第六章

1.实践案例一

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

import seaborn as sns

import scipy.stats as stats

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error,r2\_score,mean\_absolute\_error

data = pd.read\_excel(r"C:\Users\26634\Desktop\多元回归模型案例.xlsx")

display(data)

data.info()

scaler = StandardScaler()

x = data.drop(['y'],axis=1)

y = data['y']

scaler = StandardScaler()

x = scaler.fit\_transform(x)

y\_mean = np.mean(y)

y\_std = np.std(y)

y = (y-y\_mean)/y\_std

x\_train,x\_test,y\_train,y\_test=train\_test\_split(x,y,train\_size=0.8, random\_state=66)

reg = LinearRegression()

reg.fit(x\_train,y\_train)

y\_pred = reg.predict(x\_test)

np.set\_printoptions(precision=6, suppress=True)

print("n\_features\_in\_ = " ,reg.n\_features\_in\_)

print('reg.coef\_ = ' ,reg.coef\_)

print('reg.intercept\_ = ' ,reg.intercept\_)

print("train\_score = %.6f" %reg.score(x\_train,y\_train))

print("test\_score = %.6f" %reg.score(x\_test,y\_test))

print("mse = %.6f" %mean\_squared\_error(y\_test,y\_pred))

print("mae = %.6f" %mean\_absolute\_error(y\_test,y\_pred))

print("r2 = %.6f" %r2\_score(y\_test,y\_pred))

print("adj-r2=%.6f"%(1-mean\_squared\_error(y\_test,y\_pred)\*(len(y)-1)/np.var(y\_test)/(len(y)-reg.n\_features\_in\_-1)))

y\_pred\_all = reg.predict(x)

y\_pred\_all = reg.predict(x)

y\_pred\_all = (y\_pred\_all\*y\_std)+y\_mean

result = pd.concat([data['y'],pd.DataFrame(y\_pred\_all)],axis=1)

result.to\_excel(r"C:\Users\11389\Desktop\result.xlsx")

2.实践案例二

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

import seaborn as sns

import scipy.stats as stats

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import classification\_report,confusion\_matrix

from sklearn.datasets import load\_iris

iris = load\_iris()

x = iris.data

y = iris.target

x\_train,x\_test, y\_train, y\_test = train\_test\_split(x, y, train\_size=0.8, random\_state=66)

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

clf = LogisticRegression()

clf.fit(x\_train,y\_train)

y\_pred = clf.predict(x\_test) #0,1分类

y\_pred\_prob = clf.predict\_proba(x\_test)

print("train\_score = ",clf.score(x\_train,y\_train))

print('test\_score = ',clf.score(x\_test,y\_test))

cm = confusion\_matrix(y\_test, y\_pred)

sns.heatmap(cm,annot=True, fmt='d', cmap='Blues', xticklabels=iris.target\_names, yticklabels=iris.target\_names)

plt.title('Confusion Matrix for Multi-Class Classification')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

display(confusion\_matrix(y\_test, y\_pred))

print(classification\_report(y\_test, y\_pred, target\_names=iris.target\_names))

3.实验一

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

import seaborn as sns

import scipy.stats as stats

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.preprocessing import StandardScaler

from sklearn.metrics import mean\_squared\_error,r2\_score,mean\_absolute\_error

data = pd.read\_csv(r"C:\Users\26634\Desktop\house\_price\_regression\_dataset.csv")

display(data)

data.info()

x = data.drop(['House\_Price'],axis=1)

y = data['House\_Price']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x,y,train\_size=0.8, random\_state=66)

reg = LinearRegression()

reg.fit(x\_train,y\_train)

y\_pred = reg.predict(x\_test)

print("n\_features\_in\_ = ",reg.n\_features\_in\_)

print('reg.coef\_ = ' , reg.coef\_)

print('reg.intercept\_ = ' , reg.intercept\_)

print("train\_score = %.6f" %reg.score(x\_train,y\_train))

print("test\_score = %.6f" %reg.score(x\_test,y\_test))

print("mse = %.1f" %mean\_squared\_error(y\_test,y\_pred))

print("mae = %.1f" %mean\_absolute\_error(y\_test,y\_pred))

print("r2 = %.6f" %r2\_score(y\_test,y\_pred))

print("adj-r2=%.6f"%(1-mean\_squared\_error(y\_test,y\_pred)\*(len(y)-1)/np.var(y\_test)/(len(y)-reg.n\_features\_in\_-1)))

pred\_df = pd.concat([pd.DataFrame(y),pd.DataFrame(reg.predict(x))], axis=1)

pred\_df.to\_excel(r"C:\Users\26634\Desktop\ house\_price预测结果导出.xlsx")

4.实验二

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd

import warnings

warnings.filterwarnings('ignore')

import seaborn as sns

import scipy.stats as stats

from sklearn.model\_selection import train\_test\_split

from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LogisticRegression

from sklearn.preprocessing import StandardScaler

from sklearn.model\_selection import GridSearchCV,RandomizedSearchCV,learning\_curve

from sklearn.metrics import classification\_report,confusion\_matrix

data = pd.read\_excel(r"C:\Users\26634\Desktop\nomocc2.dta.xlsx")

data['occ'].value\_counts()

x = data.drop(['occ'],axis=1)

y = data['occ']

x\_train, x\_test, y\_train, y\_test = train\_test\_split(x, y, train\_size=0.8, random\_state=66)

scaler = StandardScaler()

x\_train = scaler.fit\_transform(x\_train)

x\_test = scaler.transform(x\_test)

clf = LogisticRegression()

clf.fit(x\_train,y\_train)

y\_pred = clf.predict(x\_test)

y\_pred\_prob = clf.predict\_proba(x\_test)

print("train\_score = ",clf.score(x\_train,y\_train))

print('test\_score = ',clf.score(x\_test,y\_test))

cm = confusion\_matrix(y\_test, y\_pred)

# 可视化混淆矩阵

sns.heatmap(cm, annot=True, fmt='d', cmap='Blues')

plt.title('Confusion Matrix for Multi-Class Classification')

plt.xlabel('Predicted Label')

plt.ylabel('True Label')

plt.show()

print(classification\_report(y\_test, y\_pred))

pred\_df=pd.concat([pd.DataFrame(y\_test),pd.DataFrame(y\_pred), pd.DataFrame(y\_pred\_prob)], axis=1)

pred\_df.to\_excel(r"C:\Users\26634\Desktop\ nomocc2.dta结果导出.xlsx")