机器学习在各行各业中广泛使用，如食品安全追溯、智慧交通管理、电商推荐等，促进企业提升运营效率，帮助消费者获得更好的用户体验。**举例说明机器学习在某个行业中的具体应用、优势，存在不足及未来改善的可能，等。**（800-2000字，逻辑要通，排版清晰，请不要网络大幅摘抄）。

机器学习是一种利用数据和算法来让计算机自动学习和优化的技术，它在各个领域都有广泛的应用，其中之一就是医疗行业。机器学习可以帮助医生和病人提高诊断、治疗、预防和管理的效率和质量，同时也可以降低成本和风险。本文将介绍机器学习在医疗行业中的几个具体应用，分别是医学影像分析、基因组学、药物发现。

在医学影像分析方面，机器学习可以通过自动化和智能化的方式，来提高医学影像的分析和识别的效率和准确性。机器学习可以利用深度学习等方法，来训练出能够识别和分类各种疾病和异常的模型，如肿瘤、病变、出血等，从而辅助医生做出更快速和准确的诊断。机器学习也可以利用强化学习等方法，来优化医学影像的采集和处理的过程，从而减少图像的噪声和失真，提高图像的清晰度和质量。但仍存在一些不足，因为它需要大量的高质量的医学影像数据来训练和验证模型，而这些数据的获取和标注往往需要专业的人员和设备，而且涉及到病人的隐私和安全问题，因此数据的可获取性和共享性有限。所以未来可增加医学影像数据的获取和标注的效率和质量，通过使用更先进的仪器和技术，以及采用更合理的数据管理和保护的方法，来提高数据的可获取性和共享性。

在基因组学方面，机器学习可以通过自动化和智能化的方式，来提高基因组的分析和解读的效率和准确性。机器学习可以利用聚类、分类、回归等方法，来分析基因组的结构和功能，如基因的位置、功能、表达、相互作用等，从而辅助研究人员发现基因的规律和意义。机器学习也可以利用关联、预测、推荐等方法，来解读基因组的影响和作用，如基因与表型、疾病、药物等的关系，从而辅助医生和病人做出更合理和个性化的决策。但仍存在一些不足，它需要大量的高质量的基因组数据来训练和验证模型，而这些数据的获取和标注往往需要专业的人员和设备，而且涉及到生物体问题。所以未来可增加基因组数据的获取和标注的效率和质量，通过使用更先进的仪器和技术，以及采用更合理的数据管理和保护的方法，来提高数据的可获取性和共享性。

在药物发现方面，机器学习可以通过自动化和智能化的方式，来提高药物发现的效率和准确性。机器学习可以利用生成对抗网络等方法，来生成和优化新的药物分子，从而辅助化学家设计更有效和安全的药物候选物。机器学习也可以利用图神经网络等方法，来分析和预测药物分子与靶点、受体、酶等的相互作用，从而辅助生物学家评估和选择更合适的药物靶标。但仍存在一些不足，它需要大量的高质量的药物分子和靶点数据来训练和验证模型，而这些数据的获取和标注往往需要专业的人员和设备，而且涉及到药物的专利和安全问题，因此数据的可获取性和共享性有限，以及需要高性能的计算机和算法来处理和分析药物分子和靶点，而这些计算机和算法的成本和复杂度都很高，因此技术的可普及性和扩展性有限。所以在未来可增加药物分子和靶点数据的获取和标注的效率和质量，通过使用更先进的仪器和技术，以及采用更合理的数据管理和保护的方法，来提高数据的可获取性和共享性。还可以降低计算机和算法的成本和复杂度，通过使用更高效的硬件和软件，以及采用更简洁的模型和方法，来提高技术的可普及性和扩展性。

今后随着机器学习技术的不断进步和医疗数据的不断增加，机器学习将会在医疗领域发挥更大的作用，为医疗行业带来更多的创新和变革。机器学习将会与其他人工智能技术如深度学习、自然语言处理、智能机器人等相结合，实现更高层次的智能化和自动化。机器学习也将会与其他医疗技术如基因编辑、纳米医疗、虚拟现实等相融合，实现更前沿的医疗方法和手段。机器学习还将会与互联网、物联网、云计算等技术相结合，实现更广泛的医疗服务和应用。总之，机器学习将会为医疗行业带来更多的机遇和挑战，也将会为人类的健康和福祉做出更大的贡献。

**2.利用线性模型解决问题（回归预测分析或二分类）。**

附：Sklearn常用数据集

|  |  |  |
| --- | --- | --- |
| 序号 | 数据集 | 学号尾号 |
| 1 | 鸢尾花数据集：load\_iris（） | 0 1 |
| 2 | 手写数字数据集：load\_digits（） | 2 3 |
| 3 | 乳腺癌数据集load-barest-cancer（） | 4 5 |
| 4 | 糖尿病数据集：load-diabetes（） | 6 7 |
| 5 | 波士顿房价数据集：load-boston（） | 8 |
| 6 | 体能训练数据集：load-linnerud（） | 9 |

利用以上数据集（学号尾号不同采用不同数据集）对线性模型（Ridge、Lasso回归）进行分析，确定哪个回归（哪种参数下）更适合本数据集（准确率更高，不同回归、不同参数下的不同准确率要列出）。

要求：

1）数据集要解释；

2）代码要列出，重点代码加注释说明，特别是自己调试过程中的自我理解；

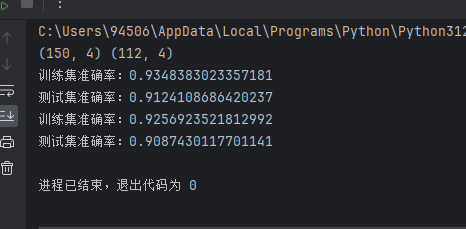
3）运行结果要截图，结果要文字解释；

4）注意排版

**岭回归：**

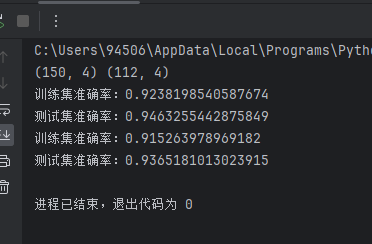
import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import Ridge  
from sklearn.linear\_model import Lasso  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
x,y = load\_iris().data, load\_iris().target  
# 将数据集拆分成训练集和测试集  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state = 50)  
print(x.shape,x\_train.shape)  
  
ridge = Ridge(alpha = 0.01).fit(x\_train,y\_train)# 设置岭回归参数进行训练  
print('训练集准确率：',end='')  
print(ridge.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(ridge.score(x\_test,y\_test))  
  
ridge = Ridge(alpha = 10).fit(x\_train,y\_train)# 设置岭回归参数进行训练  
print('训练集准确率：',end='')  
print(ridge.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(ridge.score(x\_test,y\_test))  
  
# 训练4个不同alpha参数的岭回归  
ridge01 = Ridge(alpha = 0.1).fit(x\_train,y\_train)  
ridge1 = Ridge(alpha = 1).fit(x\_train,y\_train)  
ridge5 = Ridge(alpha = 5).fit(x\_train,y\_train)  
ridge10 = Ridge(alpha = 10).fit(x\_train,y\_train)  
  
#绘制模型系数  
plt.plot(ridge01.coef\_,'s',label = 'Ridge alpha = 0.1')  
plt.plot(ridge1.coef\_,'^',label = 'Ridge alpha = 1')  
plt.plot(ridge5.coef\_,'v',label = 'Ridge alpha = 5')  
plt.plot(ridge10.coef\_,'o',label = 'Ridge alpha = 10')  
  
plt.xlabel("coefficient index")  
plt.ylabel("coefficient magnitude")  
plt.hlines(0,0,len(ridge01.coef\_)) #绘制一条直线作为参考线  
plt.legend(loc = 'best')  
plt.show()

**结果：**



**在alpha值为0.01至10时，训练集准确率>测试集准确率，也可看出准确率非常高，此前将random数据修改至10至90，alph数据修改至0.001至1000仍然得出结果为训练集准确率>测试集准确率，并且都在0.9以上，但过拟合的问题仍未解决。**

**当random state=100时训练集准确率<测试集准确率，准确率仍然很高，并且解决了过拟合的情况。**



**在此图可看出：**

**alpha=0.1时，模型权重蓝色正方形最小值小于-0.1，最大值接近0.6**

**随着alpha值逐渐增大，岭回归的权重离散度下降**

**当alpha=10时，模型权重红色圆形最小值开始靠近0附近**

图表, 散点图

描述已自动生成

**Lasso回归：**

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import Ridge  
from sklearn.linear\_model import Lasso  
from sklearn.datasets import load\_iris  
from sklearn.model\_selection import train\_test\_split  
x,y = load\_iris().data, load\_iris().target  
# 将数据集拆分成训练集和测试集  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state = 100)  
print(x.shape,x\_train.shape)  
  
lasso = Lasso(alpha = 0.01).fit(x\_train,y\_train)# 设置套索回归参数进行训练  
print('训练集准确率：',end='')  
print(lasso.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(lasso.score(x\_test,y\_test))  
  
lasso1 = Lasso(alpha = 0.001).fit(x\_train,y\_train)# 设置套索回归参数进行训练  
print('训练集准确率：',end='')  
print(lasso1.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(lasso1.score(x\_test,y\_test))  
  
# 训练4个不同alpha参数的套索回归  
Lasso01 = Lasso(alpha = 0.1).fit(x\_train,y\_train)  
Lasso001 = Lasso(alpha = 0.01).fit(x\_train,y\_train)  
Lasso0001 = Lasso(alpha = 0.001).fit(x\_train,y\_train)  
Lasso1 = Lasso(alpha = 1).fit(x\_train,y\_train)  
  
#绘制模型系数  
plt.plot(Lasso01.coef\_,'s',label = 'Ridge alpha = 0.1')  
plt.plot(Lasso001.coef\_,'^',label = 'Ridge alpha = 0.01')  
plt.plot(Lasso0001.coef\_,'v',label = 'Ridge alpha = 0.001')  
plt.plot(Lasso1.coef\_,'o',label = 'Ridge alpha = 1')  
  
plt.xlabel("coefficient index")  
plt.ylabel("coefficient magnitude")  
plt.hlines(0,0,len(Lasso01.coef\_)) #绘制一条直线作为参考线  
plt.legend(loc = 'best')  
plt.show()

**结果：**

文本

描述已自动生成

**Alpha值为0.01和1时训练集准确率<测试集准确率，未出现过拟合情况，但alpha值逐渐增大时，准确率大大减小**

图表

描述已自动生成

**能看出：alpha=1.0时，模型的权重红色圆形基本被压缩至0，无非0特征**

**随着alpha值逐渐减小，模型的权重个数增加**

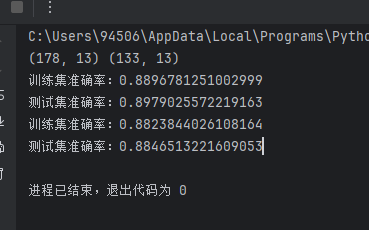
**当alpha=0.001时，复杂度最高，但并未出现过拟合**

**3.[附加题]**利用线性模型解决**自己遇到的真实问题**。[解题要求同题2]

红酒数据集分析——岭回归：

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as plt  
from sklearn.linear\_model import Ridge  
from sklearn.linear\_model import Lasso  
from sklearn.datasets import load\_wine  
from sklearn.model\_selection import train\_test\_split  
x,y = load\_wine().data, load\_wine().target  
# 将数据集拆分成训练集和测试集  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state = 100)  
print(x.shape,x\_train.shape)  
  
ridge = Ridge(alpha = 0.01).fit(x\_train,y\_train)# 设置岭回归参数进行训练  
print('训练集准确率：',end='')  
print(ridge.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(ridge.score(x\_test,y\_test))  
  
ridge = Ridge(alpha = 10).fit(x\_train,y\_train)# 设置岭回归参数进行训练  
print('训练集准确率：',end='')  
print(ridge.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(ridge.score(x\_test,y\_test))  
  
# 训练4个不同alpha参数的岭回归  
ridge01 = Ridge(alpha = 0.1).fit(x\_train,y\_train)  
ridge1 = Ridge(alpha = 1).fit(x\_train,y\_train)  
ridge5 = Ridge(alpha = 5).fit(x\_train,y\_train)  
ridge10 = Ridge(alpha = 10).fit(x\_train,y\_train)  
  
#绘制模型系数  
plt.plot(ridge01.coef\_,'s',label = 'Ridge alpha = 0.1')  
plt.plot(ridge1.coef\_,'^',label = 'Ridge alpha = 1')  
plt.plot(ridge5.coef\_,'v',label = 'Ridge alpha = 5')  
plt.plot(ridge10.coef\_,'o',label = 'Ridge alpha = 10')  
  
plt.xlabel("coefficient index")  
plt.ylabel("coefficient magnitude")  
plt.hlines(0,0,len(ridge01.coef\_)) #绘制一条直线作为参考线  
plt.legend(loc = 'best')  
plt.show()

**结果：**



**训练集准确率<测试集准确率，未出现过拟合**

图表, 散点图

描述已自动生成**可看出：alpha=0.1时，模型权重蓝色正方形最小值小于-0.45，最大值接近0.1**

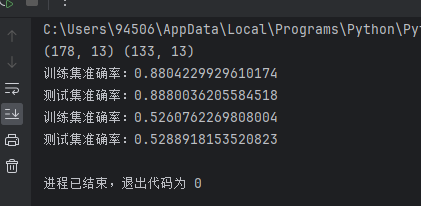
**随着alpha值逐渐增大，岭回归的权重离散度下降**

**当alpha=10时，模型权重红色圆形最小值开始靠近0**

红酒数据集分析——套索回归：

import matplotlib.pyplot as plt  
from sklearn.linear\_model import Lasso  
from sklearn.datasets import load\_wine  
from sklearn.model\_selection import train\_test\_split  
x,y = load\_wine().data, load\_wine().target  
# 将数据集拆分成训练集和测试集  
x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,random\_state = 100)  
print(x.shape,x\_train.shape)  
  
lasso = Lasso(alpha = 0.01).fit(x\_train,y\_train)# 设置lasso回归参数进行训练  
print('训练集准确率：',end='')  
print(lasso.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(lasso.score(x\_test,y\_test))  
  
lasso1 = Lasso(alpha = 0.5).fit(x\_train,y\_train)# 设置lasso回归参数进行训练  
print('训练集准确率：',end='')  
print(lasso1.score(x\_train,y\_train))  
print('测试集准确率：',end='')  
print(lasso1.score(x\_test,y\_test))  
  
# 训练4个不同alpha参数的lasso回归  
Lasso01 = Lasso(alpha = 0.1).fit(x\_train,y\_train)  
Lasso001 = Lasso(alpha = 0.01).fit(x\_train,y\_train)  
Lasso0001 = Lasso(alpha = 0.001).fit(x\_train,y\_train)  
Lasso1 = Lasso(alpha = 1).fit(x\_train,y\_train)  
  
#绘制模型系数  
plt.plot(Lasso01.coef\_,'s',label = 'Ridge alpha = 0.1')  
plt.plot(Lasso001.coef\_,'^',label = 'Ridge alpha = 0.01')  
plt.plot(Lasso0001.coef\_,'v',label = 'Ridge alpha = 0.001')  
plt.plot(Lasso1.coef\_,'o',label = 'Ridge alpha = 1')  
  
plt.xlabel("coefficient index")  
plt.ylabel("coefficient magnitude")  
plt.hlines(0,0,len(Lasso01.coef\_)) #绘制一条直线作为参考线  
plt.legend(loc = 'best')  
plt.show()

**结果：**



**训练集准确率<测试集准确率，未出现过拟合情况，但alpha值逐渐增大时，准确率大大减小**

图表, 散点图

描述已自动生成

**能看出：alpha=1.0时，模型的权重红色圆形基本被压缩至0，无非0特征**

**随着alpha值逐渐减小，模型的权重个数增加**

**当alpha=0.001时，复杂度最高，绿色倒三角出现过拟合情况**