例 5.2 Python 程序代码

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from pandas import DataFrame, Series
from sklearn.cross_validation import train_test_split
from sklearn.linear_model import LinearRegression

# 创建数据集
examDict = {'学习时间': [0.50, 0.75, 1.00, 1.25, 1.50, 1.75, 1.75,
                      2.00, 2.25, 2.50, 2.75, 3.00, 3.25, 3.50, 4.00, 4.25, 4.50, 4.75,
5.00, 5.50],
            '分数': [10, 22, 13, 43, 20, 22, 33, 50, 62,
                   48, 55, 75, 62, 73, 81, 76, 64, 82, 90, 93]}

# 转换为 DataFrame 的数据格式
examDf = DataFrame(examDict)

# 绘制散点图
plt.scatter(examDf.分数, examDf.学习时间, color='b', label="Exam Data")

# 添加图的标签（x 轴，y 轴）
plt.xlabel("Hours")
plt.ylabel("Score")
# 显示图像
plt.savefig("examDf.jpg")
plt.show()

# 相关系数矩阵 r(相关系数) = x 和 y 的协方差/(x 的标准差*y 的标准差) == cov（x,y）
/σx*σy
# 相关系数 0~0.3 弱相关 0.3~0.6 中等程度相关 0.6~1 强相关
rDf = examDf.corr()
print(rDf)

# 回归方程 y = a + b*x（模型建立最佳拟合线）
# 点误差 = 实际值 - 拟合值
# 误差平方和（Sum of square error） SSE = Σ（实际值-预测值）^2
# 最小二乘法 ： 使得误差平方和最小（最佳拟合）
exam_X = examDf.loc[:, '学习时间']
exam_Y = examDf.loc[:, '分数']

# 将原数据集拆分训练集和测试集
X_train, X_test, Y_train, Y_test = train_test_split(exam_X, exam_Y, train_size=.8)
# X_train 为训练数据标签,X_test 为测试数据标签,exam_X 为样本特征,exam_y 为样本标签，
```

```
train_size 训练数据占比

print("原始数据特征:", exam_X.shape,
      ",训练数据特征:", X_train.shape,
      ",测试数据特征:", X_test.shape)

print("原始数据标签:", exam_Y.shape,
      ",训练数据标签:", Y_train.shape,
      ",测试数据标签:", Y_test.shape)

# 散点图
plt.scatter(X_train, Y_train, color="blue", label="train data")
plt.scatter(X_test, Y_test, color="red", label="test data")

# 添加图标标签
plt.legend(loc=2)
plt.xlabel("Hours")
plt.ylabel("Pass")
# 显示图像
plt.savefig("tests.jpg")
plt.show()

model = LinearRegression()

# 对于下面的模型错误我们需要把我们的训练集进行 reshape 操作来达到函数所需要的要求
# model.fit(X_train,Y_train)

# reshape 如果行数=-1 的话可以使我们的数组所改的列数自动按照数组的大小形成新的数组
# 因为 model 需要二维的数组来进行拟合但是这里只有一个特征所以需要 reshape 来转换为二维数组
X_train = X_train.values.reshape(-1, 1)
X_test = X_test.values.reshape(-1, 1)

model.fit(X_train, Y_train)

a = model.intercept_  # 截距

b = model.coef_  # 回归系数

print("最佳拟合线:截距", a, ",回归系数：", b)

# 决定系数 r 平方
# 对于评估模型的精确度
# y 误差平方和 = Σ(y 实际值 - y 预测值)^2
```

```
# y 的总波动 = Σ(y 实际值 - y 平均值)^2
# 有多少百分比的 y 波动没有被回归拟合线所描述 = SSE/总波动
# 有多少百分比的 y 波动被回归线描述 = 1 - SSE/总波动 = 决定系数 R 平方
# 对于决定系数 R 平方来说 1） 回归线拟合程度：有多少百分比的 y 波动刻印有回归线来描述(x 的波动变化)
# 2）值大小：R 平方越高，回归模型越精确(取值范围 0~1)，1 无误差，0 无法完成拟合

plt.scatter(X_train, Y_train, color='blue', label="train data")

# 训练数据的预测值
y_train_pred = model.predict(X_train)
# 绘制最佳拟合线：标签用的是训练数据的预测值 y_train_pred
plt.plot(X_train, y_train_pred, color='black', linewidth=3, label="best line")

# 测试数据散点图
plt.scatter(X_test, Y_test, color='red', label="test data")

# 添加图标标签
plt.legend(loc=2)
plt.xlabel("Hours")
plt.ylabel("Score")
# 显示图像
plt.savefig("lines.jpg")
plt.show()

score = model.score(X_test, Y_test)

print(score)
import pandas as pd
import seaborn as sns
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt
from sklearn.cross_validation import train_test_split

# 通过 read_csv 来读取我们的目的数据集
adv_data = pd.read_csv("C:/Users/Administrator/Desktop/Advertising.csv")
# 清洗不需要的数据
new_adv_data = adv_data.ix[:, 1:]
# 得到我们所需要的数据集且查看其前几列以及数据形状
print('head:', new_adv_data.head(), '\nShape:', new_adv_data.shape)

# 数据描述
print(new_adv_data.describe())
# 缺失值检验
```

```
print(new_adv_data[new_adv_data.isnull() == True].count())

new_adv_data.boxplot()
plt.savefig("boxplot.jpg")
plt.show()
##相关系数矩阵 r(相关系数) = x 和 y 的协方差/(x 的标准差*y 的标准差) == cov（x,y）/σx*σy
# 相关系数 0~0.3 弱相关 0.3~0.6 中等程度相关 0.6~1 强相关
print(new_adv_data.corr())

# 建立散点图来查看数据集里的数据分布
# seaborn 的 pairplot 函数绘制 X 的每一维度和对应 Y 的散点图。通过设置 size 和 aspect 参数来调节显示的大小和比例。
# 可以从图中看出，TV 特征和销量是有比较强的线性关系的，而 Radio 和 Sales 线性关系弱一些，Newspaper 和 Sales 线性关系更弱。
# 通过加入一个参数 kind='reg'，seaborn 可以添加一条最佳拟合直线和 95%的置信带。
sns.pairplot(new_adv_data, x_vars=['TV', 'radio', 'newspaper'], y_vars='sales', size=7, aspect=0.8, kind='reg')
plt.savefig("pairplot.jpg")
plt.show()

# 利用 sklearn 里面的包来对数据集进行划分，以此来创建训练集和测试集
# train_size 表示训练集所占总数据集的比例
X_train, X_test, Y_train, Y_test = train_test_split(new_adv_data.ix[:, :3], new_adv_data.sales, train_size=.80)

print("原始数据特征:", new_adv_data.ix[:, :3].shape,
      ",训练数据特征:", X_train.shape,
      ",测试数据特征:", X_test.shape)

print("原始数据标签:", new_adv_data.sales.shape,
      ",训练数据标签:", Y_train.shape,
      ",测试数据标签:", Y_test.shape)

model = LinearRegression()

model.fit(X_train, Y_train)

a = model.intercept_  # 截距

b = model.coef_  # 回归系数

print("最佳拟合线:截距", a, ",回归系数：", b)
# y=2.668+0.0448*TV+0.187*Radio-0.00242*Newspaper
```

```
# R 方检测
# 决定系数 r 平方
# 对于评估模型的精确度
# y 误差平方和 = Σ(y 实际值 - y 预测值)^2
# y 的总波动 = Σ(y 实际值 - y 平均值)^2
# 有多少百分比的 y 波动没有被回归拟合线所描述 = SSE/总波动
# 有多少百分比的 y 波动被回归线描述 = 1 - SSE/总波动 = 决定系数 R 平方
# 对于决定系数 R 平方来说 1） 回归线拟合程度：有多少百分比的 y 波动刻印有回归线来描述(x 的波动变化)
# 2）值大小：R 平方越高，回归模型越精确(取值范围 0~1)，1 无误差，0 无法完成拟合
score = model.score(X_test, Y_test)

print(score)

# 对线性回归进行预测

Y_pred = model.predict(X_test)

print(Y_pred)

plt.plot(range(len(Y_pred)), Y_pred, 'b', label="predict")
# 显示图像
# plt.savefig("predict.jpg")
plt.show()

plt.figure()
plt.plot(range(len(Y_pred)), Y_pred, 'b', label="predict")
plt.plot(range(len(Y_pred)), Y_test, 'r', label="test")
plt.legend(loc="upper right")  # 显示图中的标签
plt.xlabel("the number of sales")
plt.ylabel('value of sales')
plt.savefig("ROC.jpg")
plt.show()
```