Open in Colab

```
import pandas as pd
import numpy as np
import random
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix, auc
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn import svm
from sklearn.linear_model import LogisticRegression
import matplotlib.pyplot as plt
from sklearn.metrics import roc_curve
```

```
data = pd.read_csv('C:/Data analysis Project/practical/default.csv')
data['industry'] = data['industry'].str.replace('Other Services \(except Public Administration\)','Oth
data['industry'] = data['industry'].str.replace('Accomodation','Accommodation')
data['industry'] = data['industry'].str.replace('Fishing or Hunting','Fishing and Hunting')
data['industry'] = data['industry'].fillna('Other Services')
data['industry'] = data['industry'].str.strip()
test = data.groupby('industry').size()

data = data.dropna(subset=['years_of_operation'], how='any')
data = data.dropna(subset=['fico_score'], how='any')
data = data[data['fico_score'] != 0]
data = data.dropna(subset=['business_employee_count'], how='any')
data = data.dropna(subset=['Homeowner'], how='any')

data['principal'] = data['principal'].str.replace('$','').str.replace(',','')
data['principal'] = data['principal'].astype(int)
data = data[data['status'] != 'current']
data['status'] = np.where(data['status'] == 'defaulted', 1, 0)
```

```
data.head(5)
```

	credit_rating	industry	state	years_of_operation	principal	interest_rate
198	B	Other Services	CO	3.0	200000	0.1599
619	C	Real Estate and Rental and Leasing	TX	2.0	25000	0.1849
620	A	Other Services	SC	18.0	75000	0.1124
623	C	Transportation and	AR	5.0	91000	0.1949

```
#Descriptive Statistics
data.describe()
```

	years_of_operation	principal	interest_rate	term	status	fi
count	4709.000000	4709.000000	4709.000000	4709.000000	4709.000000	470
mean	10.649529	138240.356764	0.141907	40.885114	0.086855	70
std	7.991417	104803.995393	0.044720	14.908321	0.281652	4
min	0.000000	25000.000000	0.049900	6.000000	0.000000	60
25%	5.000000	60000.000000	0.111900	36.000000	0.000000	67
50%	8.600000	100000.000000	0.139900	36.000000	0.000000	70
75%	13.800000	200000.000000	0.162900	60.000000	0.000000	73
max	67.000000	500000.000000	0.287900	60.000000	1.000000	84

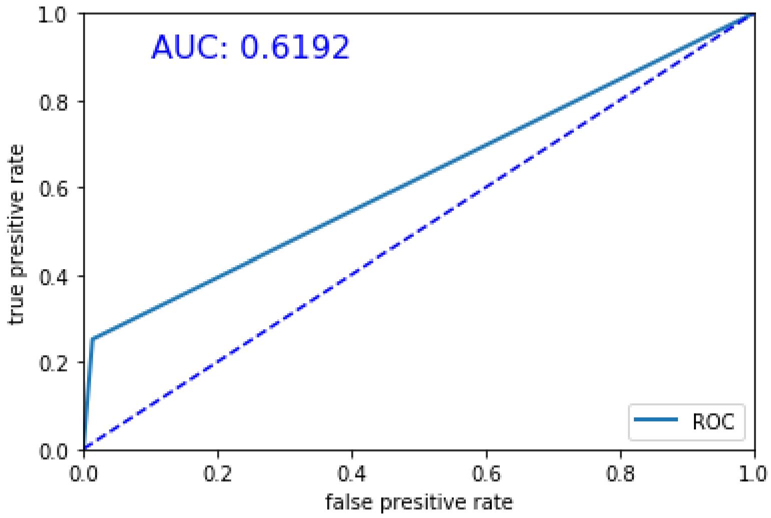
```
#dummy
data = pd.get_dummies(data,columns=['credit_rating'],prefix=['credit_rating'],prefix_sep="_",dummy_na=
data = pd.get_dummies(data,columns=['industry'],prefix=['industry'],prefix_sep="_",dummy_na=False,drop
data = data.reset_index(drop=True)
```

```
#split into train and test
random.seed(110)
train,test=train_test_split(data,test_size=0.3)
train_y = train['status']
train_x = train.drop(['status','state'],axis=1)
test_y = test['status']
test_x = test.drop(['status','state'],axis=1)
```

```
#Logistic regression
lm = LogisticRegression(solver='liblinear')
lm.fit(train_x,train_y)
predict_lm = lm.predict(test_x)
accuracy_lm = accuracy_score(test_y, predict_lm)
tn_lm, fp_lm, fn_lm, tp_lm = confusion_matrix(test_y, predict_lm).ravel()
sensitivity_lm = tp_lm / (tp_lm+fn_lm)
print("Accuracy is %s" %accuracy_lm)
print("Sensitivity is %s" %sensitivity_lm)
print(confusion_matrix(test_y, predict_lm))
```

```
Accuracy is 0.9285208775654635
Sensitivity is 0.25225225225225223
[[1284   18]
 [  83   28]]
```

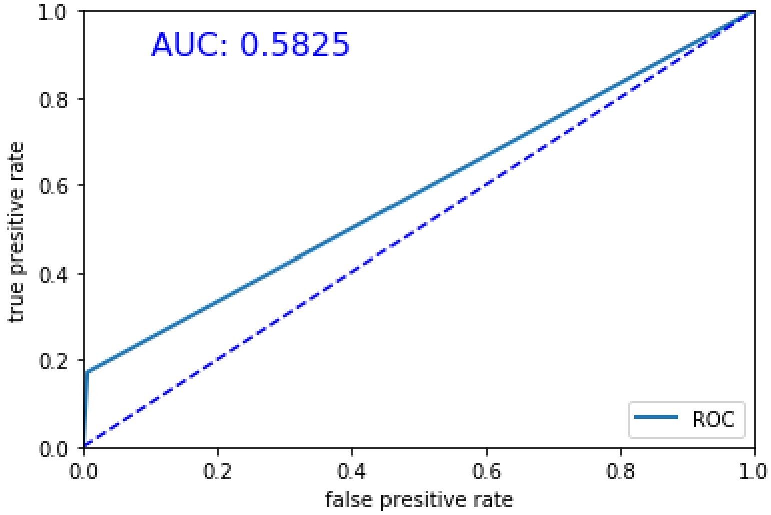
```
#Logistic regression ROC
fpr_lm,tpr_lm,thresholds = roc_curve(test_y, predict_lm,pos_label=1)
auc_lm = round(auc(fpr_lm, tpr_lm),4)
plt.plot(fpr_lm,tpr_lm,linewidth=2,label="ROC")
plt.xlabel("false presitive rate")
plt.ylabel("true presitive rate")
plt.ylim(0,1)
plt.xlim(0,1)
plt.plot([0, 1], [0, 1], '--', color=(0 , 0, 1))
plt.legend(loc=4)
plt.text(0.1,0.9,'AUC: %s' %auc_lm,fontdict={'size':'16','color':'b'})
plt.show()
```



```
#random forest
rf = RandomForestClassifier(n_estimators=1000,random_state=110)
rf.fit(train_x,train_y)
predict_rf = rf.predict(test_x)
accuracy_rf = accuracy_score(test_y, predict_rf)
tn_rf, fp_rf, fn_rf, tp_rf = confusion_matrix(test_y, predict_rf).ravel()
sensitivity_rf = tp_rf / (tp_rf+fn_rf)
print("Accuracy is %s" %accuracy_rf)
print("Sensitivity is %s" %sensitivity_rf)
print(confusion_matrix(test_y, predict_rf))
```

```
Accuracy is 0.9292285916489739
Sensitivity is 0.17117117117117117
[[1294    8]
 [  92   19]]
```

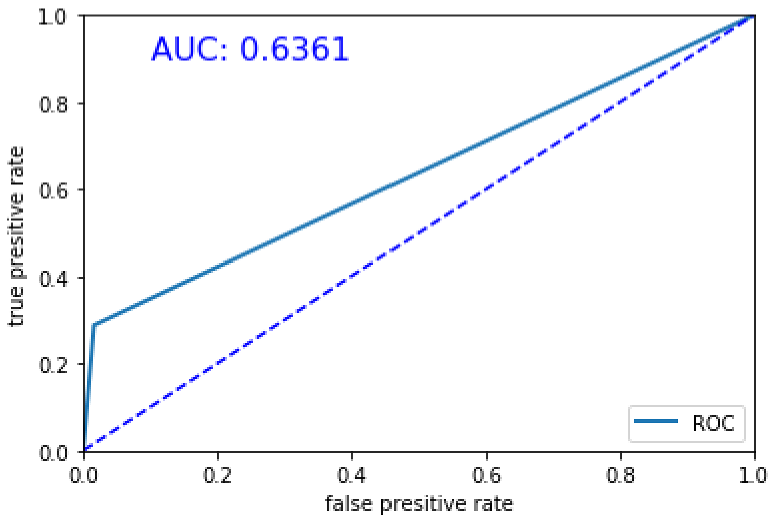
```
#random forest ROC
fpr_rf,tpr_rf,thresholds = roc_curve(test_y, predict_rf,pos_label=1)
auc_rf = round(auc(fpr_rf, tpr_rf),4)
plt.plot(fpr_rf,tpr_rf,linewidth=2,label="ROC")
plt.xlabel("false presitive rate")
plt.ylabel("true presitive rate")
plt.ylim(0,1)
plt.xlim(0,1)
plt.plot([0, 1], [0, 1], '--', color=(0 , 0, 1))
plt.legend(loc=4)
plt.text(0.1,0.9,'AUC: %s' %auc_rf,fontdict={'size':'16','color':'b'})
plt.show()
```



```
#bagging
tree = DecisionTreeClassifier(criterion='entropy', max_depth=None)
bag = BaggingClassifier(base_estimator=tree, n_estimators=1000, max_samples=1.0, max_features=1.0, boo
bag.fit(train_x,train_y)
predict_bag = bag.predict(test_x)
accuracy_bag = accuracy_score(test_y, predict_bag)
tn_bag, fp_bag, fn_bag, tp_bag = confusion_matrix(test_y, predict_bag).ravel()
sensitivity_bag = tp_bag / (tp_bag+fn_bag)
print("Accuracy is %s" %accuracy_bag)
print("Sensitivity is %s" %sensitivity_bag)
print(confusion_matrix(test_y, predict_bag))
```

```
Accuracy is 0.9292285916489739
Sensitivity is 0.2882882882882883
[[1281   21]
 [  79   32]]
```

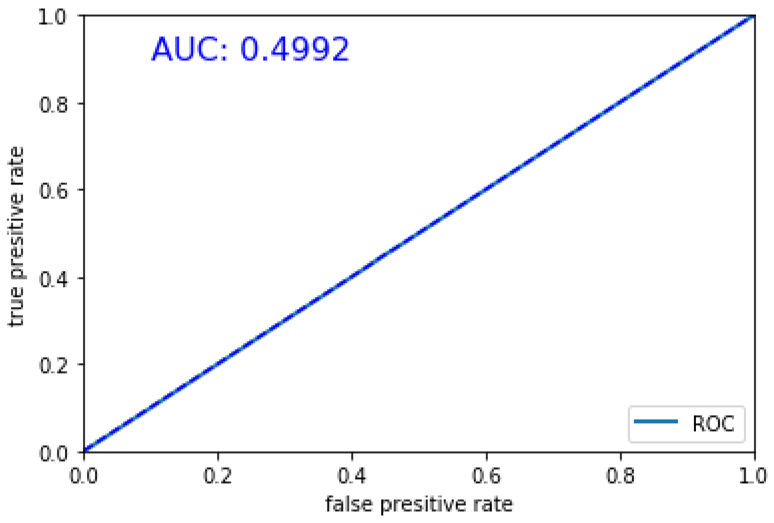
```
#bagging ROC
fpr_bag,tpr_bag,thresholds = roc_curve(test_y, predict_bag,pos_label=1)
auc_bag = round(auc(fpr_bag, tpr_bag),4)
plt.plot(fpr_bag,tpr_bag,linewidth=2,label="ROC")
plt.xlabel("false presitive rate")
plt.ylabel("true presitive rate")
plt.ylim(0,1)
plt.xlim(0,1)
plt.plot([0, 1], [0, 1], '--', color=(0 , 0, 1))
plt.legend(loc=4)
plt.text(0.1,0.9,'AUC: %s' %auc_bag,fontdict={'size':'16','color':'b'})
plt.show()
```



```
#svm
sv = svm.SVC(gamma='auto')
sv.fit(train_x,train_y)
predict_svm = sv.predict(test_x)
accuracy_svm = accuracy_score(test_y, predict_svm)
tn_svm, fp_svm, fn_svm, tp_svm = confusion_matrix(test_y, predict_svm).ravel()
sensitivity_svm = tp_svm / (tp_svm+fn_svm)
print("Accuracy is %s" %accuracy_svm)
print("Sensitivity is %s" %sensitivity_svm)
print(confusion_matrix(test_y, predict_svm))
```

```
Accuracy is 0.9200283085633404
Sensitivity is 0.0
[[1300    2]
 [ 111    0]]
```

```
#svm ROC
fpr_svm,tpr_svm,thresholds = roc_curve(test_y, predict_svm,pos_label=1)
auc_svm = round(auc(fpr_svm, tpr_svm),4)
plt.plot(fpr_svm,tpr_svm,linewidth=2,label="ROC")
plt.xlabel("false presitive rate")
plt.ylabel("true presitive rate")
plt.ylim(0,1)
plt.xlim(0,1)
plt.plot([0, 1], [0, 1], '--', color=(0 , 0, 1))
plt.legend(loc=4)
plt.text(0.1,0.9,'AUC: %s' %auc_svm,fontdict={'size':'16','color':'b'})
plt.show()
```



Github link: https://github.com/MMoRann/UTS_ML2019_13065040/tree/master

13065040 - Ran Mo