Compute performance metrics for the given Y and Y_score without sklearn

```
import numpy as np
import pandas as pd
# other than these two you should not import any other packages
from google.colab import files
files.upload()
```


Choose Files 4 files

- **5_a.csv**(application/vnd.ms-excel) - 241203 bytes, last modified: 6/19/2019 - 100% done
- **5_b.csv**(application/vnd.ms-excel) - 247322 bytes, last modified: 6/24/2019 - 100% done
- **5_c.csv**(application/vnd.ms-excel) - 63471 bytes, last modified: 6/19/2019 - 100% done
- **5_d.csv**(application/vnd.ms-excel) - 1742949 bytes, last modified: 6/19/2019 - 100% done

```
Saving 5_a.csv to 5_a.csv
Saving 5_b.csv to 5_b.csv
Saving 5_c.csv to 5_c.csv
Saving 5_d.csv to 5_d.csv
{'5_a.csv': b'y,proba\r\n1.0,0.6373866237658206\r\n1.0,0.6351650448158641\r\n1.0,0.76
 '5_b.csv': b'y,proba\r\n0.0,0.28103452586590194\r\n0.0,0.4651517681088171\r\n0.0,0.3
 '5_c.csv': b'y,prob\r\n0,0.4585206750276927\r\n0,0.5050369299746849\r\n0,0.41865173
 '5_d.csv': b'y,pred\r\n101.0,100.0\r\n120.0,100.0\r\n131.0,113.0\r\n164.0,125.0\r\n1
```


A. Compute performance metrics for the given data **5_a.csv**

Note 1: in this data you can see number of positive points >> number of negatives poi

Note 2: use pandas or numpy to read the data from **5_a.csv**

Note 3: you need to derive the class labels from given score

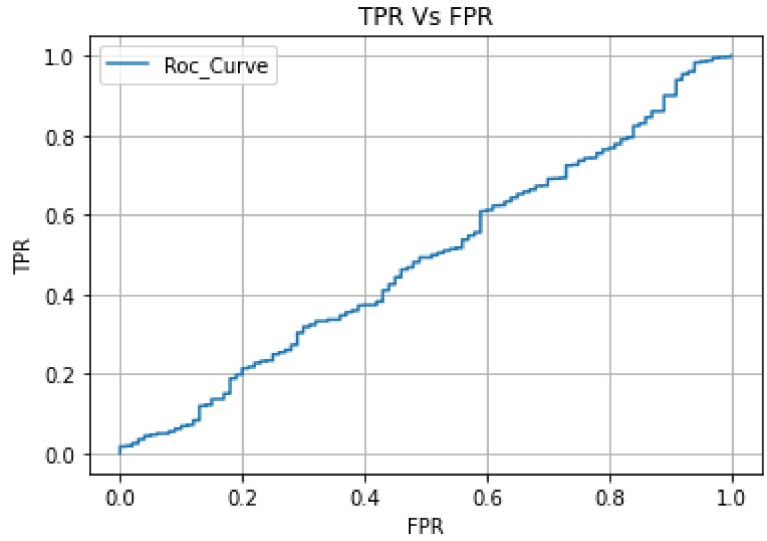
$$y^{pred} = [0 \text{ if y\_score} < 0.5 \text{ else } 1]$$

1. Compute Confusion Matrix
2. Compute F1 Score
3. Compute AUC Score, you need to compute different thresholds and for eacl
4. Compute Accuracy Score

```
from tqdm import tqdm
import matplotlib.pyplot as plt
# calculation of TP,FP,TN,FN and accuracy count
def calc_TP_FP_TN_FN(y_predict,y_actual):
  TP = 0
  TN = 0
  FP = 0
  FN = 0
  count = 0
  for j in range(len(y_predict)):
    if y_predict[j] == y_actual[j]:
      count+=1
      if y_predict[j] == 0:
        TN+=1
      else:
        TP+=1
    else:
      if y_predict[j] == 0:
        FN+=1
      else:
        FP+=1
  return TP,TN,FP,FN,count
# Reading CSV file
df = pd.read_csv("5_a.csv")
# Accesing columns in CSV file
y_score = df["y"]
y_proba = df["proba"]
#Calculation of confusion matrix and display
y_predict = np.zeros(y_proba.shape)
for i in range(len(y_predict)):
  if y_proba[i] < 0.5:
    y_predict[i] = 0
  else:
    y_predict[i] = 1
TP,TN,FP,FN,acc_count = calc_TP_FP_TN_FN(y_predict,y_score)
confuse_matrix = np.zeros((2,2))
confuse_matrix[0][0] = TN
confuse_matrix[0][1] = FN
confuse_matrix[1][0] = FP
confuse_matrix[1][1] = TP
print(confuse_matrix)
#Calculation of accuracy score
acc_score = acc_count/len(y_predict)
print("Accuracy score is: "+str(acc_score))
#Calculation of F1 score
precision = TP/(TP+FP)
recall = TP/(FN+TP)
F1_score = 2*precision*recall/(precision+recall)
print("F1 score is: "+str(F1_score))
#Calculation of AUC Value
sort_df = df.sort_values(by=["proba"], ascending=False)
y_score = sort_df["y"]
y_proba = sort_df["proba"]
```

```
desc_y_proba = np.array(y_proba)
y_score = np.array(y_score)
TPR_lst = []
FPR_lst = []
for k in tqdm(range(len(desc_y_proba))):
  temp_array = np.zeros(desc_y_proba.shape)
  #Based on the threshold value of Y_proba, manipulating the values(0 or 1) in sorted arra
  temp_array[:k+1] = 1
  temp_array[k+1:] = 0
  TP,TN,FP,FN,acc_count = calc_TP_FP_TN_FN(temp_array,y_score)
  TPR = TP/(FN+TP)
  FPR = FP/(TN+FP)
  TPR_lst.append(TPR)
  FPR_lst.append(FPR)
AUC_val = np.trapz(TPR_lst,FPR_lst)
print("AUC Score is: "+str(AUC_val))
# Plotting ROC Curve
plt.plot(FPR_lst,TPR_lst,label="Roc_Curve")
plt.xlabel("FPR")
plt.ylabel("TPR")
plt.legend()
plt.title("TPR Vs FPR")
plt.grid()
plt.show()
```

```
[[    0.     0.]
 [  100. 10000.]]
Accuracy score is: 0.9900990099009901
F1 score is: 0.9950248756218906
100%|██████████| 10100/10100 [01:35<00:00, 106.15it/s]
AUC Score is: 0.48829900000000004
```



B. Compute performance metrics for the given data **5_b.csv**

Note 1: in this data you can see number of positive points << number of negatives poi

Note 2: use pandas or numpy to read the data from **5_b.csv**

Note 3: you need to derive the class labels from given score

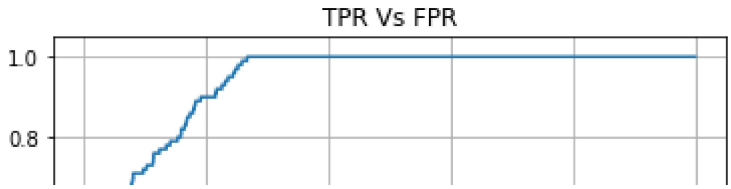
$$y^{pred} = [0 \text{ if } \text{y\_score} < 0.5 \text{ else } 1]$$

1. Compute Confusion Matrix
2. Compute F1 Score
3. Compute AUC Score, you need to compute different thresholds and for eacl
4. Compute Accuracy Score

```
from tqdm import tqdm
import matplotlib.pyplot as plt
# calculation of TP,FP,TN,FN and accuracy count
def calc_TP_FP_TN_FN(y_predict,y_actual):
  TP = 0
  TN = 0
  FP = 0
  FN = 0
  count = 0
  for j in range(len(y_predict)):
    if y_predict[j] == y_actual[j]:
      count+=1
      if y_predict[j] == 0:
        TN+=1
      else:
        TP+=1
    else:
      if y_predict[j] == 0:
        FN+=1
      else:
        FP+=1
  return TP,TN,FP,FN,count
# Reading CSV file
df = pd.read_csv("5_b.csv")
# Accesing columns in CSV file
y_score = df["y"]
y_proba = df["proba"]
#Calculation of confusion matrix and display
y_predict = np.zeros(y_proba.shape)
for i in range(len(y_predict)):
  if y_proba[i] < 0.5:
    y_predict[i] = 0
  else:
```

```
        y_predict[i] = 1
    TP,TN,FP,FN,acc_count = calc_TP_FP_TN_FN(y_predict,y_score)
    confuse_matrix = np.zeros((2,2))
    confuse_matrix[0][0] = TN
    confuse_matrix[0][1] = FN
    confuse_matrix[1][0] = FP
    confuse_matrix[1][1] = TP
    print(confuse_matrix)
    #Calculation of accuracy score
    acc_score = acc_count/len(y_predict)
    print("Accuracy score is: "+str(acc_score))
    #Calculation of F1 score
    precision = TP/(TP+FP)
    recall = TP/(FN+TP)
    F1_score = 2*precision*recall/(precision+recall)
    print("F1 score is: "+str(F1_score))
    #Calculation of AUC Value
    sort_df = df.sort_values(by=["proba"], ascending=False)
    y_score = sort_df["y"]
    y_proba = sort_df["proba"]
    desc_y_proba = np.array(y_proba)
    y_score = np.array(y_score)
    TPR_lst = []
    FPR_lst = []
    for k in tqdm(range(len(desc_y_proba))):
      temp_array = np.zeros(desc_y_proba.shape)
      #Based on the threshold value of Y_proba, manipulating the values(0 or 1) in sorted arra
      temp_array[:k+1] = 1
      temp_array[k+1:] = 0
      TP,TN,FP,FN,acc_count = calc_TP_FP_TN_FN(temp_array,y_score)
      TPR = TP/(FN+TP)
      FPR = FP/(TN+FP)
      TPR_lst.append(TPR)
      FPR_lst.append(FPR)
    AUC_val = np.trapz(TPR_lst,FPR_lst)
    print("AUC Score is: "+str(AUC_val))
    # Plotting ROC Curve
    plt.plot(FPR_lst,TPR_lst,label="Roc_Curve")
    plt.xlabel("FPR")
    plt.ylabel("TPR")
    plt.legend()
    plt.title("TPR Vs FPR")
    plt.grid()
    plt.show()
```

```
[[9761.   45.]
 [ 239.   55.]]
Accuracy score is: 0.9718811881188119
F1 score is: 0.2791878172588833
100%|██████████| 10100/10100 [01:33<00:00, 107.74it/s]
AUC Score is: 0.9377570000000001
```



C. Compute the best threshold (similarly to ROC curve computation) of probability which gives lowest values of metric **A** for the given data **5_c.csv**

you will be predicting label of a data points like this:

$y^{pred} = [0 \text{ if y\_score} < \text{threshold else } 1]$

$A = 500 \times \text{number of false negative} + 100 \times \text{numebr of false positive}$

Note 1: in this data you can see number of negative points > number of positive point

Note 2: use pandas or numpy to read the data from **5_c.csv**

```
from tqdm import tqdm
# calculation of TP,FP,TN,FN and accuracy count
def calc_TP_FP_TN_FN(y_predict,y_actual):
  TP = 0
  TN = 0
  FP = 0
  FN = 0
  count = 0
  for j in range(len(y_predict)):
    if y_predict[j] == y_actual[j]:
      count+=1
      if y_predict[j] == 0:
        TN+=1
      else:
        TP+=1
    else:
      if y_predict[j] == 0:
        FN+=1
      else:
        FP+=1
  return TP,TN,FP,FN,count
# Reading CSV file
df = pd.read_csv("5_c.csv")
# Accesing columns in CSV file
metrc_A_dict = {}
sort_df = df.sort_values(by=["prob"], ascending=False)
y_proba = np.array(sort_df["prob"])
y_score = np.array(sort_df["y"])
for i in tqdm(range(len(y_proba))):
```

```
    temp_array = np.zeros(y_proba.shape)
    # y_actual < y_threshold value resetting temp array to calculate TP,TN,FP,FN,acc_count
    temp_array[:i+1] = 1
    temp_array[i:] = 0
    TP,TN,FP,FN,acc_count = calc_TP_FP_TN_FN(temp_array,y_score)
    metrc_A = 500*FN+100*FP
    metrc_A_dict[y_proba[i]] = metrc_A
  # calculation of least metric A value
  least_metric_A = sorted(metrc_A_dict.items(),key=lambda x:x[1])[0][1]
  print("Least Metric_A Value: "+str(least_metric_A))
  Best_thersh_Proba = list(metrc_A_dict.keys())[list(metrc_A_dict.values()).index(least_metr
  print("Best threshold probability value for least Metric A is: "+str(Best_thersh_Proba))
```

```
100%|██████████| 2852/2852 [00:08<00:00, 345.72it/s]Least Metric_A Value: 141000
Best threshold probability value for least Metric A is: 0.22987164436159915
```

D. Compute performance metrics(for regression) for the given data **5_d.csv**

Note 2: use pandas or numpy to read the data from **5_d.csv**

Note 1: 5_d.csv will having two columns Y and predicted_Y both are real valued featu

1. Compute Mean Square Error
2. Compute MAPE: https://www.youtube.com/watch?v=ly6ztgIkUxk
3. Compute R^2 error: https://en.wikipedia.org/wiki/Coefficient_of_determi

```
from tqdm import tqdm
# Reading CSV file
df = pd.read_csv("5_d.csv")
# Accesing columns in CSV file
y_score = np.array(df["y"])
y_pred = np.array(df["pred"])
sqrd_err_mtrx = np.zeros(y_pred.shape)
abs_pr_err_mtrx = np.zeros(y_pred.shape)
sqrd_tot_err_mtrx = np.zeros(y_pred.shape)
#Calculation of square error, Total error and mean absolute percentage error
for i in tqdm(range(len(y_pred))):
  sq_err = (y_score[i]-y_pred[i])**2
  sqrd_err_mtrx[i] = sq_err
  abs_pr_err = abs((y_score[i]-y_pred[i])/np.mean(y_pred))
  abs_pr_err_mtrx[i] = abs_pr_err
  sq_tot_err = (y_score[i]-np.mean(y_pred))**2
  sqrd_tot_err_mtrx[i] = sq_tot_err
```

```
print("Mean Square Error is: "+str(np.mean(sqrd_err_mtrx)))
print("MAPE is: "+str(np.mean(abs_pr_err_mtrx)))
SS_residual = np.mean(sqrd_err_mtrx)
SS_total = np.mean(sqrd_tot_err_mtrx)
Rsqr_err = 1-(SS_residual/SS_total)
print("R^2 error is: "+str(Rsqr_err))
```

```
100%|██████████| 157200/157200 [00:24<00:00, 6388.12it/s]Mean Square Error is: 177.16
MAPE is: 0.12927250737711504
R^2 error is: 0.9563583447288622
```

✓ 8s completed at 5:43 AM