# 实验三：支持向量机应用实践

学号: 18401190103 姓名： 谢威

学号: 18401190202 姓名： 陆镇涛

学号: 18401190120 姓名： 曹鹏霄

学号: 18401190107 姓名： 王云浩 提交时间： 2020-10-27

## 一、实验目的

1．了解支持向量机相关概念；

2．理解支持向量机构造过程；

3．应用支持向量机解决实际问题

4．学会应用支持向量机可视化工具；

## 实验内容

1. 运用支持向量机对Titanic 乘客生存预测泰坦尼克海难是著名的十大灾难之一，究竟多少人遇难，各方统计的的结果不一。

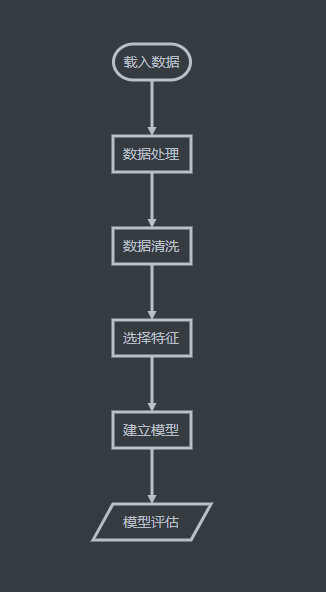
其中数据集格式为 csv，一共有两个文件：train.csv 是训练数据集，包含特征信息和存活与否的标签；test.csv: 测试数据集，只包含特征信息。在训练集中，包括了以下字段，它们具体为：

* 1. 

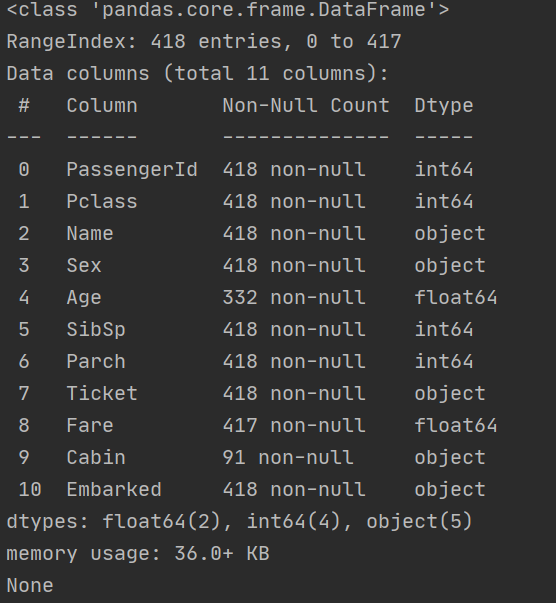
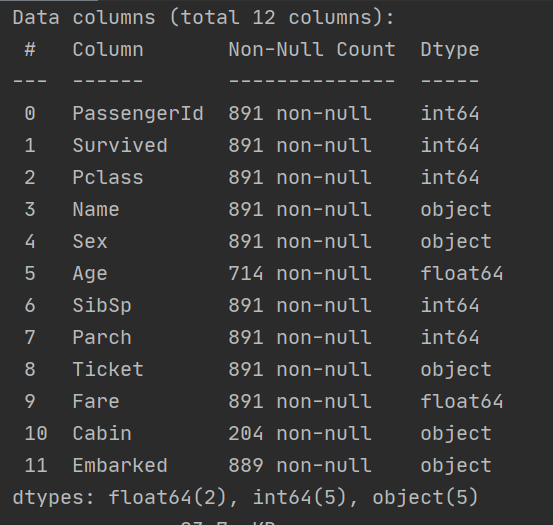
1. 运用支持向量回归模型对血糖进行预测，并对结果进行分析与可视化展示。

## 实现过程分析

题目1：泰坦尼克号生存率预测

按照流程图进行实现

首先进行数据导入，使用info对test和train数据集进行数据分析



可以得知Age、Fare、Cabin、Embarked 这4个字段的数据有所缺失。

其中Age 为年龄字段，是数值型，可以通过平均值进行补齐

Fare为船票价格，是数值型，也可以通过其他人购买船票的平均值进行补齐

Cabin 为船舱，有大量的缺失值。在训练集和测试集中的缺失率分别为 77% 和 78%，无法补齐

Embarked 为为登陆港口，有少量的缺失值，可以通过其他人的数据进行补齐

如此我们完成了数据清洗，接下来进行特征选择与提取

PassenID为乘客编号无用

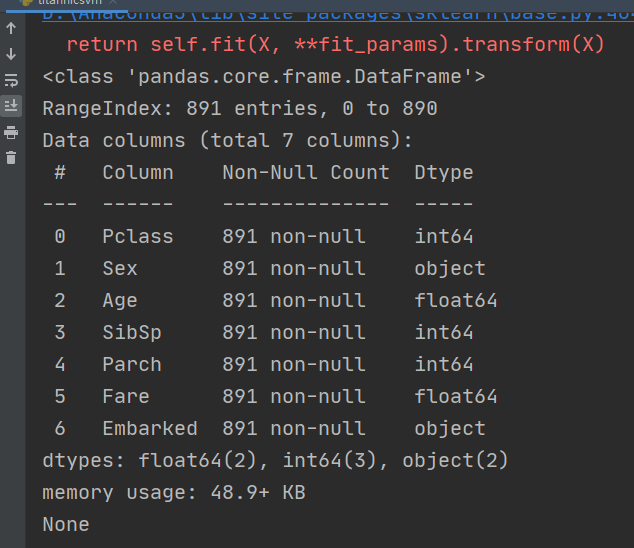
NAME为乘客姓名，无用

Cabin缺省过多，无用

Ticket 字段为船票号码，无用

Pclass、Sex、Age、SibSp、Parch 和 Fare，这些属性分别表示了乘客的船票等级、性别、年龄、亲戚数量以及船票价格，可能会和乘客的生存预测分类有关系。先将 Pclass、Sex、Age 等这些其余的字段作特征，放到特征向量feature中。

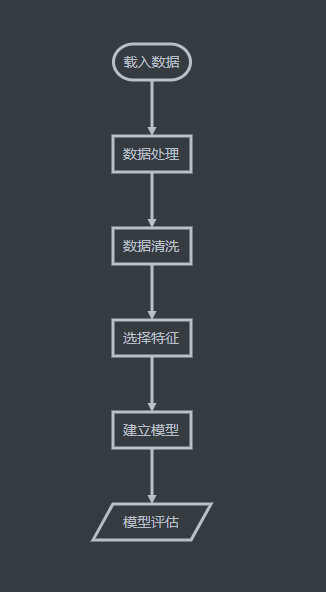
使用info查看feature



我们发现Emarked和sex还不是数据型，我们通过LaberEncoder()，fit\_transfrom()将其转换为数值类型

此时特征提取并清洗完毕，使用lin\_svr=SVC(random\_state=42)函数建立支持向量机模型，使用fit()函数进行训练，然后使用accuracy\_score函数进行预测，最终使用score()函数进行测试与评估模型。最终模型的准确率如下：准确率为0.926163723

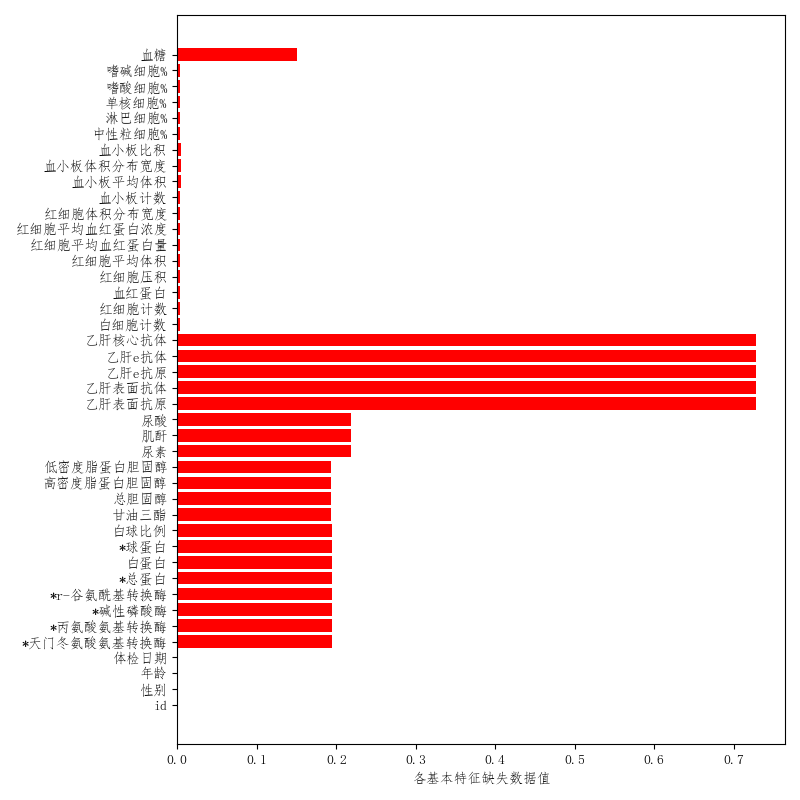
题目2：血糖预测

按照流程图进行实现

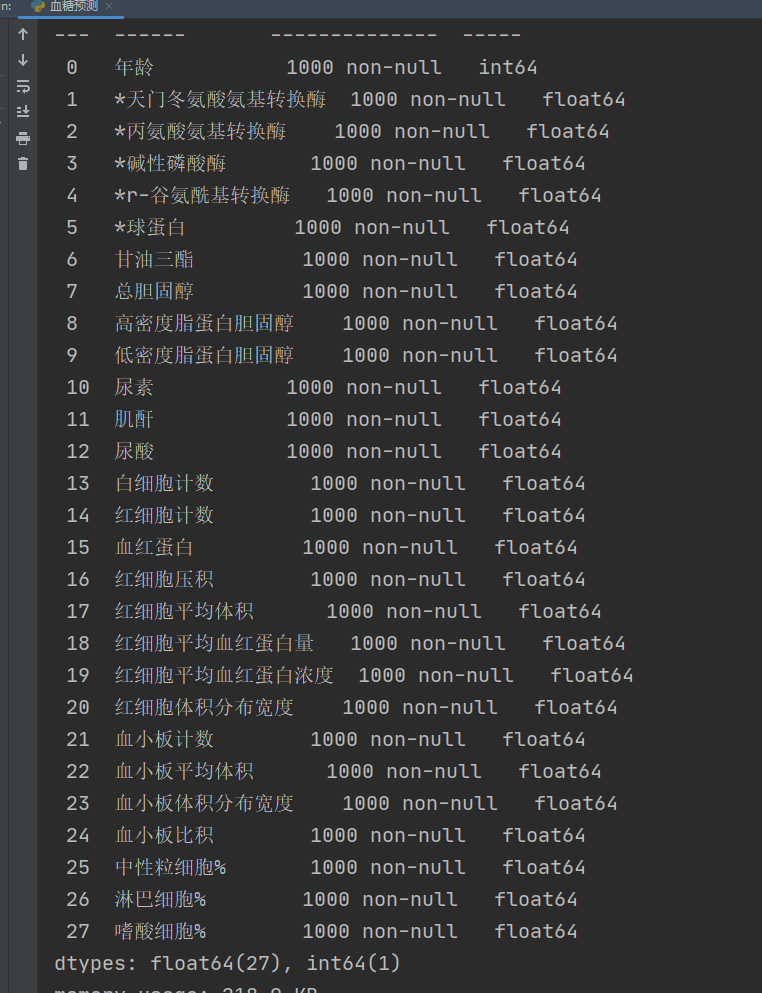
首先利用pandas从三个文件中读入数据，然后通过info函数对数据进行分析



通过分析我们得到，大部分数据虽然都有所缺失，但是缺失值不大，除乙肝表面抗原、乙肝表面抗体、乙肝核心抗体、乙肝e抗原、乙肝e抗体这五个基本特征缺失比例过大，直接删除，其他的数据通过平均值的方法进行补全。

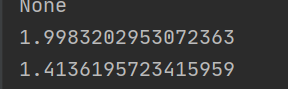


接下来进行特征的选择和提取，通过数据分析我们发现id，体检日期，性别对血糖浓度无影响，直接放弃。蛋白类中发现球蛋白与血糖浓度关系更大，留下球蛋白，除去一个嗜酸细胞。最后将余下字段作为特征放到向量feature中，通过info函数对向量feature进行查看。



最后通过scaller=StandardScaler()对数据进行标准化处理，同时对测试集进行相同的操作让其与训练集的格式相同。

提取完毕后，通过lin\_svr=SVC(random\_state=42)建立支持向量机模型，使用fit函数进行训练，然后使用predict()函数进行预测。最终使用mean\_squared\_error函数进行测试与评估模型。



结果显示训练效果一般，通过RandomizedSearchCV对SVC进行参数优化，使用rnd\_search\_cv =RandomizedSearchCV(SVR(),param\_distributions,n\_iter=10, verbose=2, cv=3, random\_state=42)

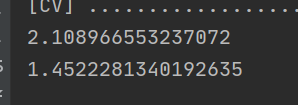
将参数设置为param\_distributions = {

'gamma': reciprocal([0.001, 0.1]),

'C': [uniform(1,10),uniform(10,1)]

}以此对参数进行优化处理

最终预测结果为：



## 源代码

## Titanic

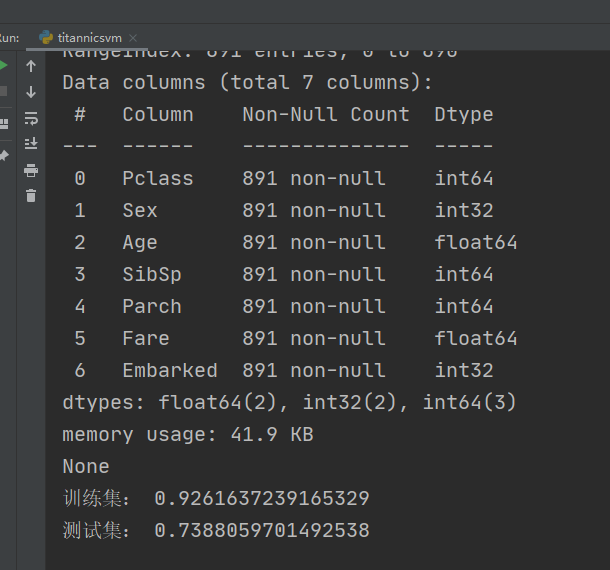
import random  
import pandas as pd  
import numpy as np  
from sklearn.preprocessing import LabelEncoder, StandardScaler # 标签编码  
from sklearn.model\_selection import cross\_val\_score, RandomizedSearchCV, train\_test\_split # k折交叉验证  
from sklearn.svm import SVC  
from sklearn.metrics import accuracy\_score  
train\_data = pd.read\_csv('titanic\_train.csv')  
test\_data = pd.read\_csv('titanic\_test.csv')  
# 数据分析  
# print(test\_data.info())#训练集信息  
  
train\_data['Age'].fillna(train\_data['Age'].mean(), inplace=True)  
train\_data['Fare'].fillna(train\_data['Fare'].mean(), inplace=True)  
train\_data['Embarked'].fillna('S', inplace=True)  
#数据清洗结束，进行特征提取  
feature = ['Pclass', 'Sex', 'Age', 'SibSp', 'Parch', 'Fare', 'Embarked']  
train\_feature = train\_data[feature]  
train\_survived = train\_data['Survived']  
  
#观看feature信息并处理  
# print(train\_feature.info())#观看feature信息  
encoder = LabelEncoder()  
train\_feature['Sex'] = encoder.fit\_transform(train\_feature['Sex'])  
train\_feature['Embarked'] = encoder.fit\_transform(train\_feature['Embarked'])  
print(train\_feature.info())#观看修改过后的feature信息  
  
#归一化  
scaler = StandardScaler()  
train\_feature.astype(float)  
X\_train\_scaled = scaler.fit\_transform(train\_feature)#均值归一化  
feature\_train, feature\_test, survived\_train, survived\_test = train\_test\_split(X\_train\_scaled, train\_survived, test\_size=0.3, random\_state=0)#划分测试集与训练集  
classifier = SVC(C=2, kernel='rbf', gamma=10, decision\_function\_shape='ovo')  
classifier.fit(feature\_train,survived\_train)  
tra\_label = classifier.predict(feature\_train)#训练集的预测标签  
tes\_label = classifier.predict(feature\_test)#测试集的预测标签  
print("训练集：",accuracy\_score(survived\_train, tra\_label))  
print("测试集：",accuracy\_score(survived\_test, tes\_label))

## 血糖预测

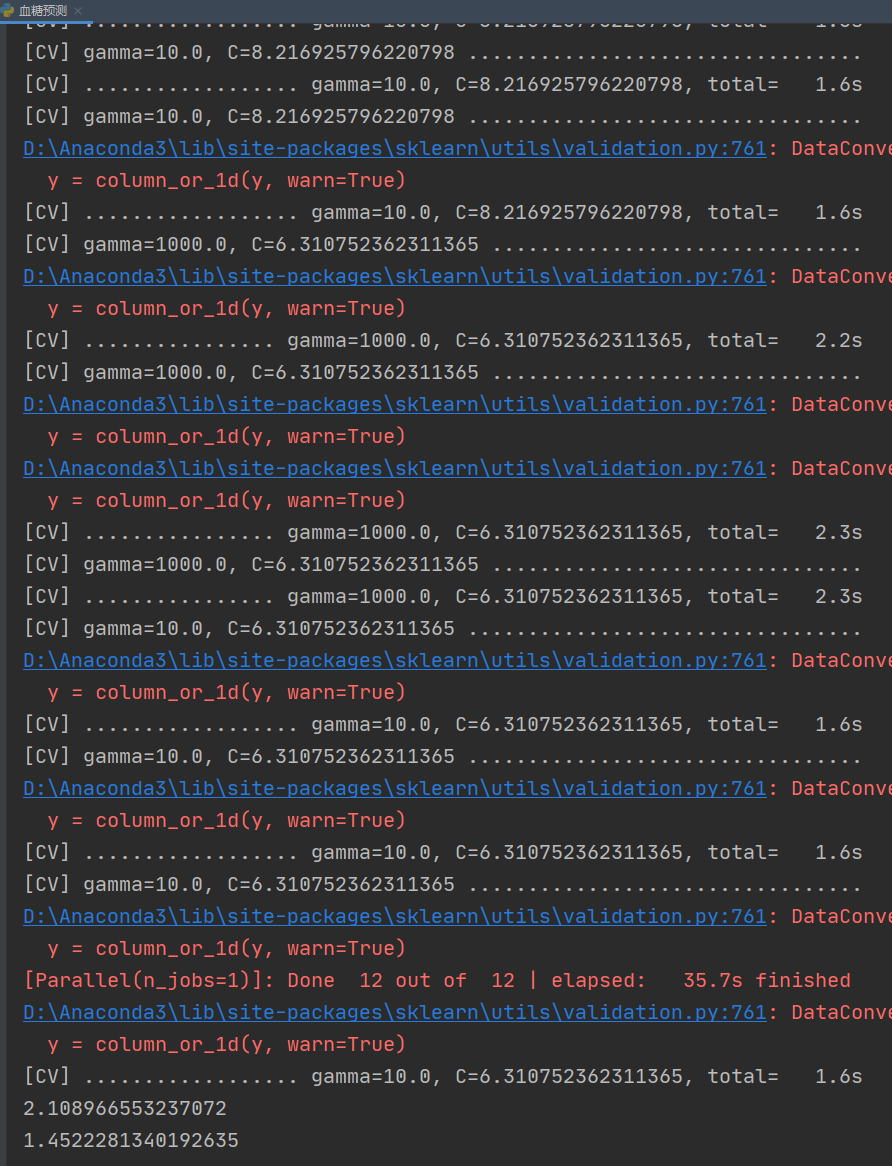
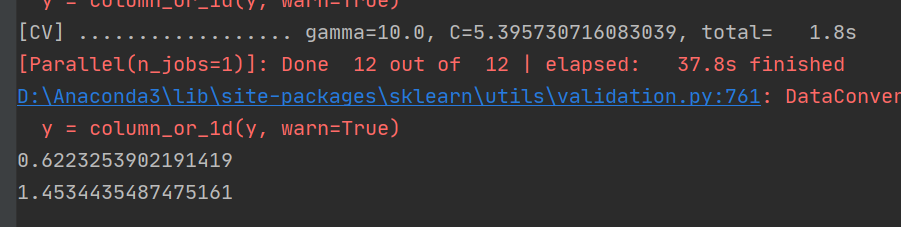
import pandas as pd#导入数据文件  
import numpy as np#科学计算计算库  
from sklearn.preprocessing import LabelEncoder#标签编码  
from sklearn.preprocessing import MinMaxScaler#归一化  
from sklearn.model\_selection import train\_test\_split#数据集的划分  
from sklearn.model\_selection import cross\_val\_score, RandomizedSearchCV  
from sklearn.metrics import mean\_squared\_error#评估函数  
from sklearn.preprocessing import StandardScaler  
from sklearn.svm import LinearSVR ,SVR  
from numpy import reciprocal  
from random import uniform  
#数据读取  
train\_data= pd.read\_csv("d\_train\_20180102.csv",engine='python')  
test\_dataA= pd.read\_csv("d\_test\_A\_20180102.csv",engine='python')  
test\_labels=pd.read\_csv("d\_answer\_a\_20180128.csv",engine='python')  
#数据清理  
train\_data.drop(['乙肝表面抗原', '乙肝表面抗体', '乙肝e抗原', '乙肝e抗体', '乙肝核心抗体', 'id', '体检日期','性别'],axis=1,inplace=True)  
test\_dataA.drop(['乙肝表面抗原', '乙肝表面抗体', '乙肝e抗原', '乙肝e抗体', '乙肝核心抗体', 'id', '体检日期','性别'],axis=1,inplace=True)  
train\_data.drop(['单核细胞%','白球比例','白蛋白','\*总蛋白'],axis=1,inplace=True)  
test\_dataA.drop(['单核细胞%','白球比例','白蛋白','\*总蛋白'],axis=1,inplace=True)  
train\_data.drop(['嗜碱细胞%'],axis=1,inplace=True)  
test\_dataA.drop(['嗜碱细胞%'],axis=1,inplace=True)  
#对列的空值进行填充  
for i in train\_data.columns:  
 train\_data[i].fillna(train\_data[i].mean(), inplace=True)  
for i in test\_dataA:  
 test\_dataA[i].fillna(test\_dataA[i].mean(), inplace=True)  
print(train\_data.info())  
train\_data\_y=train\_data['血糖']  
train\_data.drop(['血糖'],axis=1,inplace=True)  
print(test\_dataA.info())  
#归一化  
scaler = StandardScaler()  
train\_data= scaler.fit\_transform(train\_data.astype(float))  
test\_dataA= scaler.transform(test\_dataA.astype(float))  
#建立模型  
lin\_svr= LinearSVR(random\_state=42,max\_iter=5000)  
lin\_svr.fit(train\_data, train\_data\_y)  
test\_features\_labers= lin\_svr.predict(test\_dataA)  
#评估模型  
mse= mean\_squared\_error(test\_labels, test\_features\_labers)  
print(mse)  
print(np.sqrt(mse))  
#使用RandomizedSearchCV  
param\_distributions = {  
 'gamma': reciprocal([0.001, 0.1]),  
 # 'C': uniform(1,10)  
 'C': [uniform(1,10),uniform(1,10)]  
}  
rnd\_search\_cv =RandomizedSearchCV(SVR(),param\_distributions,n\_iter=10, verbose=2, cv=3, random\_state=42)  
train\_bloods = pd.DataFrame(train\_data\_y)  
rnd\_search\_cv.fit(train\_data, train\_bloods)  
#预测和评估  
test\_features\_labers = rnd\_search\_cv.best\_estimator\_.predict(test\_dataA)  
mse= mean\_squared\_error(test\_labels, test\_features\_labers)  
print(mse)  
print(np.sqrt(mse))

## 实验结果

Titanic



血糖预测

优化前：优化后：

## 结论

1. **泰坦尼克生存模型**

在泰坦尼克生存模型中，Pclass、Sex、Age、SibSp、Parch 和 Fare，这些属性分别表示了乘客的船票等级、性别、年龄、亲戚数量以及船票价格，与预测结果关系较大，将这几个变量加入feature向量中，最终在测试集结果中可以达到0.738的准确率

1. **血糖预测模型**

在血糖预测模型中通过数据分析我们发现id，体检日期，性别对血糖浓度无影响，直接放弃。蛋白类中发现球蛋白与血糖浓度关系更大，留下球蛋白，除去一个嗜酸细胞。最后将余下字段作为特征放到向量feature中，通过info函数对向量feature进行查看。最终的结果利用RandomizedSearchCV对SVC进行参数优化最终的MSE控制在0.622532左右，达到了不错的效果