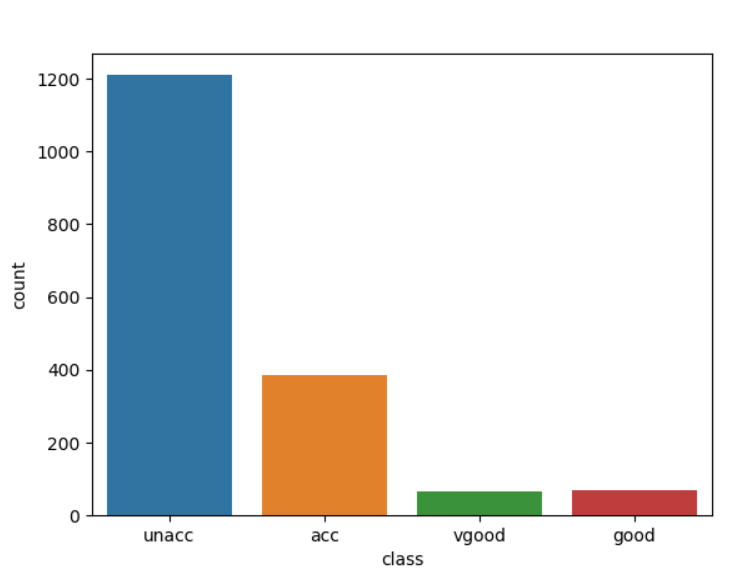




Seeing the description of the car evaluation dataset, you will find that the 'class ' distribution is really unbalance.

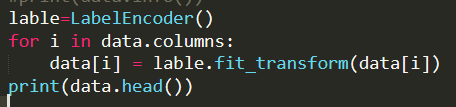


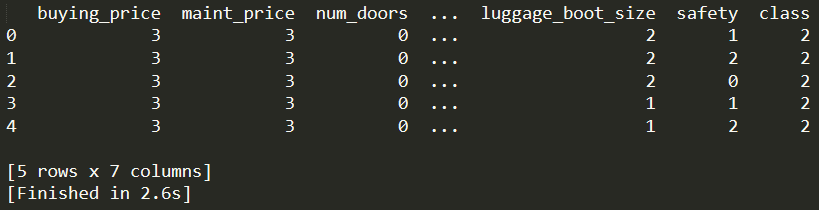




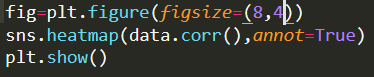
In this case, it will have an imbalanced multiclass classification problem. So that, the accuracy may not be the best stander to check this model.

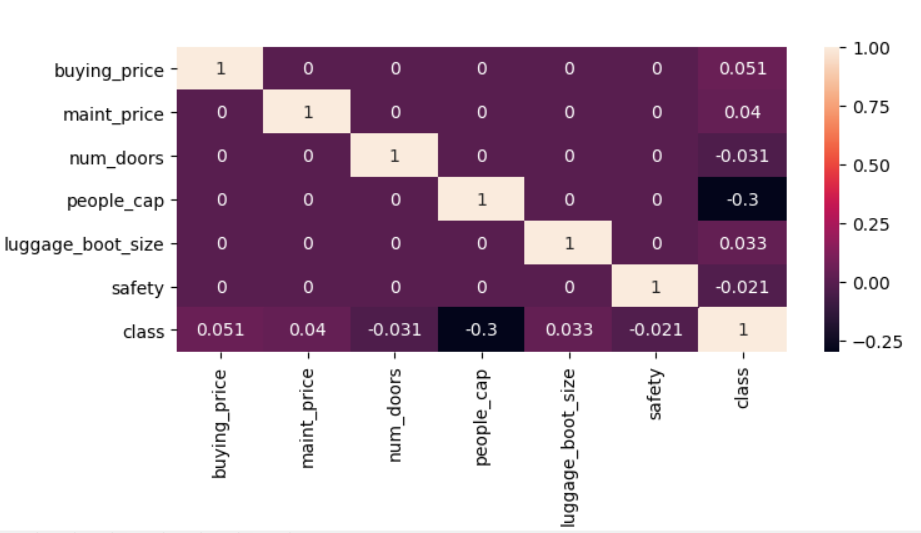
As scikit-learn algorithms do not work with string values, so I converted the string categories to integers.





Looking at the Heatmap of the columns on dataset with each other. Will find that most of the columns shows very weak correlation with 'class'. ('persons' column is the weakest). Other columns except 'class' shows no correlation with each other.





So, doing any analysis on them might not give any useful output.

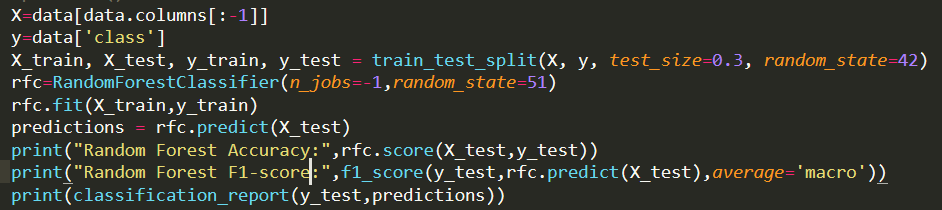
Training and Analyze dataset

X is the dataframe containing input data / features

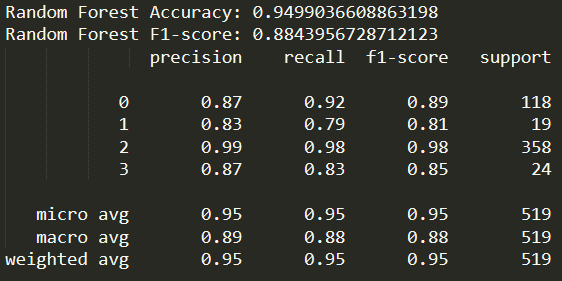
y is the series which has results which are to be predicted.

**Random Forest**

Divide data in train and test sets and train by Random Forest Algorithm.

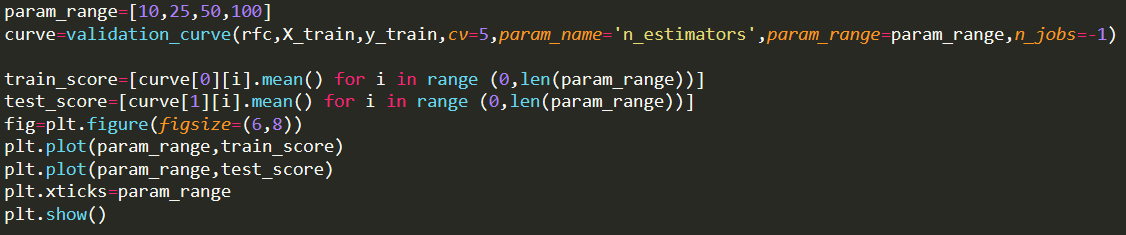


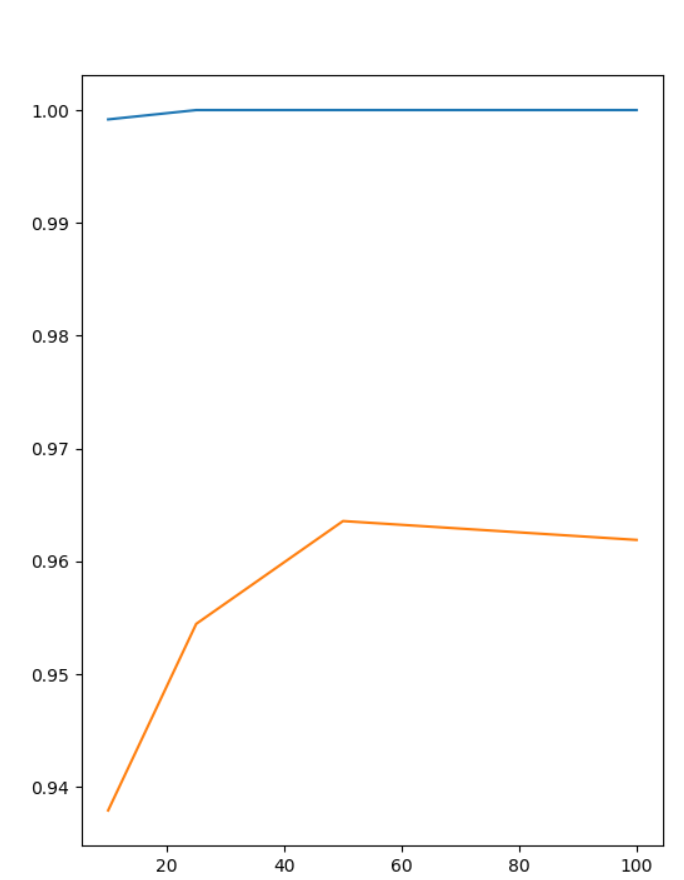
Because this dataset is an imbalanced dataset, so like I mention before, we should also consider the f1 score instead of only see the accuracy.



The basic model of RFC is giving almost 95% accuracy.

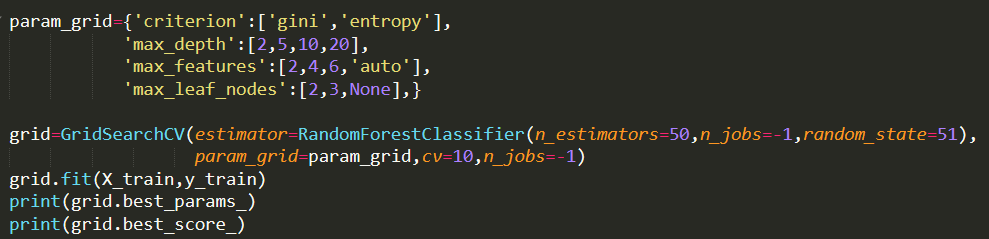
Then, we should check the effect of n\_estimators on the model.





So, with the increasing n\_estimators, the accuracy is also increasing. The best evaluating value is at n\_estimators=50. After n\_estimators=50, model starts overfitting. Therefore, we reached the accuracy to almost 96%.

Also, we can use the GridSearch to get the combination of best parameters. It is a simpler way to get all the parameters, but it takes more time to complete.

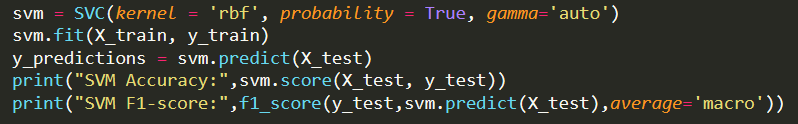




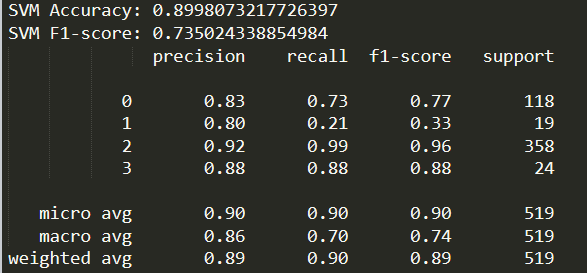
So, with those parameters. We can reach 98.4% accuracy.

**Support Vector Machine - SVM**

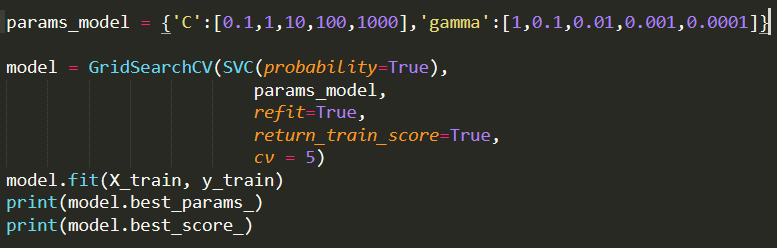
Divide data in train and test sets and train by SVM Algorithm.



like I mention before, we should also consider the f1 score instead of only see the accuracy.



We can also use the GridSearch to find the best parameters. (Check the kernel separate)

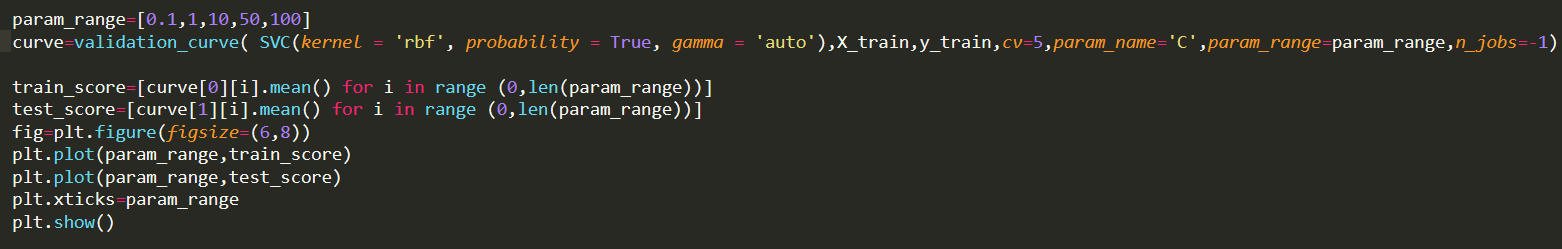




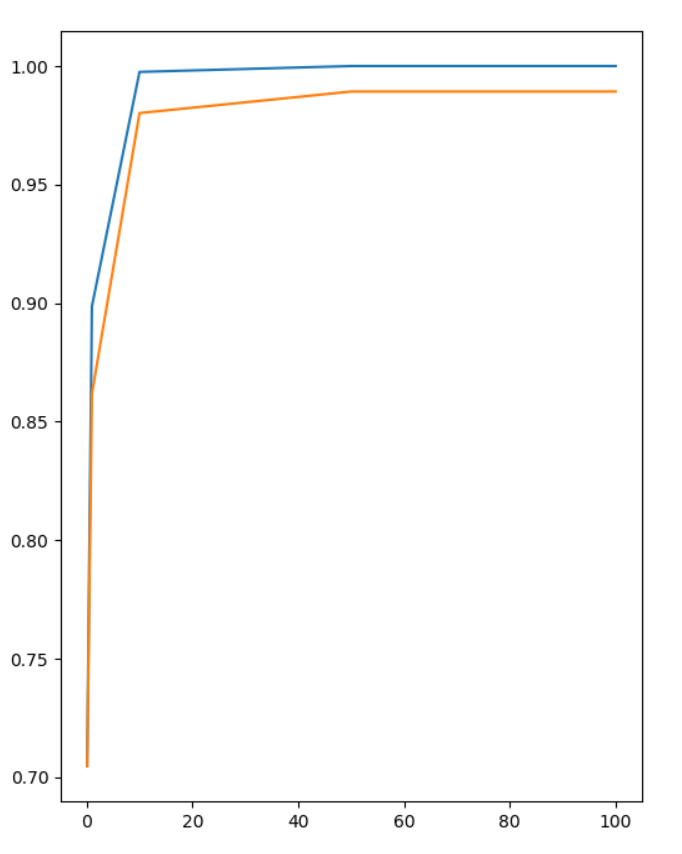
It looks high accuracy, but the value of C is too large. When C tends to infinity, the problem is that samples with classification errors are not allowed. That is a hard-margin SVM problem (overfitting). So, it might not the result we want.

The reason why it causes the overfitting is because of the dataset is imbalance.

Then, we should check the effect value of C in this model.



This is for kernel rbf.

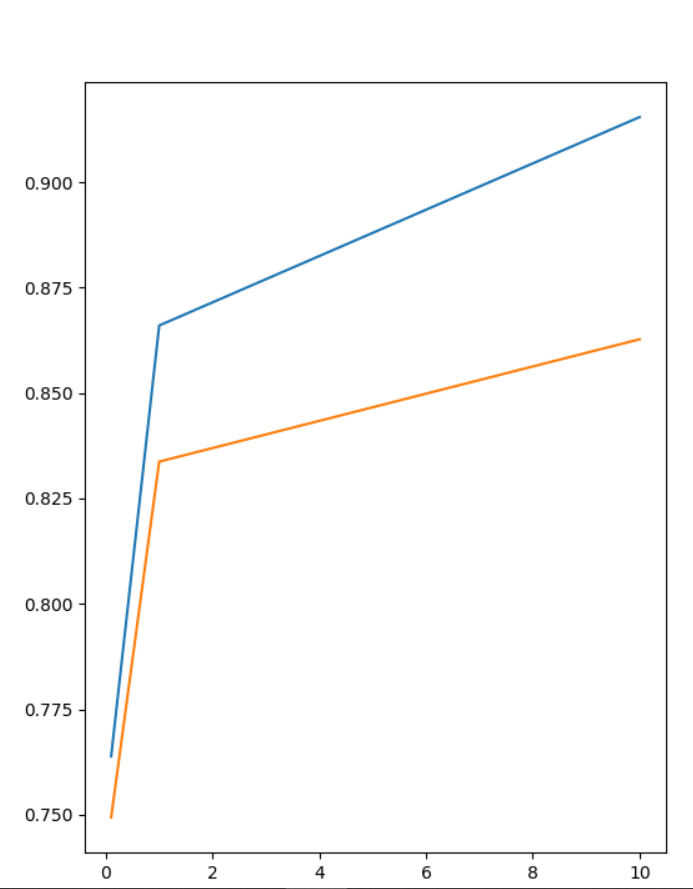


For the kernel rbf. When C is bigger than 15, it will overfitting. The best value of C will be 15 for kernel rbf.



This is for kernel poly.

When the value of C equal to 10.



For the kernel poly. When C is bigger than 1. It will be overfitting. Also the accuracy is lower than the kernel rbf.

So, the best kernel to choose is rbf.

**Conclusion**

Random Forest Classifier is the better model for this data with following parameters: n\_estimators: 50, criterion: entropy, max\_depth: 10, max\_features: 6, max\_leaf\_nodes: None

Achieve 98.43% accuracy with this model

**Code**

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.preprocessing import LabelEncoder

from sklearn.model\_selection import train\_test\_split,cross\_val\_score

from sklearn.metrics import classification\_report, confusion\_matrix

from sklearn.model\_selection import validation\_curve

from sklearn.svm import SVC

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import f1\_score

from sklearn.model\_selection import GridSearchCV

from matplotlib.colors import ListedColormap

url = 'http://archive.ics.uci.edu/ml/machine-learning-databases/car/car.data'

column\_names = ['buying\_price', 'maint\_price', 'num\_doors',

'people\_cap', 'luggage\_boot\_size', 'safety', 'class']

data = pd.read\_csv(url, names=column\_names)

#print(data.head())

#print(data.info())

#sns.countplot(data['class'])

#plt.show()

lable=LabelEncoder()

for i in data.columns:

data[i] = lable.fit\_transform(data[i])

#print(data.head())

#fig=plt.figure(figsize=(8,4))

#sns.heatmap(data.corr(),annot=True)

#plt.show()

X=data[data.columns[:-1]]

y=data['class']

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.3, random\_state=42)

rfc=RandomForestClassifier(n\_jobs=-1,random\_state=51)

rfc.fit(X\_train,y\_train)

predictions = rfc.predict(X\_test)

#print("Random Forest Accuracy:",rfc.score(X\_test,y\_test))

#print("Random Forest F1-score:",f1\_score(y\_test,rfc.predict(X\_test),average='macro'))

#print(classification\_report(y\_test,predictions))

param\_range=[10,25,50,100]

curve=validation\_curve(rfc,X\_train,y\_train,cv=5,param\_name='n\_estimators',param\_range=param\_range,n\_jobs=-1)

train\_score=[curve[0][i].mean() for i in range (0,len(param\_range))]

test\_score=[curve[1][i].mean() for i in range (0,len(param\_range))]

fig=plt.figure(figsize=(6,8))

plt.plot(param\_range,train\_score)

plt.plot(param\_range,test\_score)

plt.xticks=param\_range

#plt.show()

param\_grid={'criterion':['gini','entropy'],

'max\_depth':[2,5,10,20],

'max\_features':[2,4,6,'auto'],

'max\_leaf\_nodes':[2,3,None],}

grid=GridSearchCV(estimator=RandomForestClassifier(n\_estimators=50,n\_jobs=-1,random\_state=51),

#param\_grid=param\_grid,cv=10,n\_jobs=-1)

grid.fit(X\_train,y\_train)

print(grid.best\_params\_)

print(grid.best\_score\_)

svm = SVC(kernel = 'poly', probability = True, gamma = 'auto' , C = 1)

svm.fit(X\_train, y\_train)

y\_predictions = svm.predict(X\_test)

print("SVM Accuracy:",svm.score(X\_test, y\_test))

print("SVM F1-score:",f1\_score(y\_test,svm.predict(X\_test),average='macro'))

#print(classification\_report(y\_test,y\_predictions))

c\_matrix = confusion\_matrix(y\_test,y\_predictions)

print ("confusion matrix:")

print (c\_matrix)

plt.matshow(c\_matrix)

params\_model = {'C':[0.1,10],'gamma':[1,0.1,0.01,0.001,0.0001]}

model = GridSearchCV(SVC(probability=True),

params\_model,

refit=True,

return\_train\_score=True,

cv = 5)

model.fit(X\_train, y\_train)

print(model.best\_params\_)

print(model.best\_score\_)

param\_range=[0.1,1,10]

curve=validation\_curve( SVC(kernel = 'poly', probability = True, gamma = 'auto'),X\_train,y\_train,cv=5,param\_name='C',param\_range=param\_range,n\_jobs=-1)

train\_score=[curve[0][i].mean() for i in range (0,len(param\_range))]

test\_score=[curve[1][i].mean() for i in range (0,len(param\_range))]

fig=plt.figure(figsize=(6,8))

plt.plot(param\_range,train\_score)

plt.plot(param\_range,test\_score)

plt.xticks=param\_range

plt.show()