```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import sklearn.model_selection, sklearn.linear_model, sklearn.svm, sklearn.metrics
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.svm import SVC
```

```
dataset = pd.read_csv('voice.csv')
```

```
dataset.head()
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	kurt	sp.ent	sfm
0	0.059781	0.064241	0.032027	0.015071	0.090193	0.075122	12.863462	274.402906	0.893369	0.491918
1	0.066009	0.067310	0.040229	0.019414	0.092666	0.073252	22.423285	634.613855	0.892193	0.513724
2	0.077316	0.083829	0.036718	0.008701	0.131908	0.123207	30.757155	1024.927705	0.846389	0.478905
3	0.151228	0.072111	0.158011	0.096582	0.207955	0.111374	1.232831	4.177296	0.963322	0.727232
4	0.135120	0.079146	0.124656	0.078720	0.206045	0.127325	1.101174	4.333713	0.971955	0.783568

5 rows × 21 columns

```
dataset.describe()
```

	meanfreq	sd	median	Q25	Q75	IQR	skew	k
count	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000000	3168.000
mean	0.180907	0.057126	0.185621	0.140456	0.224765	0.084309	3.140168	36.568
std	0.029918	0.016652	0.036360	0.048680	0.023639	0.042783	4.240529	134.928
min	0.039363	0.018363	0.010975	0.000229	0.042946	0.014558	0.141735	2.068
25%	0.163662	0.041954	0.169593	0.111087	0.208747	0.042560	1.649569	5.669
50%	0.184838	0.059155	0.190032	0.140286	0.225684	0.094280	2.197101	8.318
75%	0.199146	0.067020	0.210618	0.175939	0.243660	0.114175	2.931694	13.648
max	0.251124	0.115273	0.261224	0.247347	0.273469	0.252225	34.725453	1309.612

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3168 entries, 0 to 3167
Data columns (total 21 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   meanfreq  3168 non-null   float64
 1   sd        3168 non-null   float64
 2   median    3168 non-null   float64
 3   Q25       3168 non-null   float64
 4   Q75       3168 non-null   float64
 5   IQR       3168 non-null   float64
 6   skew      3168 non-null   float64
 7   kurt      3168 non-null   float64
 8   sp.ent    3168 non-null   float64
 9   sfm       3168 non-null   float64
 10  mode      3168 non-null   float64
 11  centroid  3168 non-null   float64
 12  meanfun   3168 non-null   float64
 13  minfun    3168 non-null   float64
 14  maxfun    3168 non-null   float64
 15  meandom   3168 non-null   float64
 16  mindom    3168 non-null   float64
 17  maxdom    3168 non-null   float64
 18  dfrange   3168 non-null   float64
 19  modindx   3168 non-null   float64
```

```
 20  label     3168 non-null   object
dtypes: float64(20), object(1)
memory usage: 519.9+ KB
```

```
dataset.isnull().sum()
```

```
meanfreq    0
sd          0
median      0
Q25         0
Q75         0
IQR         0
skew        0
kurt        0
sp.ent      0
sfm         0
mode        0
centroid    0
meanfun     0
minfun      0
maxfun      0
meandom     0
mindom      0
maxdom      0
dfrange     0
modindx     0
label       0
dtype: int64
```

```
dataset.replace(to_replace="male", value=1, inplace=True)
dataset.replace(to_replace="female", value=0, inplace=True)
dataset.label.unique()
```

```
array([1, 0])
```

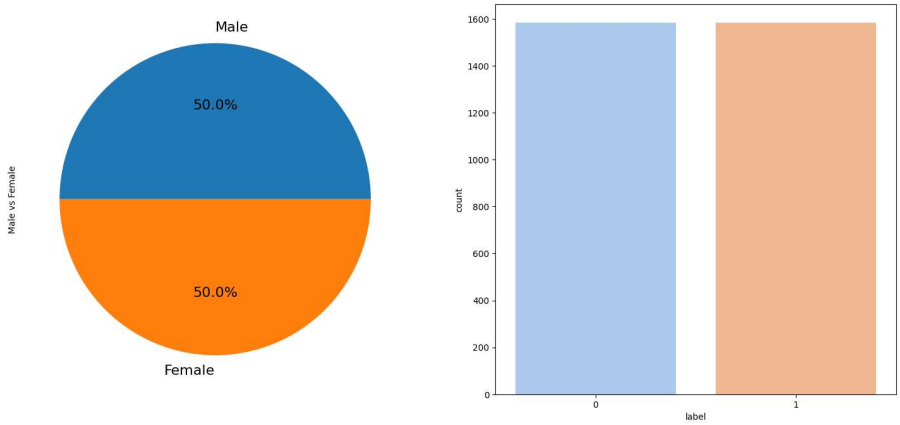
```
xData=dataset.iloc[:,:-1]
yData=dataset.iloc[:,-1]
xData.shape, yData.shape
```

```
((3168, 20), (3168,))
```

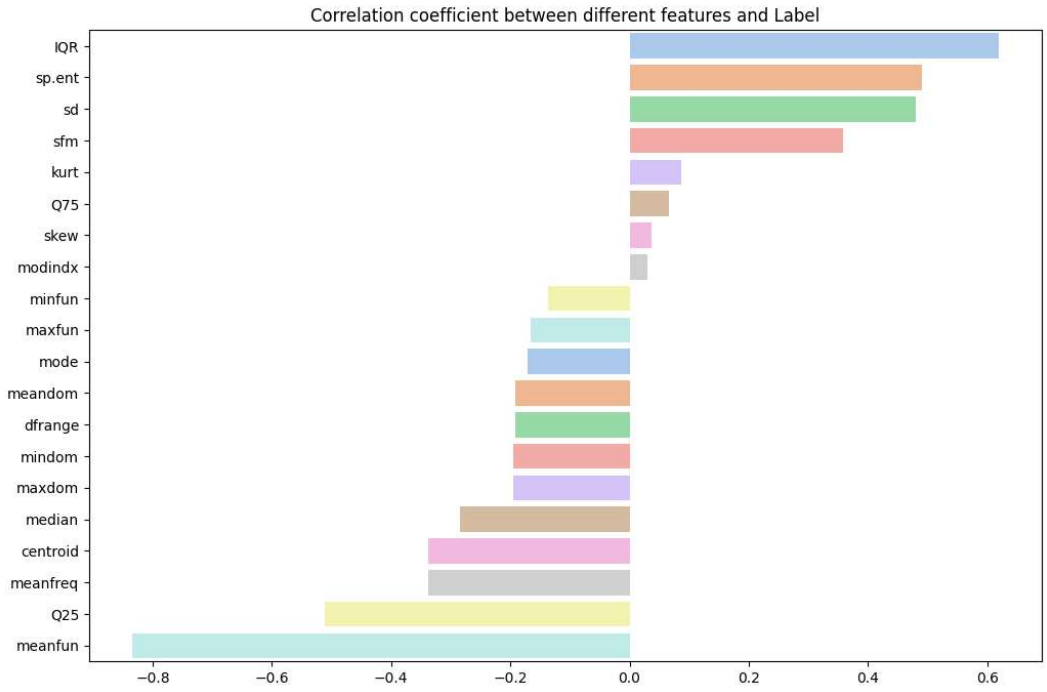
```
TRAINSPLIT = 0.8

xTrain, xTest, yTrain, yTest = sklearn.model_selection.train_test_split(xData, yData, train_size=TRAINSPLIT)
xTrain.shape, yTrain.shape
```

```
((2534, 20), (2534,))
```

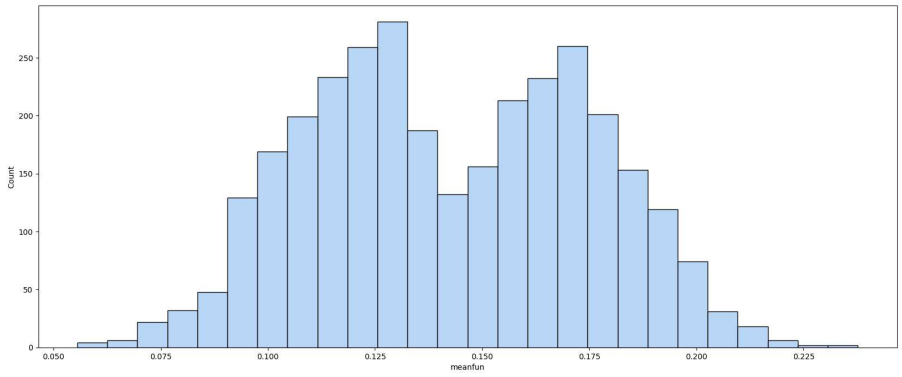
```
plt.figure(figsize=(18, 8))
plt.subplot(1, 2, 1)
dataset.label.value_counts().plot(kind="pie",
                                  fontsize=16,
                                  labels=["Male", "Female"],
                                  ylabel="Male vs Female",
                                  autopct='%1.1f%%');

plt.subplot(1, 2, 2)
sns.countplot(x="label",data=dataset, palette="pastel")
plt.show()
```




```
plt.figure(figsize=(12,8))
data = dataset.corr()["label"].sort_values(ascending=False)
indices = data.index
labels = []
corr = []
for i in range(1, len(indices)):
    labels.append(indices[i])
    corr.append(data[i])
sns.barplot(x=corr, y=labels, palette='pastel')
plt.title('Correlation coefficient between different features and Label')
plt.show()
```




```
plt.figure(figsize=(20,8))
sns.histplot(dataset.meanfun, color=sns.color_palette('pastel')[0])
plt.show()
```




```
regressionModel = LogisticRegression(solver='liblinear')
regressionModel.fit(xTrain,yTrain)
regressionModel.score(xTrain,yTrain)
```

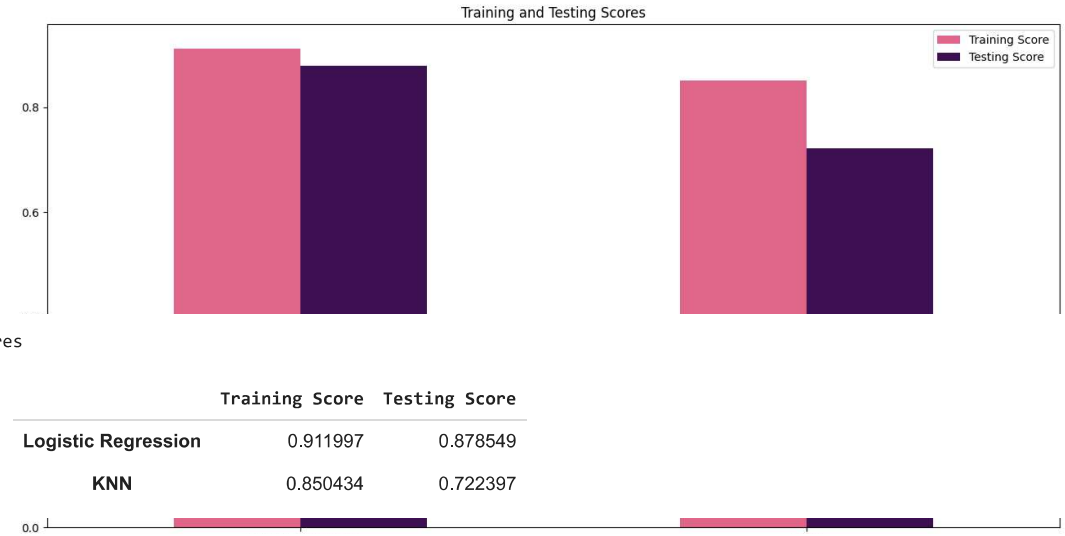
```
0.9119968429360694
```

```
KNNModel = KNeighborsClassifier(n_neighbors=3)
KNNModel.fit(xTrain,yTrain)
KNNModel.score(xTrain,yTrain)
```

```
0.8504340962904499
```

```
indices = ['Logistic Regression', 'KNN']  # Remove 'RandomForest' from indices
trainScores = [regressionModel.score(xTrain, yTrain), KNNModel.score(xTrain, yTrain)]
testScores = [regressionModel.score(xTest, yTest), KNNModel.score(xTest, yTest)]
scores = pd.DataFrame({'Training Score': trainScores, 'Testing Score': testScores}, index=indices)
```

```
plot = scores.plot.bar(figsize=(16, 8), rot=0, color=['#df6589ff', '#3c1053ff'])
plt.title('Training and Testing Scores')
plt.show()
```

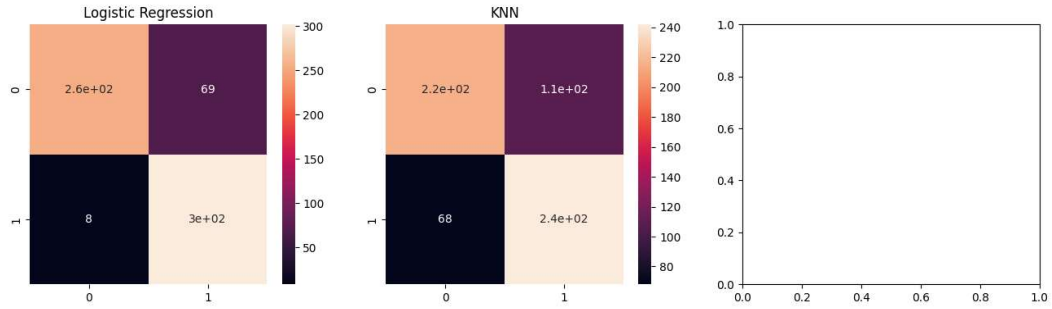


scores

	Training Score	Testing Score
Logistic Regression	0.911997	0.878549
KNN	0.850434	0.722397

```
predRegression = regressionModel.predict(xTest)
predKNN = KNNModel.predict(xTest)
predVals = pd.DataFrame(data={'truth': yTest, 'regression': predRegression, 'knn': predKNN})


plt.figure(figsize=(16, 14))
plt.subplot(3, 3, 1)
sns.heatmap(sklearn.metrics.confusion_matrix(yTest, predRegression), annot=True).set(title='Logistic Regression')
plt.subplot(3, 3, 2)
sns.heatmap(sklearn.metrics.confusion_matrix(yTest, predKNN), annot=True).set(title='KNN')
plt.subplot(3, 3, 3)
```

```
<Axes: >
```




```
print("Logistic Regression:\n\n", sklearn.metrics.classification_report(yTest, predRegression))
```

```
Logistic Regression:

              precision    recall  f1-score   support

           0       0.97      0.79      0.87       324
           1       0.81      0.97      0.89       310

    accuracy                           0.88       634
   macro avg       0.89      0.88      0.88       634
weighted avg       0.89      0.88      0.88       634
```

```
print("KNN:\n\n", sklearn.metrics.classification_report(yTest, predKNN))
```

```
KNN:

              precision    recall  f1-score   support

           0       0.76      0.67      0.71       324
```

```
                1       0.69      0.78      0.73       310

         accuracy                           0.72       634
        macro avg       0.73      0.72      0.72       634
     weighted avg       0.73      0.72      0.72       634
```

```
predVals.head()
```

	truth	regression	knn
2599	0	0	0
487	1	1	1
2543	0	0	0
3098	0	0	0
1928	0	1	1