Float_Non_Float-ROC

Bishnu Poudel

April 1, 2020

1 ROC curve

```
[125]: import numpy as np
       import pandas as pd
       import matplotlib.pyplot as plt
```

```
[126]: df = pd.read_csv("glassTrain.csv",  index_col=0)
       df_test = pd.read_csv("glassTest.csv",  index_col=0)
```

1.1 Preprocessing, dropping columns and rows

```
[127]: mlx_df = df[ ['RI', 'Na', 'Mg', 'Si', 'K', 'Ba', 'Fe', 'type'] ]
       mlx_df.type.value_counts()
```

```
[127]: 2    51
       1    47
       7    19
       3    11
       5     9
       6     6
       Name: type, dtype: int64
```

```
[128]: roc_df = mlx_df.drop(mlx_df[mlx_df.type.isin([5,6,7])].index)
       roc_df[roc_df.type == 3] = 1
       roc_df[roc_df.type == 2] = 0
```

```
[129]: roc_df.type.value_counts()
```

```
[129]: 1    58
       0    51
       Name: type, dtype: int64
```

```
[130]: X = roc_df.iloc[:, :-1 ].values
       y = roc_df.iloc[:, -1: ].values.flatten()
```

```
[131]: from sklearn.preprocessing import LabelEncoder
       le = LabelEncoder()
```

```
y = le.fit_transform(y)
le.classes_
```

[131]:
```
array([0, 1], dtype=int64)
```

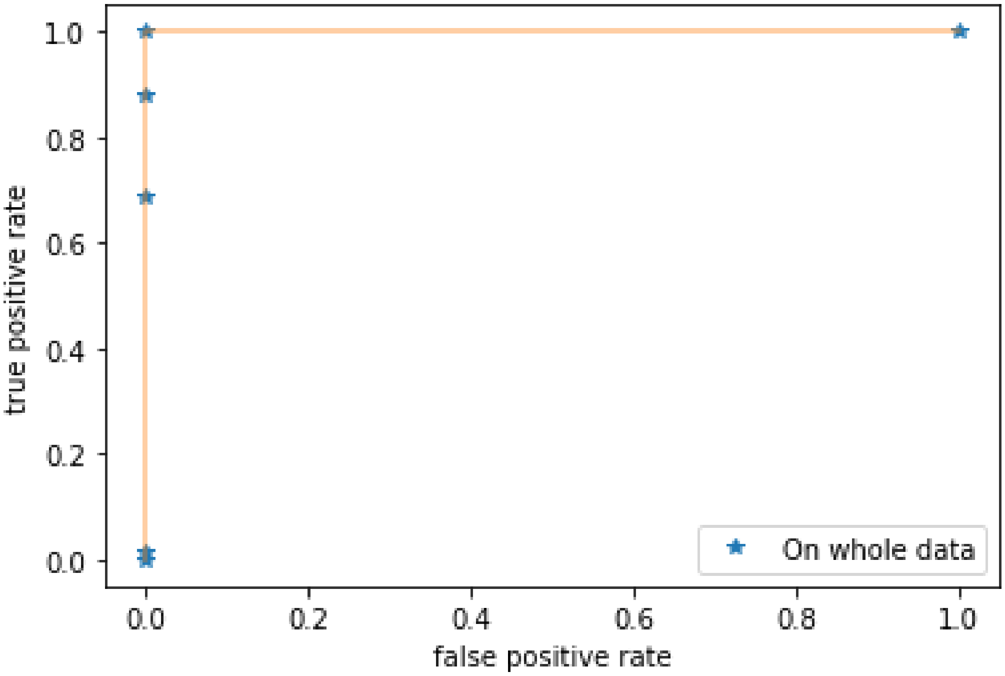
[132]:
```
from sklearn.metrics import roc_curve, auc
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import StratifiedKFold
from scipy import interp
from sklearn.ensemble import RandomForestClassifier
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
```

[133]:
```
# rf=RandomForestClassifier( n_estimators= 10
#                              , max_depth= 1
#                              , n_jobs=-1)
rf = make_pipeline( StandardScaler()
                   ,PCA(n_components=7)
                   , LogisticRegression(solver='lbfgs') )
```

[134]:
```
from sklearn.model_selection import StratifiedKFold
from scipy import interp

probas = rf.fit(X,y).predict_proba(X)

# print(probas)

fpr, tpr, thresholds = roc_curve(y,
                                 probas[:, 1],
                                 pos_label= 1)
print(fpr)
print(tpr)

plt.plot(fpr, tpr,'*', label='On whole data' )
plt.plot(fpr, tpr,'-',alpha=0.5)
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.legend(loc="lower right")
```

```
[0. 0. 0. 0. 0. 1.]
[0.         0.01724138 0.68965517 0.87931034 1.         1.        ]
```

[134]:
```
<matplotlib.legend.Legend at 0x1bf424978c8>
```

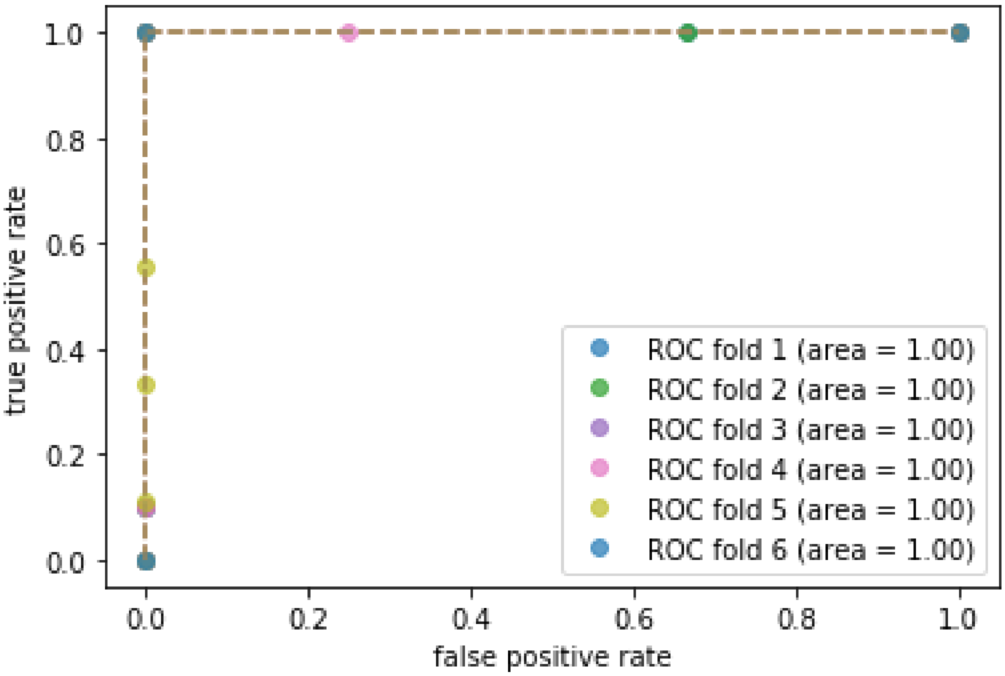


1.2 ROC Curve for various cv

[135]:

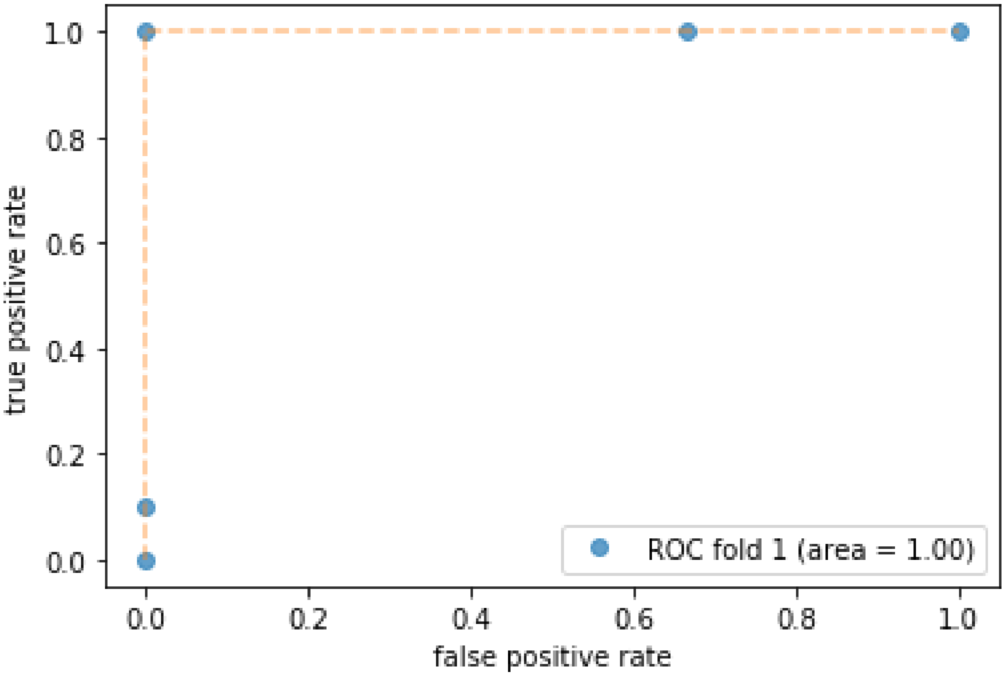
```
# Cross-validation specification
cv = list(StratifiedKFold(n_splits=6,   random_state=1).split(X, y))

for i, (train, test) in enumerate(cv):
    probas = rf.fit(X[train],
                         y[train]).predict_proba(X[test])

    # False Positive and True Positive Rates (thresholds for the decision␣
 ↪function)
    fpr, tpr, thresholds = roc_curve(y[test],
                                     probas[:, 1],
                                     pos_label=1)
#   print('fpr', fpr)
#   print('tpr', tpr)
    # Add to mean True Predictive Rate in a smoothed variant (interpolated)
    mean_tpr += interp(mean_fpr, fpr, tpr)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, 'o', alpha=0.7 ,  label='ROC fold %d (area = %0.2f)'
                  % (i+1, roc_auc))
    plt.plot(fpr, tpr,'--', alpha=0.5 )
plt.xlabel('false positive rate')
plt.ylabel('true positive rate')
plt.legend(loc="lower right")
```

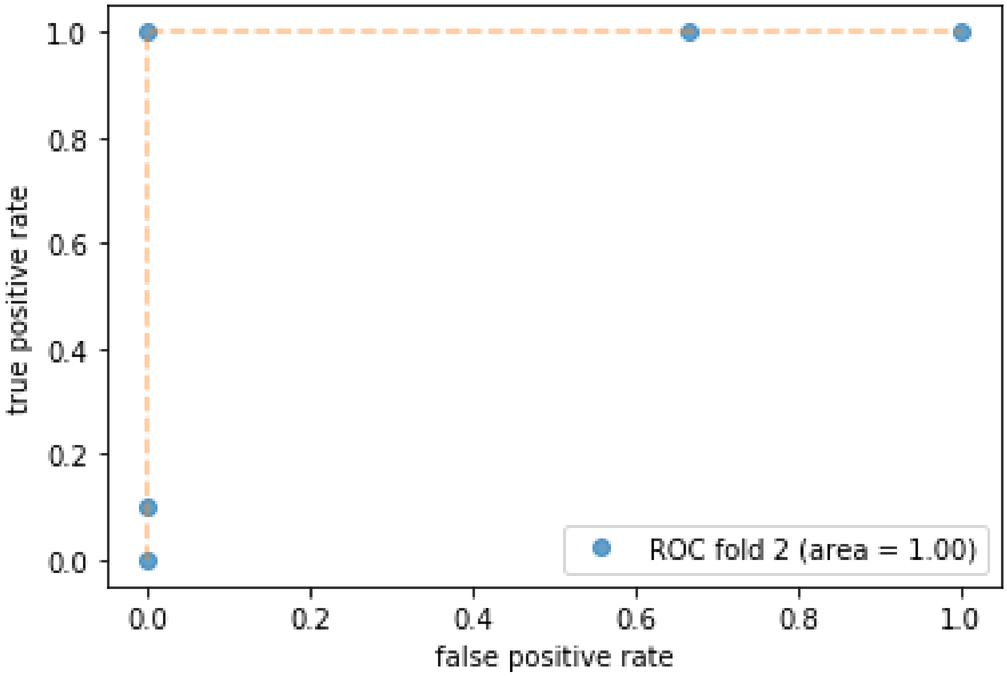
```
[135]: <matplotlib.legend.Legend at 0x1bf424e6608>
```

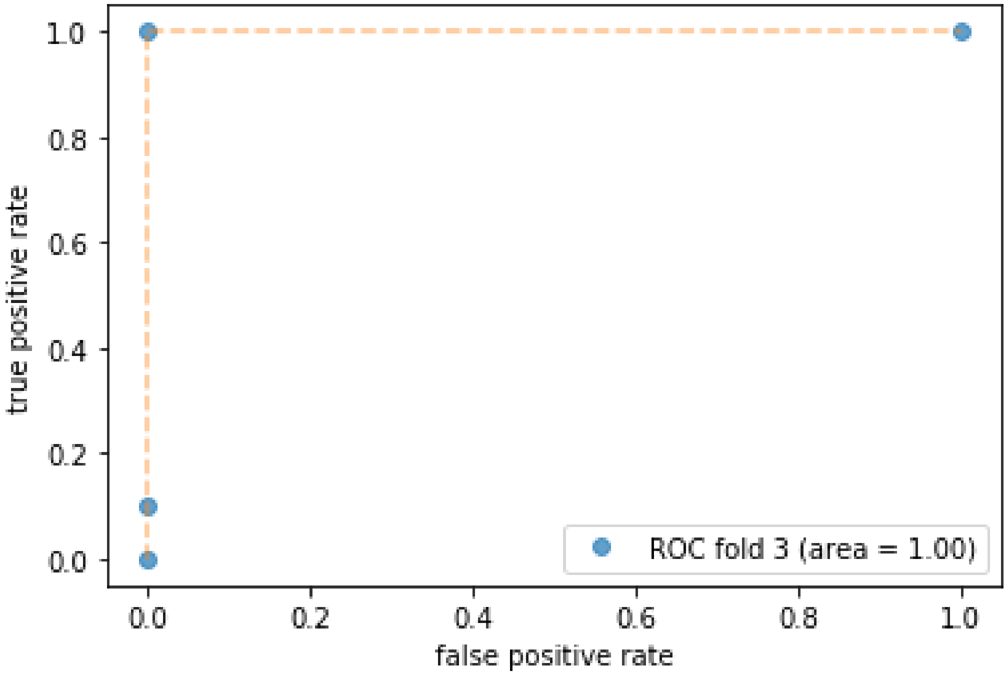


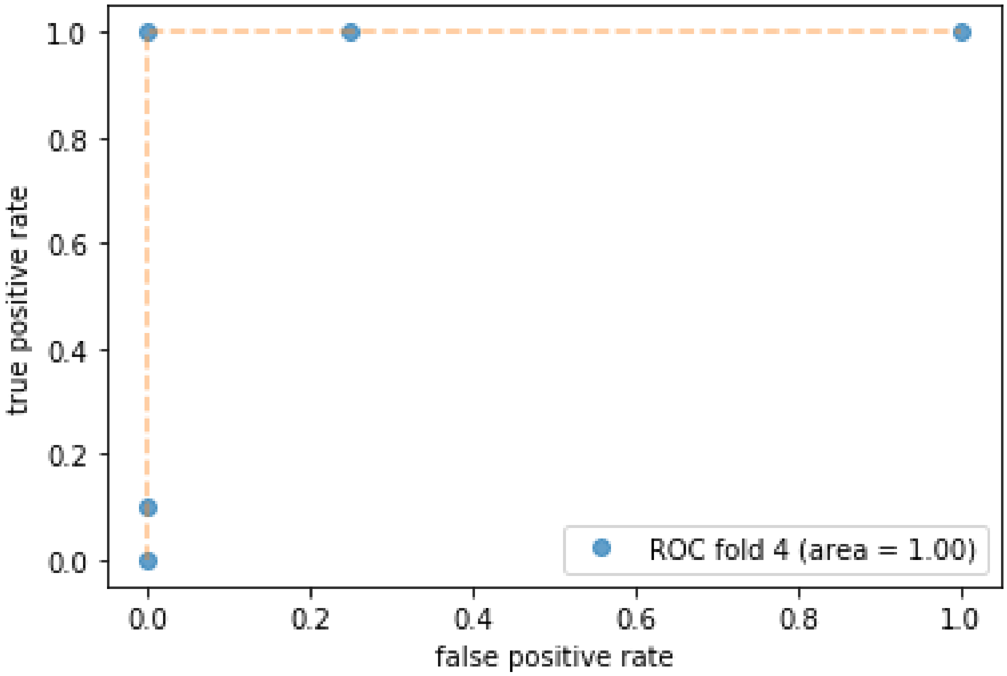
1.3 Plotting individual curves as they all overlapped above

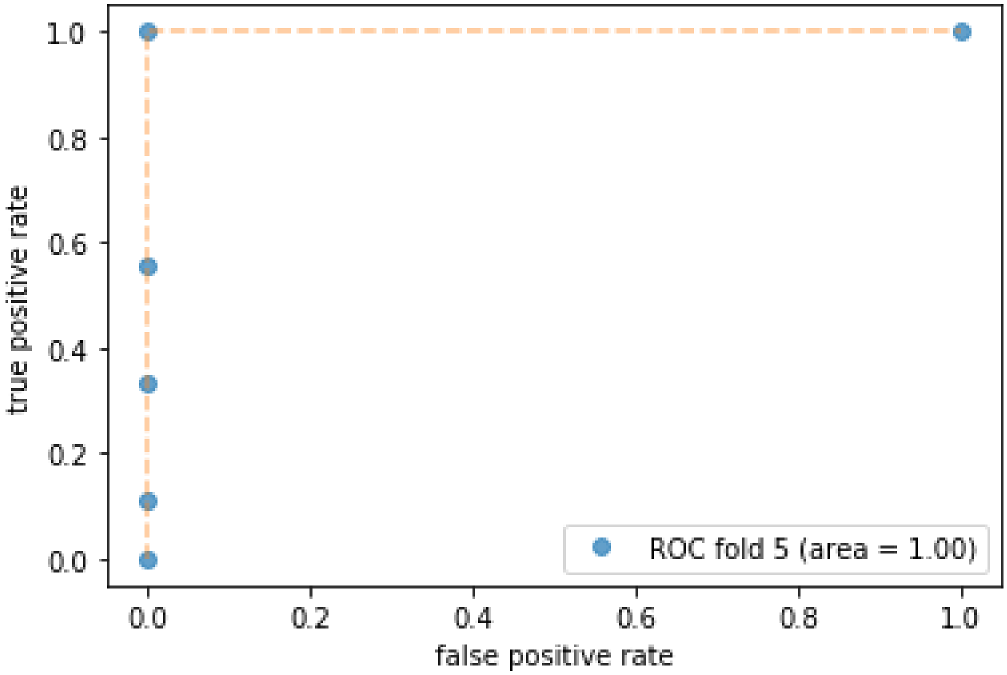
```
[137]: # Cross-validation specification
cv = list(StratifiedKFold(n_splits=6,   random_state=1).split(X, y))

for i, (train, test) in enumerate(cv):
    plt.figure()
    probas = rf.fit(X[train],
                         y[train]).predict_proba(X[test])

    # False Positive and True Positive Rates (thresholds for the decision␣
 ↪function)
    fpr, tpr, thresholds = roc_curve(y[test],
                                     probas[:, 1],
                                     pos_label=1)
#   print('fpr', fpr)
#   print('tpr', tpr)
    # Add to mean True Predictive Rate in a smoothed variant (interpolated)
    mean_tpr += interp(mean_fpr, fpr, tpr)
    roc_auc = auc(fpr, tpr)

    plt.plot(fpr, tpr, 'o', alpha=0.7 ,  label='ROC fold %d (area = %0.2f)'
                   % (i+1, roc_auc))
    plt.plot(fpr, tpr,'--', alpha=0.5 )

    plt.xlabel('false positive rate')
    plt.ylabel('true positive rate')
    plt.legend(loc="lower right")
    plt.show
```

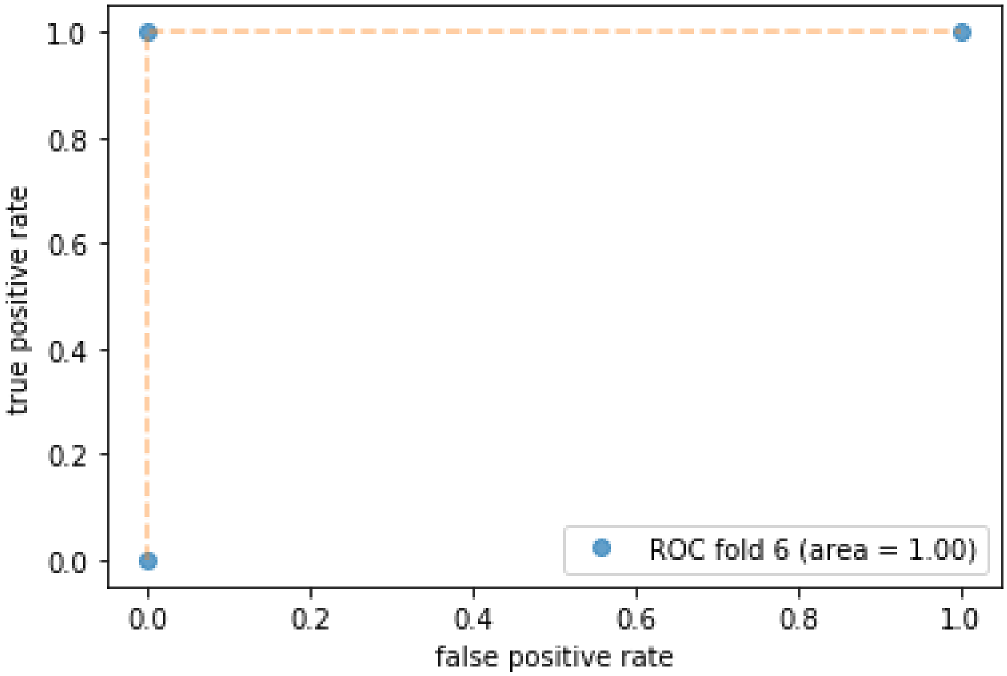



1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
● ROC fold 2 (area = 1.00)


1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
● ROC fold 3 (area = 1.00)


1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
● ROC fold 4 (area = 1.00)


1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
● ROC fold 5 (area = 1.00)


1.0
0.8
0.6
0.4
0.2
0.0
true positive rate
0.0
0.2
0.4
0.6
0.8
1.0
false positive rate
ROC fold 6 (area = 1.00)