实验二：男声女声分类

学号：2018329621262　　姓名：刘恒玮

一、实验目的

理解 SVM 模型，使用 SVM 模型解决实际问题。

二、实验环境

PC 机，Python，pandas 库，sklearn 库。

三、实验内容

给定训练数据，利用 SVM 模型判断测试数据男声还是女声。

四、实验原理（算法）

SVM 算法原理：

SVM 又称为支持向量机，是一种二分类的模型。当然如果进行修改之后也是可以用于多类别问题的分类。支持向量机可以分为线性核非线性两大类。其主要思想为找到空间中的一个更够将所有数据样本划开的超平面，并且使得本本集中所有数据到这个超平面的距离最短。SVM 学习的基本想法是求解能够正确划分训练数据集并且几何间隔最大的分离超平面。$\omega x+b=0$ 即为分离超平面，对于线性可分的数据集来说，这样的超平面有无穷多个（即感知机），但是几何间隔最大的分离超平面却是唯一的。

决策树算法原理：

决策树是通过一系列规则对数据进行分类的过程。实际上，样本所有特征中有一些特征在分类时起到决定性作用，决策树的构造过程就是找到这些具有决定性作用的特征，根据其决定性程度来构造一个倒立的树，决定性作用最大的那个特征作为根节点，然后递归找到各分支下子数据集中次大的决定性特征，直至子数据集中所有数据都属于同一类。

Logistics 回归算法原理：logistic 回归是一种广义线性回归（generalized linear model），又叫对数几率回归，因此与多重线性回归分析有很多相同之处。Logistic 回归通过使用其固有的 logistic 函数估计概率，来衡量因变量（我们想要预测的标签）与一个或多个自变量（特征）之间的关系。比如说可以进行二值化进行预测，也就是 Sigmoid 函数。Sigmoid 函数是一个 S 形曲线，它可以将任意实数值映射到介于 0 和 1 之间的值，但并不会取到 0/1。然后使用阈值分类器将 0 和 1 之间的值转换为 0 或 1。logistic 回归的因变量可以是二分类的，也可以是多分类的，但是二分类的更为常用，也更加容易解释。

五、实验步骤（分析过程）

1. 导入相关的库，并利用 pandas 库读取训练集和测试集的数据。

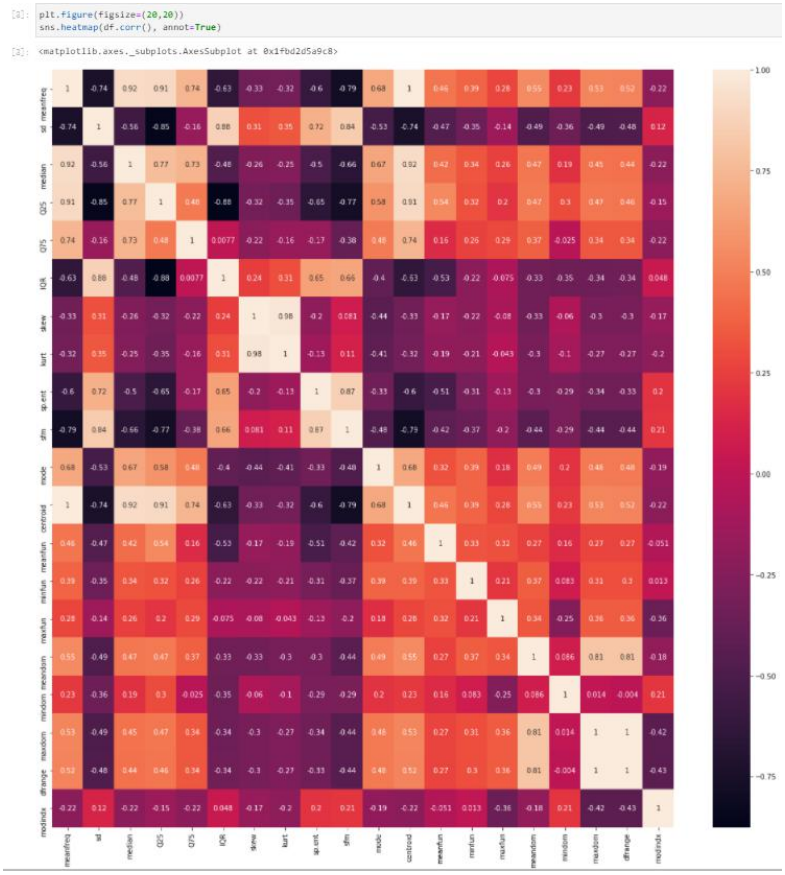
[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import cross_val_score, train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
%matplotlib inline
df = pd.read_csv('voice.csv')
```

2. 查看各特征值的数据类型以及数据缺失情况。

```
[2]: df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2535 entries, 0 to 2534
Data columns (total 21 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   meanfreq  2535 non-null   float64
 1   sd        2535 non-null   float64
 2   median    2535 non-null   float64
 3   Q25       2535 non-null   float64
 4   Q75       2535 non-null   float64
 5   IQR       2535 non-null   float64
 6   skew      2535 non-null   float64
 7   kurt      2535 non-null   float64
 8   sp.ent    2535 non-null   float64
 9   sfm       2535 non-null   float64
 10  mode      2535 non-null   float64
 11  centroid  2535 non-null   float64
 12  meanfun   2535 non-null   float64
 13  minfun    2535 non-null   float64
 14  maxfun    2535 non-null   float64
 15  meandom   2535 non-null   float64
 16  mindom    2535 non-null   float64
 17  maxdom    2535 non-null   float64
 18  dfrange   2535 non-null   float64
 19  modindx   2535 non-null   float64
 20  label     2535 non-null   object
dtypes: float64(20), object(1)
memory usage: 416.0+ KB
```

3. 查看各特征值之间的关系，显示热力图。



4. 将数据的标签划分为 y，其他划分为 X，并输出数据集的大小。

```
[4]: X = df.drop('label', axis=1)
     y = df.label
     print(f"'X' shape: {X.shape}")
     print(f"'y' shape: {y.shape}")
```

```
'X' shape: (2535, 20)
'y' shape: (2535,)
```

5. 将数据集进行标准化，即将每个属性的分布转移到平均数为零，标准差为 1。并将数据集划分为训练集和测试集。

```
[5]: scaler = StandardScaler()
     scaler.fit(X)
     X = scaler.transform(X)
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

6. 采用 svm.SVC 模型进行预测，并利用 GridSearchCV 选取最优的参数并输出测试集准确度。

```
[7]: param_grid = {'C': [0.01, 0.1, 0.5, 1, 10, 100],
                  'gamma': [1, 0.75, 0.5, 0.33, 0.3, 0.25, 0.2, 0.1, 0.01],
                  'kernel': ['rbf', 'poly', 'linear']
                 }
     grid = GridSearchCV(SVC(), param_grid, refit=True, verbose=1, cv=10, iid=True, n_jobs = -1)
     grid.fit(X_train,y_train)
     y_pred=grid.predict(X_test)
     print('Accuracy Score:',metrics.accuracy_score(y_test,y_pred))
     print(grid.best_params_)
```

```
Fitting 10 folds for each of 162 candidates, totalling 1620 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    2.8s
[Parallel(n_jobs=-1)]: Done 196 tasks      | elapsed:    4.8s
[Parallel(n_jobs=-1)]: Done 696 tasks      | elapsed:   10.5s
[Parallel(n_jobs=-1)]: Done 1396 tasks      | elapsed:   20.3s
[Parallel(n_jobs=-1)]: Done 1620 out of 1620 | elapsed:   32.6s finished
D:\anaconda3\lib\site-packages\sklearn\model_selection\_search.py:823: FutureWarning: The pa
  "removed in 0.24.", FutureWarning
Accuracy Score: 0.9822485207100592
{'C': 10, 'gamma': 0.33, 'kernel': 'rbf'}
```

7. 将用 svm.SVC 模型的预测结果保存为 result.csv。

```
[7]: grid = grid.fit(X,y)
     df_test = pd.read_csv('test.csv')
     scaler = StandardScaler()
     scaler.fit(df_test)
     df_test = scaler.transform(df_test)
     y_pred = grid.predict(df_test)
     df_test = pd.read_csv('test.csv')
     df_test['label'] = pd.DataFrame(y_pred)
     df_test.to_csv('result.csv')
```

8. 采用决策树模型进行预测，并利用 GridSearchCV 选取最优的参数，并输出测试集准确度。

```
[9]: from sklearn.tree import DecisionTreeClassifier
param_grid = {
    'criterion':['gini','entropy'],
    'max_depth':[2, 3, 4, 5, 6, 7, 8, 9],
    'min_samples_leaf':[2,3,5,10],
    'min_impurity_decrease':[0.1,0.2,0.5]
}
grid = GridSearchCV(DecisionTreeClassifier(), param_grid, refit=True, verbose=1, cv=10, iid=True, n_jobs = -1)
grid.fit(X_train,y_train)
y_pred=grid.predict(X_test)
print('Accuracy Score:',metrics.accuracy_score(y_test,y_pred))
print(grid.best_params_)
```

```
Fitting 10 folds for each of 192 candidates, totalling 1920 fits
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    2.7s
[Parallel(n_jobs=-1)]: Done 560 tasks      | elapsed:    3.8s
Accuracy Score: 0.9526627218934911
{'criterion': 'gini', 'max_depth': 2, 'min_impurity_decrease': 0.1, 'min_samples_leaf': 2}
```

9. 采用逻辑回归模型进行预测，并利用 GridSearchCV 选取最优的参数并输出测试集准确度。

```
[12]: from sklearn.linear_model.logistic import LogisticRegression
param_grid = {
    'C': [0.1, 0.5, 1, 5, 10, 50, 100],
    'penalty':['l1','l2','elasticnet']
}
grid = GridSearchCV(LogisticRegression(), param_grid, refit=True, verbose=1, cv=10, iid=True, n_jobs = -1)
grid.fit(X_train,y_train)
y_pred=grid.predict(X_test)
print('Accuracy Score:',metrics.accuracy_score(y_test,y_pred))
print(grid.best_params_)
```

```
D:\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:144: FutureWarning: The sklearn.linear_model.log
be removed in version 0.24. The corresponding classes / functions should instead be imported from sklearn.li
learn.linear_model is now part of the private API.
  warnings.warn(message, FutureWarning)
[Parallel(n_jobs=-1)]: Using backend LokyBackend with 12 concurrent workers.
Fitting 10 folds for each of 21 candidates, totalling 210 fits
[Parallel(n_jobs=-1)]: Done  26 tasks      | elapsed:    3.6s
Accuracy Score: 0.9664694280078896
{'C': 0.5, 'penalty': 'l2'}
```

六．实验结果（训练集准确度等指标）

实验结果截图：

```
[14]: print('Svc_Accuracy Score:',metrics.accuracy_score(y_test,svc_y_pred))
print('Tree_Accuracy Score:',metrics.accuracy_score(y_test,tree_y_pred))
print('Logic_Accuracy Score:',metrics.accuracy_score(y_test,Logic_y_pred))
```

```
Svc_Accuracy Score: 0.9822485207100592
Tree_Accuracy Score: 0.9526627218934911
Logic_Accuracy Score: 0.9664694280078896
```

实验结果分析：从实验结果可以看出，再利用 GridSearchCV 选择最优参数的情况下 SVM 算法的准确度高于决策树和逻辑回归模型。

七．实验心得（碰到的问题，如何解决，与 logistics 回归和决策树等相比，SVM 算法的优缺点）

遇到的问题：利用 SVC、决策树和逻辑回归各模型训练时，所得到的分数到不是特别理想，并且利用 GridSearchCV 选取最优模型参数时的速度非常慢。

解决方案：先对数据集进行预处理，对各个特征值进行归一化处理，即将每个属性的分布转移到平均数为零，标准差为 1。在进行选取各模型的最优参数，即可得到较为理想的预测结果。

算法优点：使用核函数可以向高维空间进行映射；使用核函数可以解决非线性的分类；分类思想很简单，就是将样本与决策面的间隔最大化；分类效果较好。

算法缺点：SVM 算法对大规模训练样本难以实施；用 SVM 解决多分类问题存在困难；对缺失数据敏感，对参数和核函数的选择敏感。