Lab10

October 3, 2020

[32]:
```
import pandas as p
import numpy as n
import seaborn as sns

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from sklearn.preprocessing import LabelEncoder                          #encoding
from sklearn.preprocessing import StandardScaler                       ␣
 ↪#standardisation
from sklearn.model_selection import train_test_split                    #train/test␣
 ↪split
from sklearn.model_selection import cross_val_score                     #K-fold␣
 ↪cross validation

#SVM libraries
from sklearn.svm import SVC
from sklearn import metrics
from sklearn.model_selection import KFold
from sklearn.model_selection import GridSearchCV                        #to find␣
 ↪best parameter

import matplotlib.pyplot as plt

%matplotlib inline
```

[2]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[3]:
```
# Dataframe from CSV File

dataframe = p.read_csv('/content/drive/My Drive/voice.csv')
dataframe.head()
```

[3]:
```
   meanfreq        sd    median       Q25  ...    maxdom   dfrange   modindx
label
0  0.059781  0.064241  0.032027  0.015071  ...  0.007812  0.000000  0.000000
```

```
male
1  0.066009  0.067310  0.040229  0.019414  ...  0.054688  0.046875  0.052632
male
2  0.077316  0.083829  0.036718  0.008701  ...  0.015625  0.007812  0.046512
male
3  0.151228  0.072111  0.158011  0.096582  ...  0.562500  0.554688  0.247119
male
4  0.135120  0.079146  0.124656  0.078720  ...  5.484375  5.476562  0.208274
male

[5 rows x 21 columns]
```

[4]:

```
dataframe.shape
```

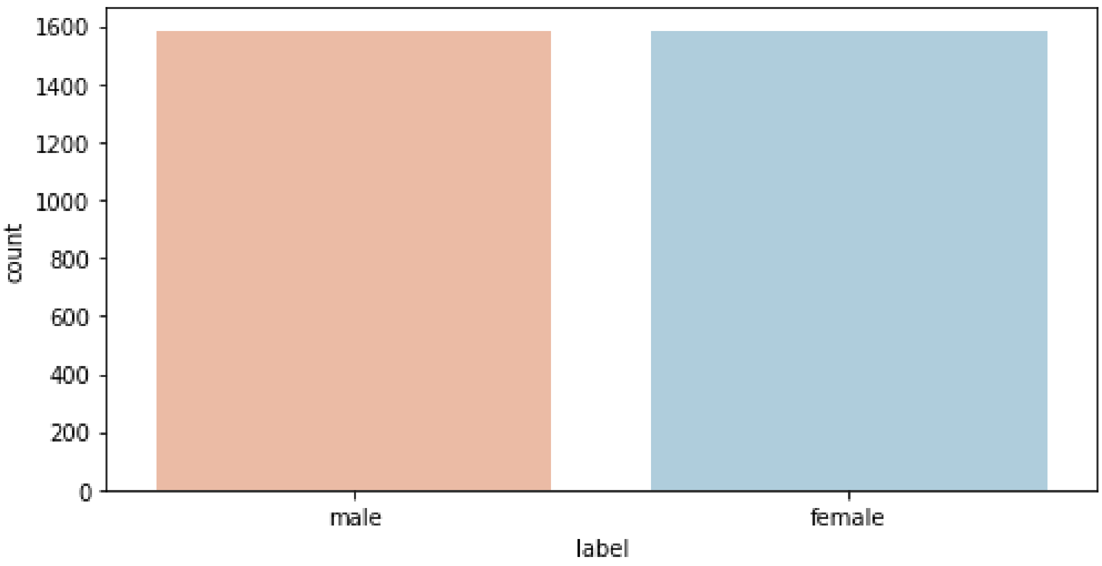
[4]:

```
(3168, 21)
```

[5]:

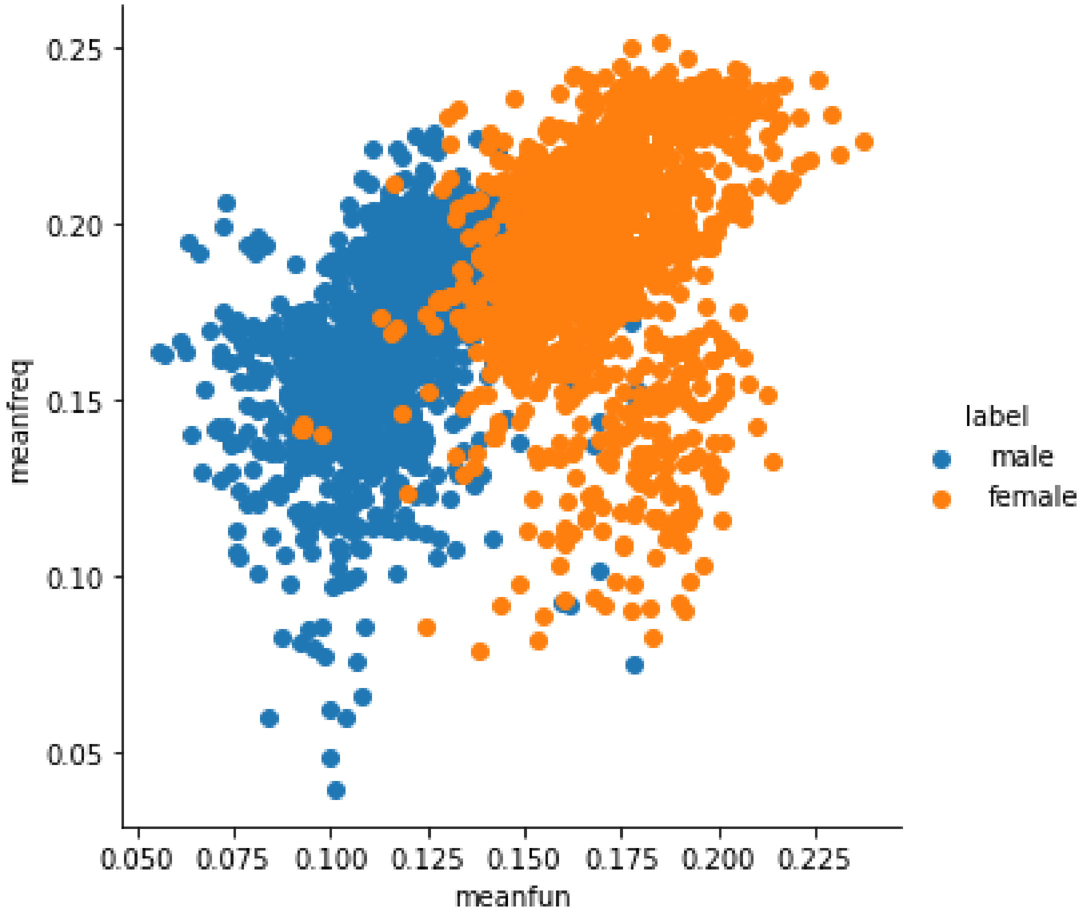
```
# visualize distribution of classes

m.figure(figsize=(8, 4))
sns.countplot(dataframe['label'], palette='RdBu')

# count number of observations in each class
male, female = dataframe['label'].value_counts()
print('Number of cells labeled Male    : ', male)
print('Number of cells labeled Female  : ', female)
print('')
print('% of Voices labeled Male           : ', round(male / len(dataframe) *␣
 ↪100, 2), '%')
print('% of Voices labeled Female         : ', round(female / len(dataframe) *␣
 ↪100, 2), '%')
```

```
Number of cells labeled Male    :  1584
Number of cells labeled Female  :  1584

% of Voices labeled Male           :  50.0 %
% of Voices labeled Female         :  50.0 %

/usr/local/lib/python3.6/dist-packages/seaborn/_decorators.py:43: FutureWarning:
Pass the following variable as a keyword arg: x. From version 0.12, the only
valid positional argument will be `data`, and passing other arguments without an
explicit keyword will result in an error or misinterpretation.
  FutureWarning
```



[6]:

```
sns.FacetGrid(dataframe, hue="label", height=5).map(m.scatter, "meanfun", "meanfreq").add_legend()
m.show()
```



[7]:
```
X = dataframe.iloc[:, :-1]
y = dataframe.iloc[:, -1]
encode = LabelEncoder()
y = encode.fit_transform(y)
y
print('Male Label Encoded as ------> 1')
print('Female Label Encoded as ----> 0')
```

```
Male Label Encoded as ------> 1
Female Label Encoded as ----> 0
```

[8]:
```
scale = StandardScaler()
scale.fit(X)
X = scale.transform(X)
```

[10]:
```
#Train/Test Split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,␣
 ↪random_state = 1)
```

[11]:

```
from sklearn.ensemble import RandomForestClassifier

# Create the model with 100 trees
model = RandomForestClassifier(n_estimators=100,
                               bootstrap = True,
                               max_features = 'sqrt')
# Fit on training data
model.fit(X_train, y_train)
```

[11]:

```
RandomForestClassifier(bootstrap=True, ccp_alpha=0.0, class_weight=None,
                       criterion='gini', max_depth=None, max_features='sqrt',
                       max_leaf_nodes=None, max_samples=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, n_estimators=100,
                       n_jobs=None, oob_score=False, random_state=None,
                       verbose=0, warm_start=False)
```

[13]:

```
# Actual class predictions
rf_predictions = model.predict(X_test)
# Probabilities for each class
rf_probs = model.predict_proba(X_test)[:, 1]
```

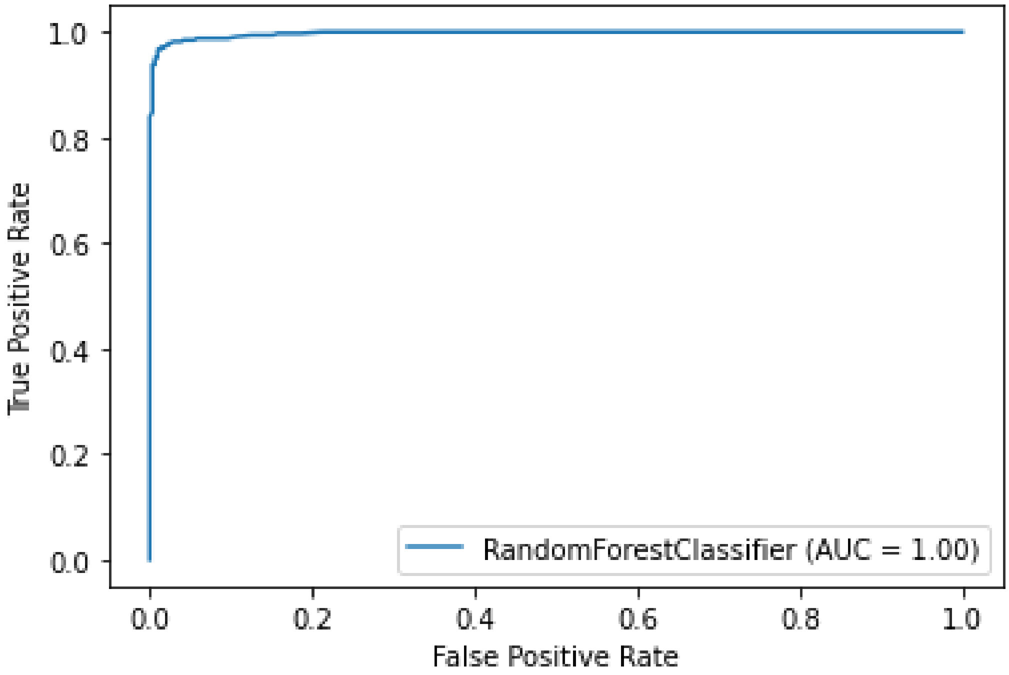
[16]:

```
from sklearn.metrics import roc_auc_score

# Calculate roc auc
roc_value = roc_auc_score(y_test, rf_probs)
print(roc_value)
```

```
0.9970931679491901
```

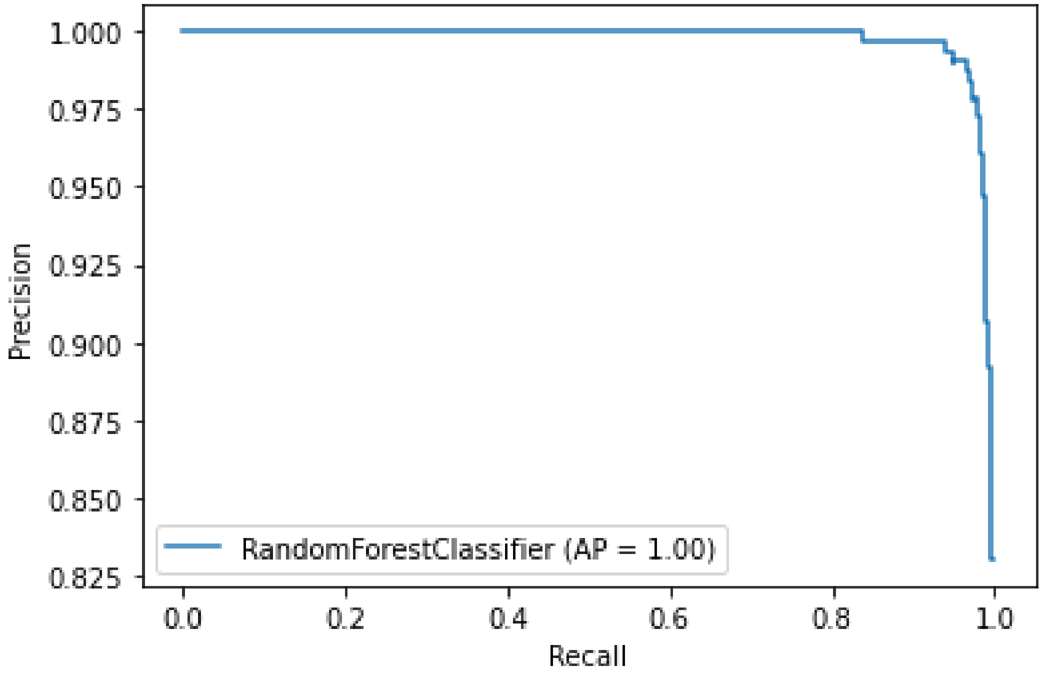
[33]:

```
from sklearn.metrics import plot_roc_curve
plot_roc_curve(model,X_test,y_test)
plt.show()
```



[36]:
```python
from sklearn.metrics import plot_precision_recall_curve
plot_precision_recall_curve(model,X_test,y_test)
```

[36]:
```
<sklearn.metrics._plot.precision_recall_curve.PrecisionRecallDisplay at
0x7ff792b160b8>
```

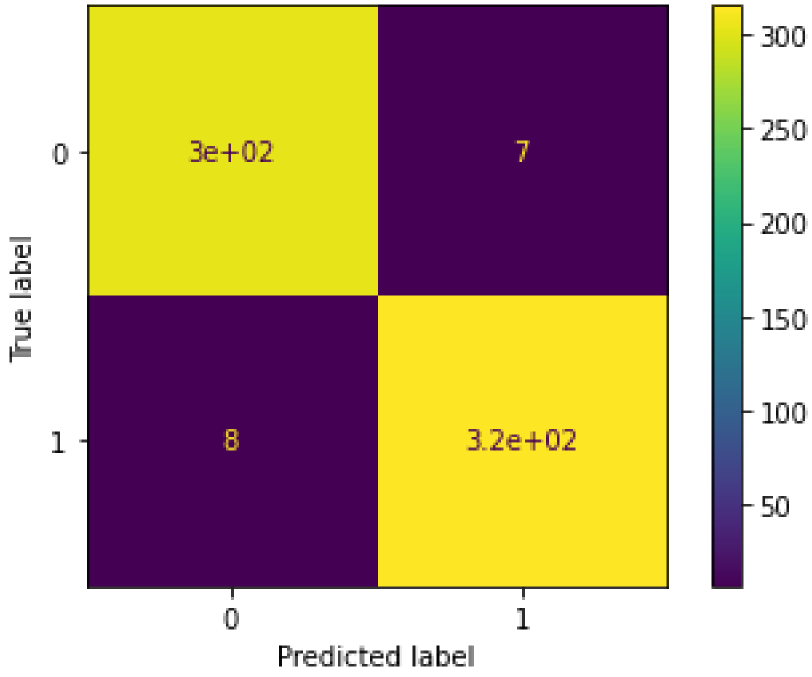


[39]:
```
print(classification_report(y_test,rf_predictions))
```

```
              precision    recall  f1-score   support

           0       0.97      0.98      0.98       311
           1       0.98      0.98      0.98       323

    accuracy                           0.98       634
   macro avg       0.98      0.98      0.98       634
weighted avg       0.98      0.98      0.98       634
```

[35]:
```
from sklearn.metrics import plot_confusion_matrix
plot_confusion_matrix(model,X_test,y_test)
```

[35]:
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at
0x7ff792aeb400>
```



[20]:
```
import pandas as pd

# Extract feature importances
fi = pd.DataFrame({'feature': list(dataframe.iloc[:,:-1].columns),
                   'importance': model.feature_importances_}).\
                    sort_values('importance', ascending = False)

# Display
fi.head()
```

[20]:

```
    feature  importance
12  meanfun    0.332216
3       Q25    0.196054
5       IQR    0.181599
1        sd    0.061186
9       sfm    0.036848
```