**1、问题一代码**

|  |  |
| --- | --- |
| **（1）数据预处理** | **Python程序** |
| import pandas as pd  #Mo数据与ER数据匹配得到Mo\_ER.csv  data = pd.read\_csv("Mo\_ER.csv")    data.head() #显示前5行  type(data)    data != 0 #data！= 0创建一个布尔数据帧,它是真的,其中df是非零的    #返回一个布尔系列,表示哪些列具有非零项.  #any操作将沿0轴的值–即沿着行–聚合成一个布尔值.因此结果是每列的一个布尔值.  (data != 0).any(axis=0)    #data.loc可用于选择这些列  data.loc[:, (data != 0).any(axis=0)]    #要“删除”零列,请重新分配data  df = data.loc[:, (data != 0).any(axis=0)]  #数据输出  df.to\_csv("Mo\_ER干净.csv") | |

|  |  |
| --- | --- |
| **（2）25个变量的相关图** | **R软件** |
| #25个变量的相关图  library(psych)  library(corrplot)#载入两个包  #相关系数矩阵可视化  r=read.csv("C:\\Users\\yg\\Desktop\\数学建模\\25个.csv",header=TRUE)  rnew<-r#所要分析的那几列  cormat<-corr.test(rnew)  corrplot(cormat$r)#  #20个变量的相关图  rr=read.csv("C:\\Users\\yg\\Desktop\\数学建模\\20个.csv",header=TRUE)  rrnew<-r#所要分析的那几列  cormat<-corr.test(rrnew)  corrplot(cormat$r) | |

|  |  |
| --- | --- |
| **（3）小提琴图** | **Python代码** |
| import pandas as pd  import seaborn as sns  import numpy as np  #导入数据  data = pd.read\_csv("描述性分析.csv",encoding='gbk')  data.head() #查看前5行  a=df[['pIC50',"MDEC-23",'nC']]  sns.violinplot(data=a)  b=df[['minsOH',"maxssO",'MDEC-33','SHBint10']]  sns.violinplot(data=b)  c=df[['C1SP2',"nHBAcc"]]  sns.violinplot(data=c)  d=df[['minHsOH','MDEO-12','ATSc3']]  sns.violinplot(data=d)  e=df[['minHBint5','minHBint10']]  sns.violinplot(data=e)  f=df[['MLFER\_A',"VC-5",'ETA\_Shape\_Y']]  sns.violinplot(data=f)  g=df[['ALogP']]  sns.violinplot(data=g)  e=df[["minsssN",'SHsOH','BCUTc-1h']]  sns.violinplot(data=e) | |

|  |  |
| --- | --- |
| **（4）随机森林提取20个变量Y=pIC50** | **Python代码** |
| from sklearn.feature\_selection import SelectKBest  from sklearn.feature\_selection import chi2  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.ensemble import RandomForestRegressor  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split    data=pd.read\_csv('try.csv')  x = data.iloc[:,1:]  y = data.iloc[:,0]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)    rf\_model = RandomForestRegressor(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)    predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前25个变量  features\_df.iloc[0:25]  #预测结果可视化  score = rf\_model.score(x\_train, y\_train)  result = rf\_model.predict(x\_train)  plt.figure() #只取前一百条可视化，因为数据太多了  plt.plot(np.arange(100), y\_train[:100], "go-", label="True value")  plt.plot(np.arange(100), result[:100], "ro-", label="Predict value")  plt.title(f"RandomForest---score:{score}")  plt.legend(loc="best")  plt.show()    #保存数据  features\_df.to\_csv("PIC50变量贡献排序.csv")    #特征重要性排序条形图  from matplotlib import pyplot as plt    feat=features\_df.iloc[:25]  index=list(feat.iloc[:,0])  values=list(feat.iloc[:,1])  color=["red","red","red","red","red",'blue','blue','blue','blue','blue','blue','blue','blue','blue',  'blue','blue','blue','blue','blue','blue','blue','blue','blue','blue','blue']  plt.bar(index,values,color=color)  plt.xticks(index,index, rotation=90)  plt.xlabel("Features")  plt.ylabel("Importance")  plt.show() | |

**2、问题二代码**

|  |  |
| --- | --- |
| **ERα生物活性的定量预测模型** | **Python代码** |
| data1= pd.read\_csv('D:\ssjm\data1.csv',encoding='gbk')  data= pd.read\_csv('D:\ssjm\data.csv',encoding='gbk')#导入数据  from sklearn import linear\_model  from sklearn.ensemble import RandomForestRegressor  from sklearn.ensemble import GradientBoostingRegressor  from sklearn.ensemble import BaggingRegressor  from sklearn.model\_selection import cross\_val\_predict  from sklearn.model\_selection import cross\_val\_score#加载模块  target=data['pIC50']#导入pIC50  #运行模型  model1\_linear = linear\_model.LinearRegression()  Model2\_RF = RandomForestRegressor()  Model3\_GBDT = GradientBoostingRegressor()  Model4\_Bagging = BaggingRegressor()  #用10折交叉验证  neg\_mse1=cross\_val\_score(model1\_linear, data1, target,scoring="neg\_mean\_squared\_error", cv=10).mean()#线性模型负的均方误差  neg\_mae1=cross\_val\_score(model1\_linear, data1, target,scoring="neg\_ mean\_absolute\_error ", cv=10).mean()#线性模型负的平均绝对误差  neg\_mse2=cross\_val\_score(model2\_RF, data1, target,scoring="neg\_mean\_squared\_error", cv=10).mean()#随机森林模型负的均方误差  neg\_mae2=cross\_val\_score(model2\_RF, data1, target,scoring="neg\_mean\_absolute\_error", cv=10).mean()#随机森林模型负的平均绝对误差  neg\_mse3=cross\_val\_score(model3\_GBDT,data1, target,scoring="neg\_mean\_squared\_error", cv=10).mean()#GradientBoostingRegressor模型负的均方误差  neg\_mae3=cross\_val\_score(model3\_GBDT,data1, target,scoring="neg\_mean\_absolute\_error", cv=10).mean()#GradientBoostingRegressor模型负的平均绝对误差  neg\_mse4=cross\_val\_score(model4\_Bagging, data1,target,scoring="neg\_mean\_squared\_error", cv=10).mean()3BaggingRegressooor模型负的均方误差  neg\_mae4=cross\_val\_score(model4\_Bagging, data1,target,scoring="neg\_mean\_absolute\_error", cv=10).mean()3BaggingRegressooor模型负的均绝对误差  #输出误差大小  print(neg\_mse1)  print(neg\_mse2)  print(neg\_mse3)  print(neg\_mse4)  print(neg\_mae1)  print(neg\_mae2)  print(neg\_mae3)  print(neg\_mae4)  #4个模型分别计算预测值  pIC50\_pred1=cross\_val\_predict(model1\_linear,data1, target, cv=10)  pIC50\_pred2=cross\_val\_predict(model2\_RF,data1, target, cv=10)  pIC50\_pred3=cross\_val\_predict(model3\_GBDT,data1, target, cv=10)  pIC50\_pred4=cross\_val\_predict(model4\_Bagging,data1, target, cv=10)    #由误差的大小选择model3\_GBDT    #模型输出误差MSE的比较图  modelnames = ['linear',  'RFR'  ,'CBDT',  'Bagging']  neg\_mse = [-1.12513708163,  -1.02537941288,  -0.908961091013,  -0.973407805883]  plt.figure(figsize=(10,9))  fig = plt.figure(figsize=(10,6))  ax = fig.add\_subplot(1, 1, 1)  ticks = ax.set\_xticks(range(0,5))#设定x轴有5个标签  ax.plot(neg\_mse,'ko--')  labels = ax.set\_xticklabels(modelnames,fontsize='14',rotation=90)  plt.title("model comparsion",fontsize='14')#设置图片标题  plt.grid(True)  plt.savefig('D:\\ssjm\\a.jpg')#保存图片  plt.show()    #model1\_linear拟合图  plt.figure(figsize=(12, 6))  plt.plot(list(pIC50\_pred1),label="forcast")  plt.plot(list(target),label="test")  plt.ylabel('pIC50',fontsize=14,horizontalalignment='center')  plt.xlabel('model1\_linear',fontsize=14,horizontalalignment='center')  plt.legend()  plt.savefig('D:\\ssjm\\拟合图1.jpg')#保存图片  plt.show()  #model2\_RFr拟合图  plt.figure(figsize=(12, 6))  plt.plot(list(pIC50\_pred2),label="forcast")  plt.plot(list(target),label="test")  plt.ylabel('pIC50',fontsize=14,horizontalalignment='center')  plt.xlabel('RF',fontsize=14,horizontalalignment='center')  plt.legend()  plt.savefig('D:\\ssjm\\拟合图2.jpg')#保存图片  plt.show()  #model3\_GBDT拟合图  plt.figure(figsize=(12, 6))  plt.plot(list(pIC50\_pred3),label="forcast")  plt.plot(list(target),label="test")  plt.ylabel('pIC50',fontsize=14,horizontalalignment='center')  plt.xlabel('GBDT',fontsize=14,horizontalalignment='center')  plt.legend()  plt.savefig('D:\\ssjm\\拟合图3.jpg')#保存图片  plt.show()  #model4\_Bagging拟合图  plt.figure(figsize=(12, 6))  plt.plot(list(pIC50\_pred4),label="forcast")  plt.plot(list(target),label="test")  plt.ylabel('pIC50',fontsize=14,horizontalalignment='center')  plt.xlabel('Bagging',fontsize=14,horizontalalignment='center')  plt.legend()  plt.savefig('D:\\ssjm\\拟合图4.jpg')#保存图片  plt.show()    #model3\_GBDT预测测试集的pIC50  test= pd.read\_csv('D:\\ssjm\\test.csv',encoding='gbk')  Model3\_GBDT = model3GBDT.fit(data1,target)  predict\_test=model3\_GBDT.predict(test)  #根据题目中给定的测试集，进行预测  data2=pd.DataFrame(predict\_test) #预测数据  data2  data2.to\_csv("D:\\ssjm\\连续变量的预测结果.csv")#保存预测结果 | |

**3、问题三代码**

|  |  |
| --- | --- |
| **（1）随机森林对5个分类变量进行特征提取** | **Python代码** |
| df5=pd.read\_csv('try分类变量.csv')  #预测对第一个分类变量的贡献程度  x = df5.iloc[:,5:]  y = df5.iloc[:,0]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)  rf\_model = RandomForestClassifier(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)  #rf\_model.score(x\_test,y\_test)  predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前30个变量  aa1=features\_df.iloc[0:30]  aa1  #预测对第二个分类变量的贡献程度  x = df5.iloc[:,5:]  y = df5.iloc[:,1]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)    rf\_model = RandomForestClassifier(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)  predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前30个变量  aa2=features\_df.iloc[0:30]  aa2  #预测对第三个分类变量的贡献程度  x = df5.iloc[:,5:]  y = df5.iloc[:,2]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)    rf\_model = RandomForestClassifier(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)    predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前30个变量  aa3=features\_df.iloc[0:30]  aa3    #预测对第四个分类变量的贡献程度  x = df5.iloc[:,5:]  y = df5.iloc[:,3]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)    rf\_model = RandomForestClassifier(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)    predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前30个变量  aa4=features\_df.iloc[0:30]  aa4    #预测对第五个分类变量的贡献程度  x = df5.iloc[:,5:]  y = df5.iloc[:,4]  x\_train,x\_test,y\_train,y\_test = train\_test\_split(x,y,test\_size=0.3,random\_state=1)    rf\_model = RandomForestClassifier(n\_estimators=100)  rf\_model.fit(x\_train,y\_train)    predict = rf\_model.predict(x\_test)  features = x.columns  feature\_importances = rf\_model.feature\_importances\_  features\_df = pd.DataFrame({'Features':features,'Importance':feature\_importances})  features\_df.sort\_values('Importance',inplace=True,ascending=False)  #提取贡献程度大的前30个变量  aa5=features\_df.iloc[0:30]  aa5    aa1.to\_csv("aa1.csv") #变量排序  aa2.to\_csv("aa2.csv")  aa3.to\_csv("aa3.csv")  aa4.to\_csv("aa4.csv")  aa5.to\_csv("aa5.csv") | |

|  |  |
| --- | --- |
| **（2）ROC图** | **Python代码** |
| y\_train\_pred = rf\_model.predict(x\_train)  y\_test\_pred = rf\_model.predict(x\_test)    # accuracy  train\_accuracy = rf\_model.score(x\_train, y\_train)  test\_accuracy = rf\_model.score(x\_test, y\_test)  # precision  train\_precision = precision\_score(y\_train, y\_train\_pred)  test\_precision = precision\_score(y\_test, y\_test\_pred)  # recall  train\_recall = recall\_score(y\_train, y\_train\_pred)  test\_recall = recall\_score(y\_test, y\_test\_pred)  # f1  train\_f1 = f1\_score(y\_train, y\_train\_pred)  test\_f1 = f1\_score(y\_test, y\_test\_pred)  # auc 计算时，计算的应该是不同的概率画出来的曲线下的面积,而不是预测值对应的曲线下的面积  # 预测值 分类模型，应该全是0 或者 1 ，但是概率是类似于得分一样的值  # 根据资料貌似两种都行，都可以作为阈值来进行ROC曲线的绘制  y\_train\_pred = rf\_model.predict\_proba(x\_train)[:, 1]  y\_test\_pred = rf\_model.predict\_proba(x\_test)[:, 1]  train\_auc = roc\_auc\_score(y\_train, y\_train\_pred)  test\_auc = roc\_auc\_score(y\_test, y\_test\_pred)    from IPython.core.interactiveshell import InteractiveShell #使所有结果都输出  InteractiveShell.ast\_node\_interactivity = "all"    #训练集  train\_accuracy  train\_precision  train\_recall  train\_f1  train\_auc  #测试集  test\_accuracy  test\_precision  test\_recall, test\_f1  test\_auc    def draw\_roc\_curve(train\_pre\_proba, test\_pre\_proba, train\_auc, test\_auc, model\_name):  fpr, tpr, roc\_auc = train\_pre\_proba  test\_fpr, test\_tpr, test\_roc\_auc = test\_pre\_proba    plt.figure()  lw = 2  plt.plot(fpr, tpr, color='darkorange',  lw=lw, label='ROC curve (area = %0.2f)' % train\_auc)  plt.plot(test\_fpr, test\_tpr, color='red',  lw=lw, label='ROC curve (area = %0.2f)' % test\_auc)  plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')  plt.xlabel('False Positive Rate')  plt.ylabel('True Positive Rate')  plt.title('Roc example ' + model\_name)  plt.legend(loc="lower right")  plt.show()    y\_train\_pred = rf\_model.predict\_proba(x\_train)[:, 1]  y\_test\_pred = rf\_model.predict\_proba(x\_test)[:, 1]    train\_roc = roc\_curve(y\_train, y\_train\_pred)  test\_roc = roc\_curve(y\_test, y\_test\_pred)    train\_auc = roc\_auc\_score(y\_train, y\_train\_pred)  test\_auc = roc\_auc\_score(y\_test, y\_test\_pred)    draw\_roc\_curve(train\_roc, test\_roc, train\_auc, test\_auc, rf\_model) | |

|  |  |
| --- | --- |
| **（3）XGBoost对5个分类变量预测** | **Python代码** |
| from IPython.core.interactiveshell import InteractiveShell #使所有结果都输出  InteractiveShell.ast\_node\_interactivity = "all"    import os  import pandas as pd  import numpy as np  from sklearn.utils import shuffle  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score  import matplotlib.pyplot as plt  import seaborn as sns  %matplotlib inline  from pandas import DataFrame  import xgboost as xgb    #对第一个分类变量预测  df = pd.read\_csv('分类1train.csv')  print(df.shape)  df.head()  # 准备数据  target = ['SMILES','Caco-2']  X = df.drop(target, axis=1)  y = df['Caco-2']  print(X.shape, y.shape)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost.sklearn import XGBClassifier  ## 定义 XGBoost模型  clf = XGBClassifier()  # 在训练集上训练XGBoost模型  clf.fit(X\_train, y\_train)  test\_predict = clf.predict(X\_test)  #test\_predict  pd.DataFrame(test\_predict) #预测结果  pd.DataFrame(y\_test) #Y原本的数据  #根据题目中给定的训练集，进行预测  aa=pd.read\_csv('分类1test.csv')  pred=clf.predict(aa)  pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量1的预测结果.csv")    #对第二个分类变量预测  df = pd.read\_csv('分类2train.csv')  print(df.shape)  df.head()  # 准备数据  target = ['SMILES','CYP3A4']  X = df.drop(target, axis=1)  y = df['CYP3A4']  print(X.shape, y.shape)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost.sklearn import XGBClassifier  ## 定义 XGBoost模型  clf = XGBClassifier()  # 在训练集上训练XGBoost模型  clf.fit(X\_train, y\_train)  test\_predict = clf.predict(X\_test)  #test\_predict  pd.DataFrame(test\_predict) #预测结果  pd.DataFrame(y\_test) #Y原本的数据  #根据题目中给定的训练集，进行预测  aa=pd.read\_csv('分类2test.csv')  pred=clf.predict(aa)  pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量2的预测结果.csv")    #对第三个分类变量预测  df = pd.read\_csv('分类3train.csv')  print(df.shape)  df.head()  # 准备数据  target = ['SMILES','hERG']  X = df.drop(target, axis=1)  y = df['hERG']  print(X.shape, y.shape)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost.sklearn import XGBClassifier  ## 定义 XGBoost模型  clf = XGBClassifier()  # 在训练集上训练XGBoost模型  clf.fit(X\_train, y\_train)  test\_predict = clf.predict(X\_test)  #test\_predict  pd.DataFrame(test\_predict) #预测结果  pd.DataFrame(y\_test) #Y原本的数据  #根据题目中给定的训练集，进行预测  aa=pd.read\_csv('分类3test.csv')  pred=clf.predict(aa)  pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量3的预测结果.csv")pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量2的预测结果.csv")    #对第四个分类变量预测  df = pd.read\_csv('分类4train.csv')  print(df.shape)  df.head()  # 准备数据  target = ['SMILES','HOB']  X = df.drop(target, axis=1)  y = df['HOB']  print(X.shape, y.shape)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost.sklearn import XGBClassifier  ## 定义 XGBoost模型  clf = XGBClassifier()  # 在训练集上训练XGBoost模型  clf.fit(X\_train, y\_train)  test\_predict = clf.predict(X\_test)  #test\_predict  pd.DataFrame(test\_predict) #预测结果  pd.DataFrame(y\_test) #Y原本的数据  #根据题目中给定的训练集，进行预测  aa=pd.read\_csv('分类4test.csv')  pred=clf.predict(aa)  pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量4的预测结果.csv")    #对第五个分类变量预测  df = pd.read\_csv('分类5train.csv')  print(df.shape)  df.head()  # 准备数据  target = ['SMILES','HOB']  X = df.drop(target, axis=1)  y = df['HOB']  print(X.shape, y.shape)  X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost.sklearn import XGBClassifier  ## 定义 XGBoost模型  clf = XGBClassifier()  # 在训练集上训练XGBoost模型  clf.fit(X\_train, y\_train)  test\_predict = clf.predict(X\_test)  #test\_predict  pd.DataFrame(test\_predict) #预测结果  pd.DataFrame(y\_test) #Y原本的数据  #根据题目中给定的训练集，进行预测  aa=pd.read\_csv('分类5test.csv')  pred=clf.predict(aa)  pred  data1=pd.DataFrame(pred) #y的预测数据  data1  data1.to\_csv("分类变量5的预测结果.csv") | |

**四、问题四代码**

|  |  |
| --- | --- |
| **（1）SHAP特征提取** | **Python代码** |
| import shap  from sklearn.feature\_selection import SelectKBest  from sklearn.feature\_selection import chi2  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  from sklearn.ensemble import RandomForestRegressor  from sklearn.ensemble import RandomForestClassifier  from sklearn.model\_selection import train\_test\_split  from sklearn.metrics import accuracy\_score  from sklearn import metrics  from sklearn.preprocessing import label\_binarize  from sklearn.metrics import roc\_curve, auc ###计算roc和auc  from sklearn import model\_selection    from sklearn.model\_selection import cross\_val\_score  from sklearn.ensemble import AdaBoostClassifier  from sklearn.datasets import make\_blobs  import matplotlib as plt  from sklearn.tree import DecisionTreeClassifier  import pandas as pd  import numpy as np  from sklearn.impute import SimpleImputer  from sklearn.preprocessing import OrdinalEncoder  from sklearn.model\_selection import train\_test\_split  from sklearn.linear\_model import LogisticRegression  from sklearn.svm import SVC  from sklearn.tree import DecisionTreeClassifier  from sklearn.ensemble import RandomForestClassifier  from xgboost import XGBClassifier  import matplotlib.pyplot as plt  from sklearn.preprocessing import minmax\_scale  from sklearn.metrics import precision\_score, roc\_curve, recall\_score, f1\_score, roc\_auc\_score, accuracy\_score  import warnings    import os  import pandas as pd  import numpy as np  from sklearn.utils import shuffle  from sklearn.model\_selection import train\_test\_split  from sklearn.ensemble import RandomForestClassifier  from sklearn.metrics import accuracy\_score  import matplotlib.pyplot as plt  import seaborn as sns  %matplotlib inline  from pandas import DataFrame  import xgboost as xgb    df = pd.read\_csv('633数据.csv')  print(df.shape)  df.head()    # 准备数据  target = ['SMILES','pIC50']  X = df.drop(target, axis=1)  y = df['pIC50']  print(X.shape, y.shape)    X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=123)  from xgboost import XGBRegressor  rf\_model = XGBRegressor(max\_depth=4,n\_estimators=150, learning\_rate=0.05)  rf\_model.fit(X\_train, y\_train, early\_stopping\_rounds=5,  eval\_set=[(X\_test, y\_test)], verbose=False)    #采用shap（SHapley Additive exPlanation）验证模型  explainer = shap.TreeExplainer(rf\_model) #解释器  shap\_values = explainer.shap\_values(df.drop(target, axis=1))  print(shap\_values.shape) #504个特征，所以是504列    # The SHAP value plot可以进一步显示预测因子与目标变量之间的正、负关系  shap.summary\_plot(shap\_values, X\_train) | |

|  |  |
| --- | --- |
| **（2）模型拟合** | **Stata16.0** |
| gen mdec23\_2=mdec23\*mdec23  gen c1sp2\_2=c1sp2\*c1sp2  gen minhsoh\_2=minhsoh\*minhsoh  gen mdeo12\_2=mdeo12\*mdeo12  gen bcutc1h\_2=bcutc1h\*bcutc1h  gen minsssn\_2=minsssn\*minsssn  gen mlfer\_a\_2=mlfer\_a\* mlfer\_a  gen mdec33\_2=mdec33\*mdec33  gen vc5\_2=vc5\*vc5  gen alogp\_2=alogp\*alogp  reg pic50 mdec23 c1sp2 minhsoh mdeo12 bcutc1h minsssn mlfer\_a mdec33 vc5 alogp mdec23\_2 c1sp2\_2 minhsoh\_2 mdeo12\_2 bcutc1h\_2 minsssn\_2 mlfer\_a\_2 mdec33\_2 vc5\_2 alogp\_2,noconstant  reg pic50 mdec23 c1sp2 mdeo12 bcutc1h minsssn mlfer\_a mdec33 vc5 alogp mdec23\_2 c1sp2\_2 mdeo12\_2 bcutc1h\_2 minsssn\_2 mlfer\_a\_2 mdec33\_2 vc5\_2,noconstant  estimates store reg1  esttab reg1 using Word.rtf, ar2 b(%6.4f) t(%6.4f) | |

|  |  |
| --- | --- |
| **（3）包络算法** | **MATLAB** |
| clc;clear all;close all;  Target1 = xlsread(‘baoluosuanfa’);  [X,Y] = sort(Target1(:,18));  New\_Sort\_Matrix = Target1(Y,:);  length\_Cell = floor(0.2\*size(New\_Sort\_Matrix,1));  Name = {'MDEC-23','C1SP2','MDEO-12','BCUTc-1h','minsssN','MLFER\_A','MDEC-33','VC-5','ALogP','MDEC-23\_2','C1SP2\_2','MDEO-12\_2','BCUTc-1h\_2','minsssN\_2','MLFER\_A\_2','MDEC-33\_2','VC-5\_1'};  for Length\_ii = 1:1:17  fprintf('%d.分子描述符 %s 的取值范围为 %.2f to %.2f\n',Length\_ii,Name{Length\_ii}, min(New\_Sort\_Matrix(:,Length\_ii)),max(New\_Sort\_Matrix(:,Length\_ii)));  end | |