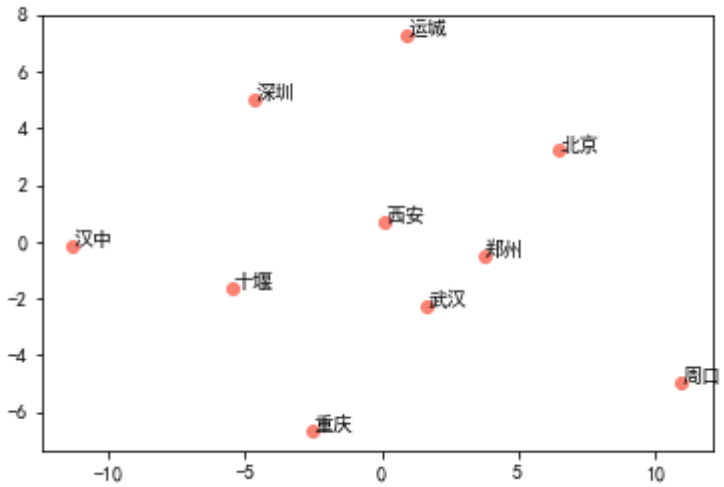
作业7：特征选择与特征提取

1. MDS降维

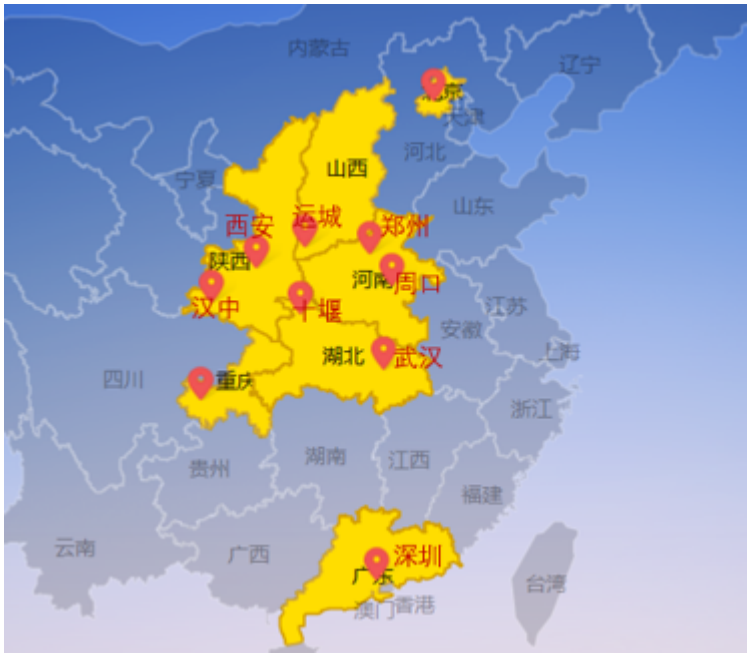
利用sklearn中MDS对数据进行降维：

```
import numpy as np
import pandas as pd
from sklearn.manifold import MDS
import matplotlib.pyplot as plt

df = pd.read_excel('题目1.xlsx','Sheet1')
data = np.array(df.iloc[:,1:])
mds = MDS(dissimilarity='precomputed')
mds.fit(data)
a = mds.embedding_
plt.scatter(a[:,0],a[:,1],color='salmon')
```

用MDS法得到的城市二维表示如下：



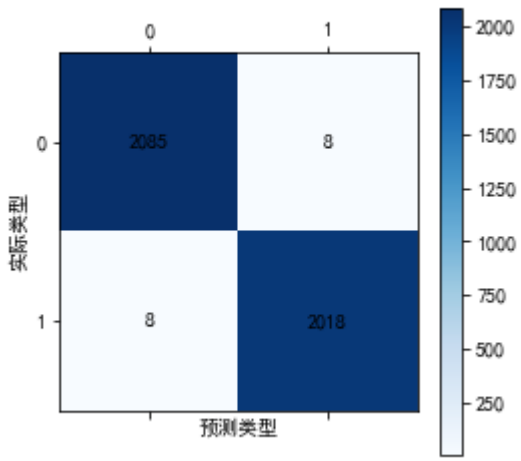
真实地图上各地点的分布如下：



分析：除个别城市外，大部分城市的相对距离与空间位置得到了较为准确的表示，如郑州、武汉、西安、北京、重庆等。但是小部分城市的相对距离出现了不准确的现象，如，汉中距离西安仅有1小时左右的通勤时间，然而降维后却相隔了很长距离；其次，个别城市间的相对位置存在缺陷，如深圳出现在了整个二维表示的西北方、周口则出现在了东南方等。由于MDS根据样本之间的距离或不相似度关系在低维空间里生成对样本的表示，因而本身即有可能出现相对位置偏差。值得注意的一点是，题目中所给的数据为城市铁路交通的通勤时间。由于受交通发达程度和通勤列车类型的影响，通勤时间有时不能很好地刻画城市间的距离，如周口到深圳的直线距离比北京到深圳的距离更近，然而北京到深圳的通勤时间仅为8小时左右，周口到深圳的通勤时间却为23小时，从而在一定程度上造成了周口和深圳相对距离的拉远。

2.PCA, t-SNE，LLE降维

使用隐层为100个节点的卷积神经网络作为分类器，不采用降维时混淆矩阵如下：



此时测试集分类正确率为99.61%（代码见`HW7_2.py`）

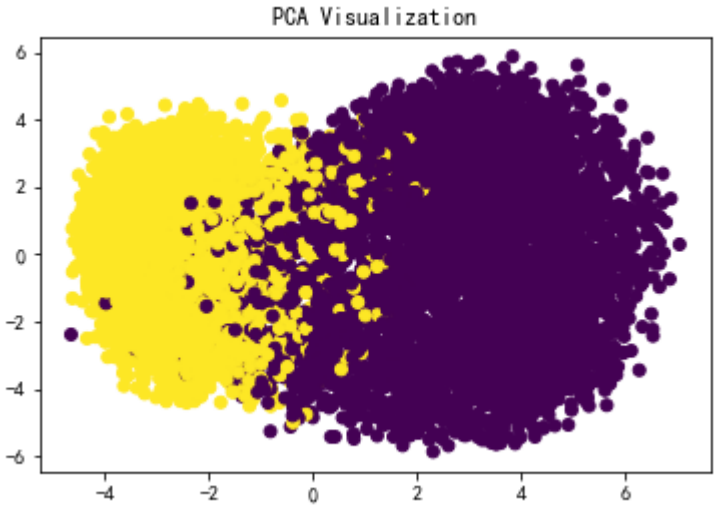
1. PCA降维

(1) 算法流程

- 计算样本均值
- 对数据进行中心化
- 计算协方差矩阵
- 对协方差矩阵进行特征值分解
- 取最大的k个特征值所对应的特征向量构成投影矩阵W
- 投影到$X^{'} = W^{T}X$

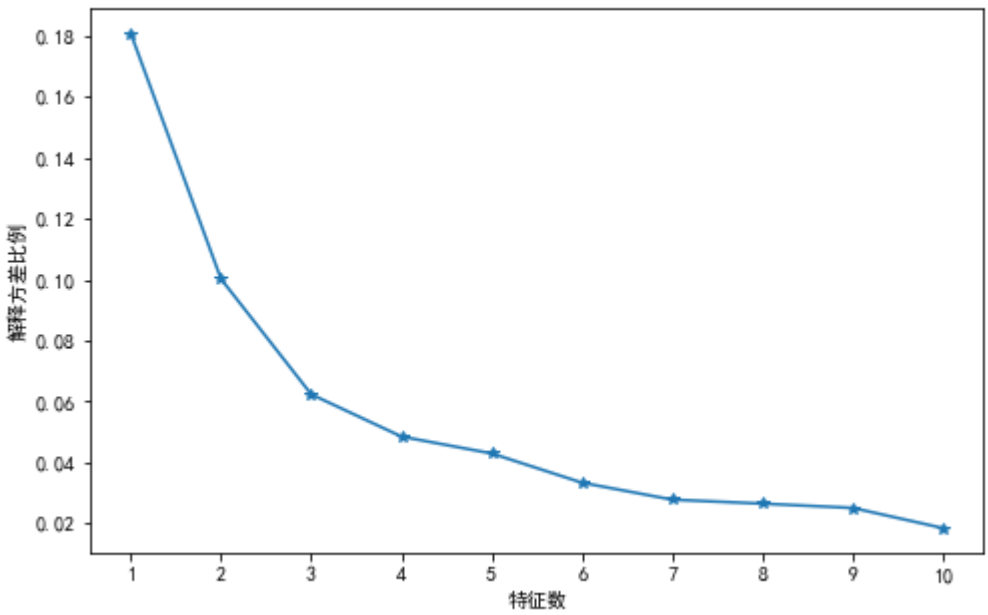
(2) 降维可视化

将数据降维至2维，可视化展现如下：



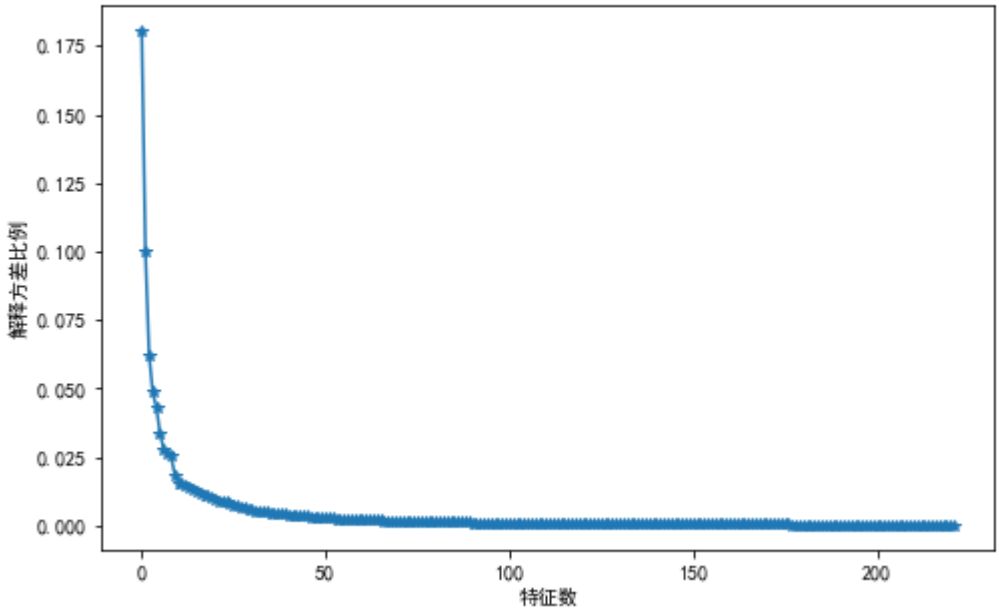
(3) 降维分类效果

前10维中，各维特征所能解释的特征百分比如下：

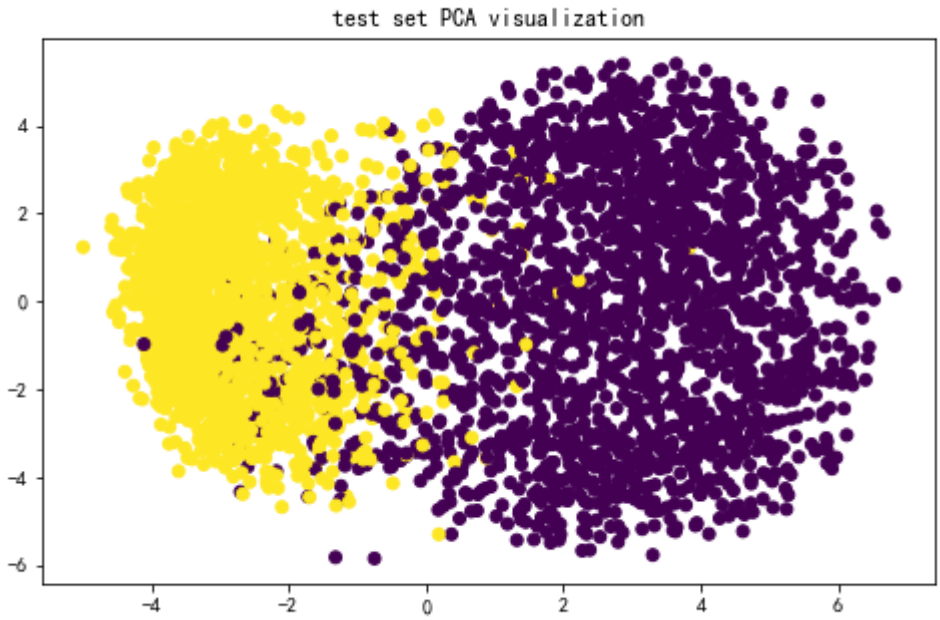


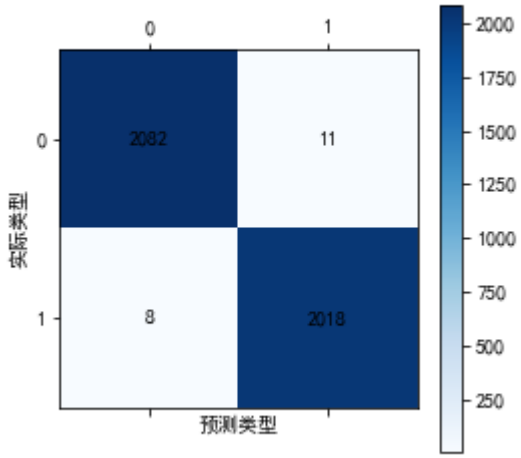
选取隐层节点数为100的单隐层卷积神经网络作为分类器，探究PCA降维后的分类效果。

设定阈值为98%，发现使用前222维特征时即可解释98%的方差，各维特征所解释的百分比如下：



用降维后选择的特征进行训练，此时得到的降维后前两维可视化结果与混淆矩阵如下：





此时，测试集上分类准确率为99.54%，和使用全部特征时相比仅下降约0.1%。可以看到PCA降维可以取得不错的分类效果。

此部分代码见`HW7_2_1.py`，核心代码如下：

```
pca1 = PCA(n_components=0.98)
pca_model2=pca1.fit(x_train_new)

a1= pca1.explained_variance_ratio_
print(a1.shape)

X_test_new = pca1.transform(x_test_new)
X_train_new = pca1.transform(x_train_new)
plt.figure(num=5, figsize=(8, 5))
plt.scatter(X_test_new[:, 0], X_test_new[:, 1],c=y_test_new)
plt.title('test set PCA visualization')
plt.savefig('./pca2.png')
plt.show()
```

2. t-SNE降维

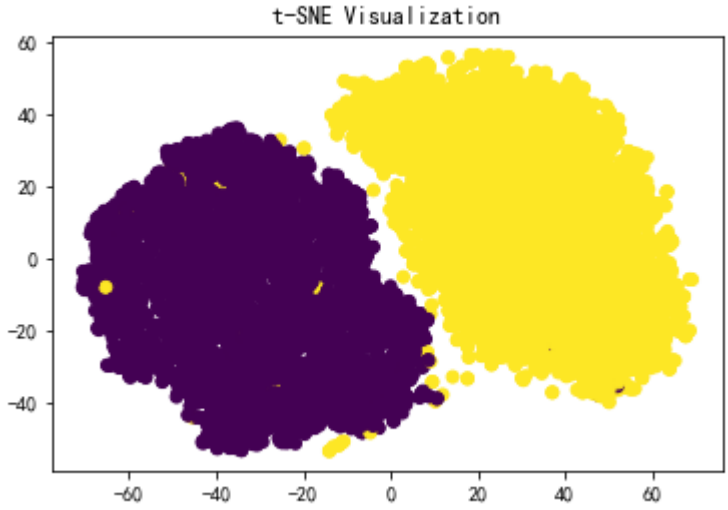
(1) 算法流程

t-SNE利用概率分布来度量样本之间的距离（j样本作为i样本近邻的条件概率）

- 计算高维空间的分布$p_{j|i} = \frac{exp(-||x_i-x_j||^2/2\sigma_i^2)}{\sum_{k\neq i} exp(-||x_i-x_k||^2/2\sigma_i^2)}$
- 降维后的分布$q_{ij} = \frac{(1+||y_i-y_j||^2)^{-1}}{\sum_{k\neq l}(1+||y_k-y_l||^2)^{-1}}$
- 计算分布之间的差异程度KL距离，利用梯度下降等方法优化损失函数

(2) 降维可视化

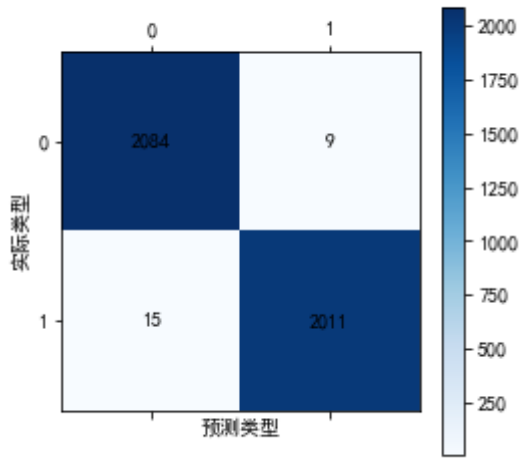
采用t-SNE对数据降维至2维，可视化展现如下：



可以看到两类间出现了明显的分界线，t-SNE可以取得更好的二维展示效果。

(3) 降维分类效果

由于sklearn中的t-SNE模块只含有fit，fit_transfrom函数，因而需要对全部数据进行降维后划分训练集、测试集进行分类。用降维后选择的特征进行训练，得到测试集混淆矩阵如下：



此时测试集正确率为99.41%，比降维前下降0.2%，可以认为t-SNE降维后仍能取得不错的分类效果。

此部分代码见`HW7_2_2.py`，核心代码如下：

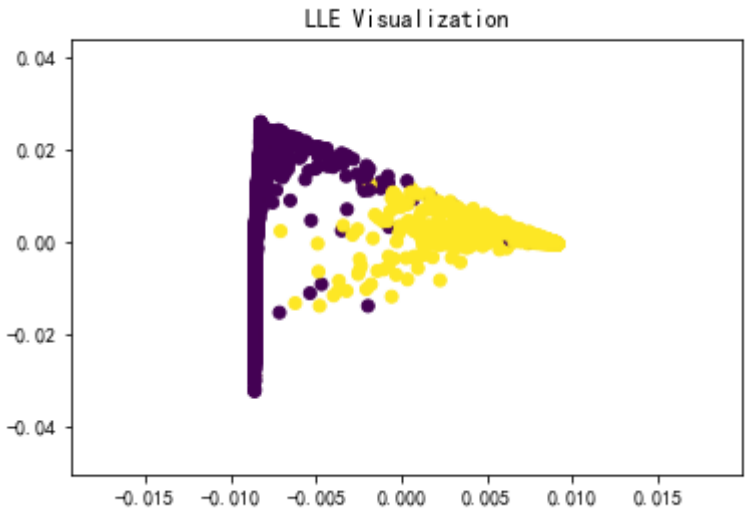
```
tsne = TSNE(n_components=2, learning_rate=100)
tsne_model = tsne.fit_transform(x_data_new)

from sklearn.model_selection import train_test_split
x_train_new, x_test_new, y_train_new, y_test_new = train_test_split(tsne_model,
y_data_new, test_size=0.3, random_state=3)

plt.figure()
plt.scatter(tsne_model[:, 0], tsne_model[:, 1], c=y_data_new)
plt.title('test set t-SNE Visualization')
```

3. LLE降维

(1) 算法流程

- 原空间选择近邻，得到由近邻重构X误差最小的权重W_{ij}，$\epsilon(W)=\sum_i|X_i-\sum_j W_{ij}X_j|^2$
- 利用W_{ij}在低维空间得到重构误差最小的Y，$\Phi(Y)=\sum_i|Y_i-\sum_j W_{ij}Y_j|^2$

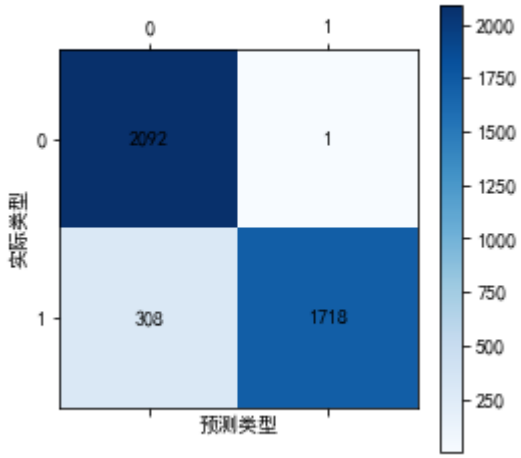
(2) 降维可视化



LLE算法降维后， 样本在低维空间中接近线性分布，两类数据能保持相对独立且数据分布保持原有拓扑结构。

(3) 降维分类效果

LLE采用局部嵌入的方法，因而也需要对全部数据进行降维后划分训练集、测试集进行分类。用降维后选择的特征进行训练，得到测试集混淆矩阵如下：



此时测试集正确率为92.50%，和降维前相比有了较为明显的下降，从混淆矩阵可以看出，分类错误主要出现在一部分数字“0”被误判为数字“8”。

对参数进行调节，将近邻点数由5增加到10后，测试集正确率上升至94.32%，但计算复杂度也随之增大，降维耗时有所增加。

此部分代码见`HW7_2_3.py`，核心代码如下：

```
lle = LocallyLinearEmbedding(n_components=2)
lle_model = lle.fit_transform(x_data_new)

from sklearn.model_selection import train_test_split
x_train_new, x_test_new, y_train_new, y_test_new = train_test_split(lle_model,
y_data_new, test_size=0.3, random_state=3)

plt.figure()
plt.scatter(lle_model[:, 0], lle_model[:, 1], c=y_data_new)
plt.title('LLE Visualization')
```

4.降维效果对比分析

特征提取通过将原特征经过变换后得到新的特征，实现特征空间维数的压缩。其中，PCA属于线性降维算法，t-SNE与LLE属于非线性降维算法。

PCA计算方法简单，利用特征值分解计算耗时少。降维可视化后，两类数据间存在一定程度的重合，没有明显的分界面。从降维后分类效果看，PCA算法可以取得与降维前非常接近的测试集准确率，说明利用低维特征可以对高维数据进行很好的表达。

t-SNE能有效将数据投影到低维空间，降维映射后不同数据在低维空间中仍能被分开且保持严格的分割界面。但t-SNE计算复杂度大，空间复杂度高。经过tSNE降维后，利用神经网络训练，可以取得与降维前接近的测试集准确率，分类效果不错。将t-SNE降维方法用于可视化可能是更好的选择。

LLE为局部嵌入法，可视化后数据在低维空间中接近线性分布，较有特点。LLE的计算复杂度也较大，计算耗时较长。利用LLE降维后，在选用近邻点较少的情况下分类效果会受到一定影响，但总体而言测试集准确率仍在90%以上，通过调节降维时的参数能取得较好的分类效果。

3. 特征选择算法

a.可分性判据选择

数据集中共有400个数据，其中有180个正常组织与220个肿瘤组织，每个组织具有1000个特征。

此处选择Fisher准则与相关性系数进行探究。

- Fisher 准则

Fisher准则是基于类内类间距离的判据

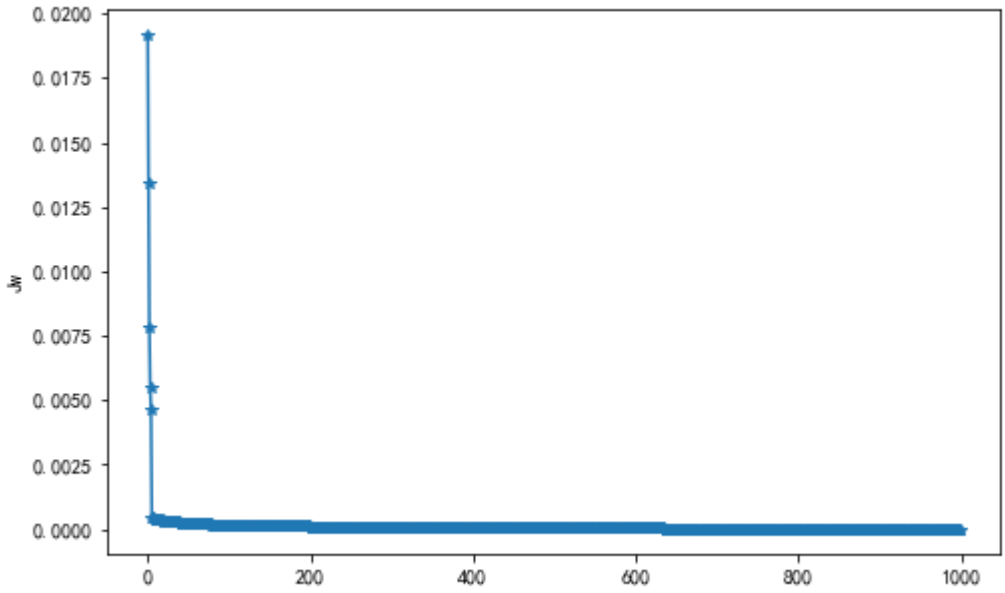
$$J_F(w)=\frac{(\mu_1-\mu_2)^2}{S_1^2+S_2^2}$$

其中类均值$\mu_i=\frac{1}{m_i}\sum_{x_j\in\Gamma_i}x_j(i=1,2)$，类内散度$S_i^2=\sum_{x_j\in\Gamma_i}(x_j-\mu_i)^2\ (i=1,2)$

代码具体实现如下：

```
x_train_df=pd.DataFrame(x_data)
y_train_df=pd.DataFrame(y_data)
y_train_df.columns=[1000]
df_train=pd.concat([x_train_df,y_train_df],axis=1,ignore_index=False)
#按照标签分类
group_by_class=df_train.groupby(1000)
train0_df=group_by_class.get_group(0)
train1_df=group_by_class.get_group(1)
x_0=np.array(train0_df)[:,0:1000]
x_1=np.array(train1_df)[:,0:1000]
#用于排序的辅助函数
def takeOrder(elem):
    return elem[1]
Jw=[]
for i in range(1000):
    m_0=np.mean(x_0[:,i],axis=0)#计算每一列的均值
    m_1=np.mean(x_1[:,i],axis=0)
    n_0=x_0[:,i].shape[0]#计算每一类的数量
    n_1=x_1[:,i].shape[0]
    S_0=0
    S_1=0
    for j in range(n_0):
        S_0=S_0+pow((x_0[j,i]- m_0),2)
    for k in range(n_1):
        S_1=S_1+pow((x_1[k,i]- m_1),2)
    Jw.append([i+1,pow((m_0- m_1),2)/(S_0 + S_1)])
Jw.sort(key=takeOrder,reverse = True)#以Jw为依据进行排序
Jw=np.array(Jw)
```

对利用Fisher准则得到的各特征对应的J_w进行降序排序后，得到J_w随特征数的变化趋势如下：



由图像可以看出，前5个特征对类内类间距离很大影响，因而选择特征数为5，对应特征的列数为48，917，220，416，5。

- 相关系数

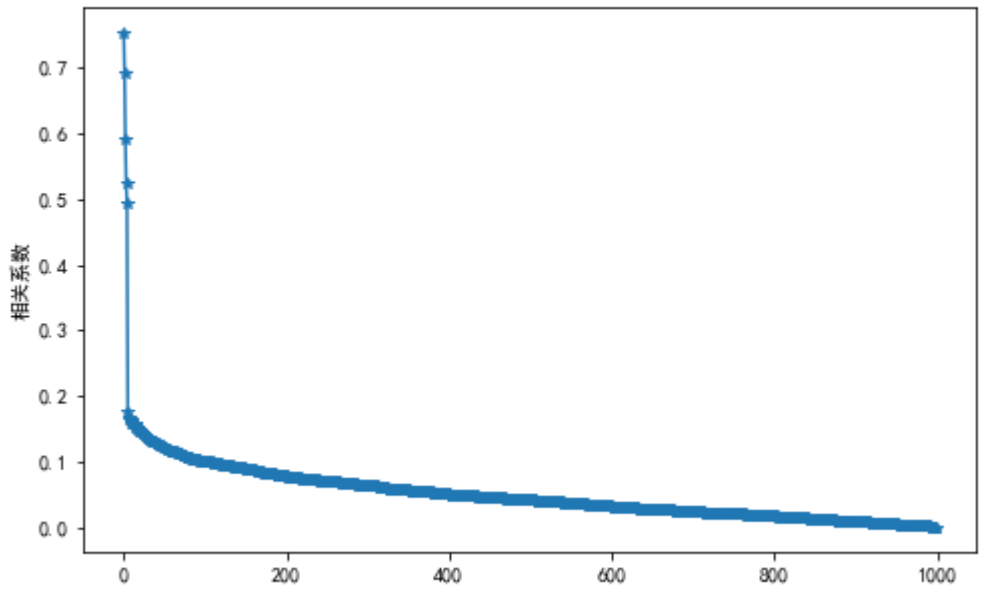
相关系数法将特征取值X和响应值Y（类别标签）都视作随机变量以衡量二者之间的关联程度

$$corr(X,Y)=\frac{\sum_{i=1}^{n}(x_i^{(k)}-\overline{x_i})(y_i^{(k)}-\overline{y_i})}{\sqrt{\sum_{i=1}^{n}(x_i^{(k)}-\overline{x_i})^2\sum_{i=1}^{n}(y_i^{(k)}-\overline{y_i})^2}}$$

代码具体实现如下：

```
Corr=[]
for i in range(1000):
    cor=np.corrcoef(x_data[:,i].T,y_data)
    Corr.append([i+1,abs(cor[0][1])])
Corr.sort(key=takeOrder,reverse = True)
Corr=np.array(Corr)
```

通过对相关系数绝对值进行排序，得到相关系数随特征数的变化趋势如下：



由图像可以看出，前5个特征取值与响应值间具有较强的关联程度，因而选择特征数为5，对应特征的列数为48，917，220，416，5。

（注：此题所指的特征列数均从1开始排序，1,2,...,1000，共有1000列）

b.分类器选择

由于样本数量较少，因而选择使用SVM与logistic regression作为分类器。

在进行特征提取前探究分类器性能，将数据集随机划分为70%训练集与30%的测试集

- SVM

使用SVM做分类器，不做特征提取时选用三次径向基核可以取得较好的分类效果，主要代码如下：

```
from sklearn.svm import SVC
clf = SVC(kernel='poly', degree=3,gamma=10,verbose=1)
clf.fit(x_train_new, y_train_new)
y_predition_train=clf.predict(x_train_new)
y_predition_test=clf.predict(x_test_new)
```

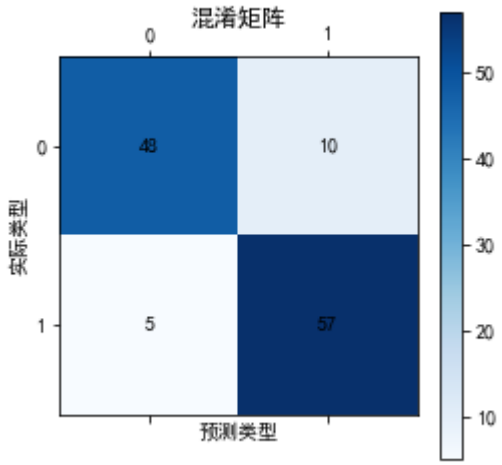
此时训练集上正确率为100%，测试集上正确率为89.17%

- Logistic Regression

选用Logistic Regression做分类器时的主要代码如下：

```
from sklearn.linear_model.logistic import LogisticRegression
from sklearn.metrics import confusion_matrix
classifier=LogisticRegression()
classifier.fit(x_train_new,y_train_new)
y_predict=classifier.predict(x_test_new)
```

此时测试集上的正确率为87.5%，混淆矩阵如下：



可分判据与分类器部分详细代码见`HW7_3.py`.

c.十折交叉验证

采用特征选择后进行十折交叉验证，每次交叉验证均重新挑选特征。由于仅挑选5个特征时，使用Fisher准则与相关性系数得到的特征一致，均为 48、917、219、4、415（排序可能略有不同），因而不同可分性判据对分类结果几乎没有影响。此时分类器正确率结果如下：

选取5个特征	**1**	**2**	**3**	**4**	**5**
Fisher + SVM	0.9	0.95	0.925	0.85	0.8
Fisher + Logistic	0.95	1.0	0.975	0.975	0.925
相关系数 + SVM	0.9	0.95	0.925	0.85	0.8
相关系数 +Logistic	0.95	1.0	0.975	0.975	0.925

选取5个特征	**6**	**7**	**8**	**9**	**10**	**平均**
Fisher + SVM	0.8	0.875	0.95	0.975	0.95	0.8975
Fisher + Logistic	0.925	0.925	0.975	0.975	0.925	0.955
相关系数 + SVM	0.8	0.875	0.95	0.975	0.95	0.8975
相关系数 +Logistic	0.925	0.925	0.975	0.975	0.925	0.955

可以看出使用两种可分判据时，SVM分类器的正确率均为89.75%，Logistic回归的正确率均为95.5%。和不使用特征选择算法是相比，分类器错误率均有所下降，正确率有所提高。

增大所选取的特征数量进行进一步探究，每次选取前 10 个能使可分性判据最大的特征。此时将SVM分类器换为线性核，得到分类器正确率结果如下：

选取10个特征	1	2	3	4	5
Fisher + SVM	0.925	1.0	0.925	0.925	0.9
Fisher + Logistic	0.925	1.0	0.95	0.925	0.95
相关系数 + SVM	0.95	1.0	0.95	0.925	0.95
相关系数 +Logistic	0.975	1.0	0.95	0.925	0.95

选取10个特征	6	7	8	9	10	平均
Fisher + SVM	0.925	1.0	0.875	0.95	0.925	0.935
Fisher + Logistic	0.95	0.975	0.925	0.95	0.925	0.9475
相关系数 + SVM	0.95	1.0	0.925	0.95	0.925	0.9525
相关系数 +Logistic	0.95	1.0	0.925	0.975	0.925	0.9575

此时Fisher准则与相关性系数选取的特征仍十分接近。可以看出，使用Fisher准则时，SVM分类器的正确率为93.5%，Logistic回归的正确率为95.75%；使用相关性关系时，SVM分类器的正确率为95.25%，Logistic回归的正确率为95.75%。和仅选择5个特征数时相比，分类器正确率进一步提高。

此部分代码详见`HW7_3_1.py`，SVM分类器，Logistic回归，Fisher特征选择，相关性关系等已分别写为函数`SVM_Classify`，`Logistic_Regression`，`FisherSelection`，`CorrSelection`。其余核心代码如下：

```
def main():
  kf = KFold(n_splits=10, shuffle=True)
  SVM_fisher_list=[]
  Logistic_fisher_list=[]
  SVM_corr_list=[]
  Logistic_corr_list=[]
  for train_index, test_index in kf.split(data):
      print("************************************************")
      train=data[train_index]
      test=data[test_index]

      featureList = FisherSelection(train)
      train_new = train[:,
[featureList[0],featureList[1],featureList[2],featureList[3],featureList[4],-1]]
      test_new = test[:,
[featureList[0],featureList[1],featureList[2],featureList[3],featureList[4],-1]]
      SVM_fisher_list.append(SVM_Classify(train_new,test_new))
      Logistic_fisher_list.append(Logistic_Regression(train_new,test_new))

      featureList = CorrSelection(train)
      train_new = train[:,
[featureList[0],featureList[1],featureList[2],featureList[3],featureList[4],-1]]
      test_new = test[:,
[featureList[0],featureList[1],featureList[2],featureList[3],featureList[4],-1]]
      SVM_corr_list.append(SVM_Classify(train_new,test_new))
      Logistic_corr_list.append(Logistic_Regression(train_new,test_new))
```

```
    print("*************************************************")
    print("Fisher + SVM Classifier:",np.mean(SVM_fisher_list))
    print("Fisher + Logistic_Regression:",np.mean(Logistic_fisher_list))
    print("Correlation coefficient + SVM Classifier:",np.mean(SVM_corr_list))
    print("Correlation coefficient + Logistic_Regression:",np.mean(Logistic_corr_list))
```

d.结果分析

	SVM	Logistic Regression
不做特征提取	89.17%	87.5%

	Fisher + SVM	相关系数 + SVM	Fisher +Logistic	相关系数 +Logistic
选取5个特征	89.75%	89.75%	95.5%	95.5%
选取10个特征	93.5%	95.25%	94.75%	95.75%

和不做特征选择前相比，分类器的错误率明显下降，正确率均明显提高，特征选择后分类效果能达到95%左右。将不同分类器进行对比可以发现，特征选择后logistic回归的分类效果提升更明显，十折交叉验证的分类效果也更稳定；而不同可分性判据所选择的特征十分接近，对分类结果影响较小。

由实验结果可以预测，随着所选特征数的从零开始逐渐增加，分类效果将先上升后下降。通过特征选择，可以忽略对输出变量预测几乎没有贡献的分量，从而使用尽量少的指标获得足够好的预测。特征选择有利于数据的理解，降低数据存储和计算模型的开销，同时避免维数灾难提高预测性能。

由于 Logistic Regression 属于线性分类器，因而特征选择非常重要。特征选择前，由于数据特征很多、可能满足n<p，因而会出现过拟合现象。模型的参数估计不稳定且模型具有较大的variance，预测效果不佳。

对于SVM分类器而言，采取特征选取前，经测试发现选用三次径向基核可以取得最好的分类效果，测试集上正确率为89.17%；选取10维特征时，若SVM选用三次径向基核，则测试集正确率为90.5%，和不进行特征选择时相比效果略有提高，说明特征选择前模型可能存在过拟合；此时如果尝试选用线性核，则选用10维特征的测试集的正确率约为95%，分类效果可以得到很大提高。这进一步说明不做特征选取时，线性分类器预测效果不佳；而通过合理的选择特征，可以将数据进行有效的降维，从而采用线性分类器即可将不同类型的数据有效分开，取得较好的分类结果。