参考答案

第二章

一、选择题

1．C

2．A

3．C

二、填空题

1．对象

2．一、二

3．回归、降维、预处理

第四章

一、选择题

1.C　2.B　3.C　4.A　5.C　6.A　7.A

二、填空题

1. 描述性统计法、箱型图法、正态分布法

2. 过滤法、包裹法、嵌入法

3. fillna()

4. dropna()

5. 嵌入法

三、设计题

略

第五章

一 . 简答题

略

二 . 练习题

```
# 导入相关库
from sklearn import linear_model              # 导入线性回归算法模块
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt
from sklearn.datasets import load_boston
import numpy as np
import statsmodels.api as sm
import pandas as pd
# 获取房屋数据
housedata=load_boston()
housedata.feature_names
X=housedata.data
y=housedata.target

# 构建线性回归模型 , 查看模型的检验结果 , 分析数据的多重共线性
x=sm.add_constant(X)                          # 回归方程添加一列 x0=1
model=sm.OLS(y,x)                             # 最小二乘法
result=model.fit()                            # 拟合数据
result.summary()                              # 模型描述
# 划分测试集和数据集
x_train,x_test,y_train,y_test= train_test_split(X, y, test_size= 0.25, random_state= 10010)   # 切分 25% 作为测试集。
## 分别构建线性回归模型岭回归模型，Lasso 回归模型
lr=linear_model.LinearRegression(normalize=True)
ks = np.logspace(-5, 2, 200)
ri=linear_model.RidgeCV(alphas=ks,scoring='neg_mean_squared_error',normalize=True)
la=linear_model.LassoCV(alphas=ks, normalize=True,cv=5)
lr.fit(x_train,y_train)
ri.fit(x_train,y_train)
la.fit(x_train,y_train)
# 输出回归系数
print(lr.coef_,lr.intercept_)
print(ri.coef_,ri.intercept_)
print(la.coef_,la.intercept_)
# 预测
plr_y=lr.predict(x_test)
pri_y=ri.predict(x_test)
pla_y=la.predict(x_test)
# 模型评估
print('-----LinearRegression----')
print('R2 ：{:.4f}'.format(metrics.r2_score(y_test, plr_y)))        # R2 判定系数
print(' 均方误差：{:.4f}'.format(metrics.mean_squared_error(y_test, plr_y)))
print(' 均方根误差：{:.4f}'.format(np.sqrt(metrics.mean_squared_error(y_test, plr_y))))
print(' 平均绝对误差：{:.4f}'.format(metrics.mean_absolute_error(y_test, plr_y)))
print(' 中位绝对值误差：{:.4f}'.format(metrics.median_absolute_error(y_test,plr_y)))

print('-----RidgeRegression----')
print('R2 ：{:.4f}'.format(metrics.r2_score(y_test, pri_y)))        # R2 判定系数
print(' 均方误差：{:.4f}'.format(metrics.mean_squared_error(y_test, pri_y)))
print(' 均方根误差：{:.4f}'.format(np.sqrt(metrics.mean_squared_error(y_test, pri_y))))
print(' 平均绝对误差：{:.4f}'.format(metrics.mean_absolute_error(y_test, pri_y)))
print(' 中位绝对值误差：{:.4f}'.format(metrics.median_absolute_error(y_test,pri_y)))

print('-----LassoRegression----')
print('R2 ：{:.4f}'.format(metrics.r2_score(y_test, pla_y)))        # R2 判定系数
```

```
print(' 均方误差：{:.4f}'.format(metrics.mean_squared_error(y_test, pla_y))
print(' 均方根误差：{:.4f}'.format(np.sqrt(metrics.mean_squared_error(y_test, pla_y))))
print(' 平均绝对误差：{:.4f}'.format(metrics.mean_absolute_error(y_test, pla_y)))
print(' 中位绝对值误差：{:.4f}'.format(metrics.median_absolute_error(y_test,pla_y)))
```

第七章

一 . 选择题

1.A　　2.B

二 . 填空题

1. 最大间隔的最优超平面

2. 线性核函数、多项式核函数、高斯径向基核函数和 S 型核函数。

三 . 设计题

```
from sklearn.datasets import load_breast_cancer()
from sklearn import svm
from sklearn.model_selection import cross_val_score
import time
X, y = load_breast_cancer(return_X_y=True)
models = (svm.SVC(kernel='linear'),
          svm.SVC(kernel='rbf'),
          svm.SVC(kernel='poly'),
          svm.SVC(kernel='sigmoid'))
titles = ('linear',
          'RBF',
          'polynomial (degree 3)',
          'sigmoid')

models = (clf.fit(X, y) for clf in models)
print("%30s%15s%17s"%("Kernel","Accuracy","runtime(s)" ) )
for clf,title in zip(models,titles):
    start_time = time.time()
    scores = cross_val_score(clf,X,y,cv=5)
    end_time = time.time()
    run_time = end_time – start_time
    print("%30s   %10.2f     %10.2f "%(title,scores.mean(),run_time))
```

第八章

参考代码如下：

```
from sklearn.linear_model import LogisticRegression
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import matplotlib as mpl
from sklearn.datasets import load_iris
```

```
# 导入数据
iris=load_iris()
iris.feature_names
X=iris.data[:,:]
y=iris.target
# 将原始数据集划分成训练集和测试集
X_train,X_test,y_train,y_test=train_test_split(X,y,random_state=1,test_size=0.3)
# 从 sklearn.naive_bayes 里导入朴素贝叶斯模型
from sklearn.naive_bayes import GaussianNB
# 使用默认配置初始化朴素贝叶斯模型
ga=GaussianNB()
# 利用训练数据对模型参数进行估计
ga.fit(X_train,y_train)
# 对测试样本进行类别预测，结果存储在变量 y_predict 中
y_predict=ga.predict(X_test)
print(' 预测的文章类别为：',y_predict)
# 分类结果评价
from sklearn.metrics import classification_report
print(classification_report(y_test,y_predict,target_names=iris.target_names))
# 概率模型评估指标
# 先对测试集的标签进行哑变量变换
proba= ga.predict_proba(X_test)
y_test_= y_test.copy()
y_test_ = pd.get_dummies(y_test_)
# 再计算每个类别的布利尔分数，就可以成功计算出来
from sklearn.metrics import brier_score_loss as BS
k=np.arange(0,3,1)
bs=[]
for i in k:
  b= BS(y_test_[i],proba[:,i])
  bs.append(b)
# 将每个标签类别下的布利尔分数可视化
plt.plot(k,bs)
plt.xticks(k)
plt.xlabel('Y')
plt.ylabel('BS')
plt.show()
# 对数似然函数
from sklearn.metrics import log_loss
ls=log_loss(y_test,proba)
print(' 对数似然函数为：',ls)
# 可靠性曲线
from sklearn.calibration import calibration_curve
trueproba=[]
predproba=[]
for i in k:
 tp, pp=calibration_curve(y_test_[i],proba[:,i],normalize=True,n_bins=4)
 trueproba.append(tp)
 predproba.append(pp)
plt.plot(predproba[0],trueproba[0],'o',linestyle='--')
plt.xlabel('Predproba')
plt.ylabel('Trueproba')
plt.show()
```

第十章

一、选择题

1.A

二、填空题

1. 线性激活函数、阶跃激活函数、Sigmoid 激活函数、Tanh 双曲正切激活函数和修正线性单元 ReLU 激活函数。

2. 输入层、隐藏层和输出层。

3. 无反馈前向网络、有反馈前向网络、层内有联结的前向网络、有向网。

三、设计题

```
from sklearn.neural_network import MLPClassifier
from sklearn.datasets import load_breast_cancer()
from sklearn.model_selection import cross_val_score
import matplotlib.pyplot as plt
breast_data,breast_id = load_breast_cancer (return_X_y=True)
test_score = []
active_fun = ['identity','logistic']
times_all =[]
print("%30s%30s%15s%17s"%("Activation function","The number of hidden nodes", "Accuracy",
"runtime(s)"))
for af in active_fun:
    test_score1 = []
    run_time_highest = []
    num_node = []
    for i in range(5,101,5):
        mlp = MLPClassifier(activation=af,hidden_layer_sizes=(i,))
        start_time = time.time()
        scores = cross_val_score(mlp,breast_data,breast_id,cv=5)
        end_time = time.time()
        run_time = end_time – start_time
        run_time_highest.append(run_time)
        times_all.append(run_time_highest)
        test_score1.append(scores.mean())
        test_score.append(test_score1)
print("%30s    %15d      %20.2f%%      %11.2f"%(af,(5 * (1+test_score1.index(max(test_score1)))),
(100*max(test_score1)),run_time_highest[test_score1.index(max(test_score1))]))
    plt.plot(range(5,101,5),test_score1,label=af)
plt.legend()
plt.title(" 包含一个隐藏层的 MLPClassifier 分类正确率 ")
plt.xlabel(' 隐藏节点个数 ')
plt.ylabel(' 分类正确率 ')
```