《大数据分析与挖掘》
实验报告

姓　名：刘军　　学　号：20212753

学　院：计算机与数学学院　　年　级：2021 级

专业班级：计算机科学与技术三班　　指导老师：周健老师

报告评语：

成绩：

实验一：一元线性回归预测模型

一、实验目的

了解一元线性回归模型的基本原理和假设。

掌握使用一元线性回归模型对数据进行拟合和预测的方法。

理解相关系数、p-value 以及置信区间在一元线性回归模型中的含义和应用。

二、实验任务

掌握有关一元线性回归的理论知识，从中了解回归分析方法的数学模型、基本思想、方法及应用

三、实验过程

以影响预测的各因素作为自变量或解释变量 x 和因变量 y 有如下关系

$$y_i = a + bx_i + u_i \quad i = 1, 2, ..., n$$

相应于 *yi* 的估计值，与 *yi* 之差称为估计误差或残差，以表示，。显然，误差的大小是衡量估计量好坏的重要标志。我们以误差平方和最小作为衡量总误差最小的准则，并依据这一准则对参数 *a*, *b* 作出估计。令：

$$\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}\ell^2{}_i = \sum_{i=1}^{n}(y_i - \hat{a} - \hat{b}x_i)^2$$

使 *Q* 达到最小以估计出的方法称为最小二乘法(Method of Least-Squares)。由多元微分学可知，使 *Q* 达到最小的参数的的最小二乘估计量（Least-Squares Estimator of Regression Coefficient）必须满足：

$$\begin{cases} \dfrac{\partial Q}{\partial \hat{a}} = -2\sum_{i=1}^{n}(y_i - \hat{a} - \hat{b}x_i) = 0 \\ \dfrac{\partial Q}{\partial \hat{b}} = -2\sum_{i=1}^{n}(y_i - \hat{a} - \hat{b}x_i)x_i = 0 \end{cases}$$

➢ （*i*=1, 2, …, *n*）

解上述方程组得

$$\hat{a} = \bar{y} - \hat{b}\bar{x}$$

$$\hat{b} = \frac{\sum_{i=1}^{n} x_i y_i - \bar{x}\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i^2 - \bar{x}\sum_{i=1}^{n} x_i} = \frac{\sum_{i=1}^{n} x_i y_i - n\overline{xy}}{\sum_{i=1}^{n} (x_i - \bar{x})^2}$$

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

相关性检验

$$R = \frac{\sum_{i=1}^{n} x_i y_i - n\bar{x}\bar{y}}{\sqrt{\sum_{i=1}^{n} x_i^2 - n\bar{x}^2}\sqrt{\sum_{i=1}^{n} y_i^2 - n\bar{y}^2}} = \frac{l_{xy}}{\sqrt{l_{xx} l_{yy}}}$$

➢ $(-1 \leq R \leq 1)$

1991-2002 年某城市的水路货运量

序号x_i	年份	水路货运量y_i
1	1991	1659
2	1992	1989
3	1993	2195
4	1994	2255
5	1995	2329
6	1996	2375
7	1997	2364
8	1998	2354
9	1999	2418
10	2000	2534
11	2001	2568
12	2002	2835

1991-2002 年某市水路货运量一元回归计算过程

序号x_i	年份	$\bar{x}$	$(x_i-\bar{x})$	$(x_i-\bar{x})^2$	水路货运量y_i	$\bar{y}$	$(y_i-\bar{y})$	$(y_i-\bar{y})^2$
1	1991	6.5	-5.5	30.25	1659	2323	-664	440896
2	1992	6.5	-4.5	20.25	1989	2323	-334	111556
3	1993	6.5	-3.5	12.25	2195	2323	-128	16384
4	1994	6.5	-2.5	6.25	2255	2323	-68	4624
5	1995	6.5	-1.5	2.25	2329	2323	6	36
6	1996	6.5	-0.5	0.25	2375	2323	52	2704
7	1997	6.5	0.5	0.25	2364	2323	41	1681
8	1998	6.5	1.5	2.25	2354	2323	31	961
9	1999	6.5	2.5	6.25	2418	2323	95	9025
10	2000	6.5	3.5	12.25	2534	2323	211	44521
11	2001	6.5	4.5	20.25	2568	2323	245	60025
12	2002	6.5	5.5	30.25	2835	2323	512	262144

四、实验结果

程序代码

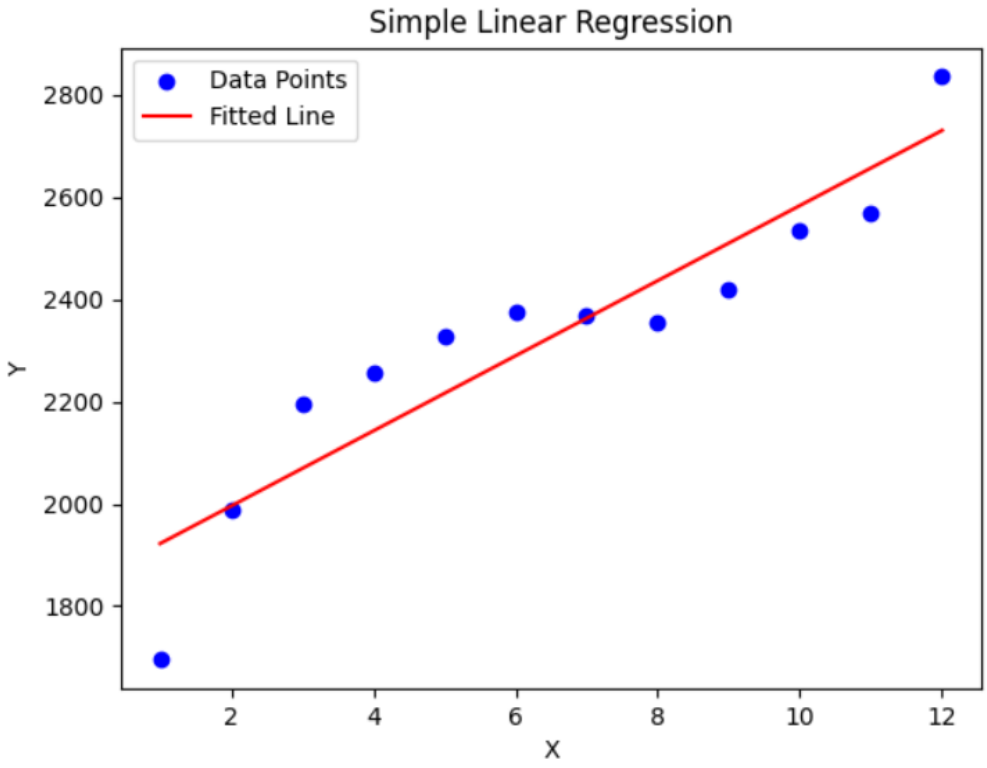
```
import numpy as np
import matplotlib.pyplot as plt
from scipy import stats
```

```
class SimpleLinearRegression:
    def __init__(self):
        self.intercept_ = None  # 截距
        self.coef_ = None  # 斜率
        self.r_value = None  # 相关系数
        self.p_value = None  # p-value
        self.ci_low = None  # 95% 置信区间下限
        self.ci_high = None  # 95% 置信区间上限

    def fit(self, X, y):
        # 计算均值
        X_mean = np.mean(X)
        y_mean = np.mean(y)

        # 计算斜率
        numerator = np.sum((X - X_mean) * (y - y_mean))
        denominator = np.sum((X - X_mean) ** 2)
        self.coef_ = numerator / denominator

        # 计算截距
        self.intercept_ = y_mean - self.coef_ * X_mean

        # 计算相关系数和 p-value
        self.r_value, self.p_value = stats.pearsonr(X, y)

        # 计算置信区间
        n = len(X)
        t_value = stats.t.ppf(0.975, n - 2)
```

```
        std_error = np.sqrt(np.sum((y - self.predict(X)) ** 2) / (n - 2))
        self.ci_low = self.coef_ - t_value * std_error / np.sqrt(np.sum((X - X_mean) ** 2))
        self.ci_high = self.coef_ + t_value * std_error / np.sqrt(np.sum((X - X_mean) ** 2))

    def predict(self, X):
        if self.intercept_ is None or self.coef_ is None:
            raise Exception("模型未拟合数据")
        return self.coef_ * X + self.intercept_


# 示例数据
X = np.array([1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12])
y = np.array([1695, 1989, 2195, 2255, 2329, 2375, 2367, 2354, 2418, 2534, 2568, 2835])

# 创建并拟合模型
model = SimpleLinearRegression()
model.fit(X, y)

# 输出相关系数和估计区间
print("相关系数:", model.r_value)
print("p-value:", model.p_value)
print("95%置信区间: [{}, {}]".format(model.ci_low, model.ci_high))

# 绘制数据点
plt.scatter(X, y, color='blue', label='Data Points')

# 绘制拟合的直线
```

```
plt.plot(X, model.predict(X), color='red', label='Fitted Line')

# 添加标签和标题
plt.xlabel('X')
plt.ylabel('Y')
plt.title('Simple Linear Regression')

# 显示图例
plt.legend()

# 显示图形
plt.show()
```

输出结果



```
D:\Users\LJ189\PycharmProjects\pythonProject\.venv\Scr
相关系数: 0.9216360621204094
p-value: 2.038208536709678e-05
95%置信区间: [51.65772376013493, 95.23738113496998]
```

五、个人总结

通过本次实验，我们深入理解了一元线性回归模型的原理和应用，并掌握了相关的数据分析方法和技巧。同时，我们对统计学中相关系数、p-value 和置信区间的概念有了更深入的理解，这些知识对于正确理解和解释模型结果至关重要。

实验二：多元线性回归模型

一、 实验目的

理解多元线性回归模型的基本原理和假设。

掌握使用多元线性回归模型对多个自变量与一个因变量之间的关系进行建模的方法。

了解多元线性回归模型参数的解释和评估方法。

二、 实验任务

使用多元线性回归模型拟合样本数据，估计模型的参数。

使用拟合好的模型进行预测，并评估预测结果的准确性和可信度。

三、 实验过程

多元回归公式

$$y_i = b_0 + b_1 x_1 + b_2 x_2 + \ldots + b_k x_k + u_i \quad i = 1, 2, \ldots, n$$

对应的样本回归模型

$$\hat{y}_i = \hat{b}_o + \hat{b}_1 x_{1i} + \hat{b}_2 x_{2i} + \quad + \hat{b}_k x_{ki} \quad (i=1, 2, \cdots, n)$$

1991-2002 年某城市的水路客运量，人口数及城市 GDP

序号	年份	水路客运量 y	市人口数 x1	城市 GDPx2
1	1991	342	520	211.9
2	1992	466	522.9	244.6
3	1993	492	527.1	325.1
4	1994	483	531.5	528.1
5	1995	530	534.7	645.1
6	1996	553	537.4	733.1
7	1997	581.5	540.4	829.7
8	1998	634.8	543.2	926.3
9	1999	656.1	545.3	1003.1
10	2000	664.4	551.5	1110.8

11	2001	688.3	554.6	1235.6
12	2002	684.4	557.93	1406

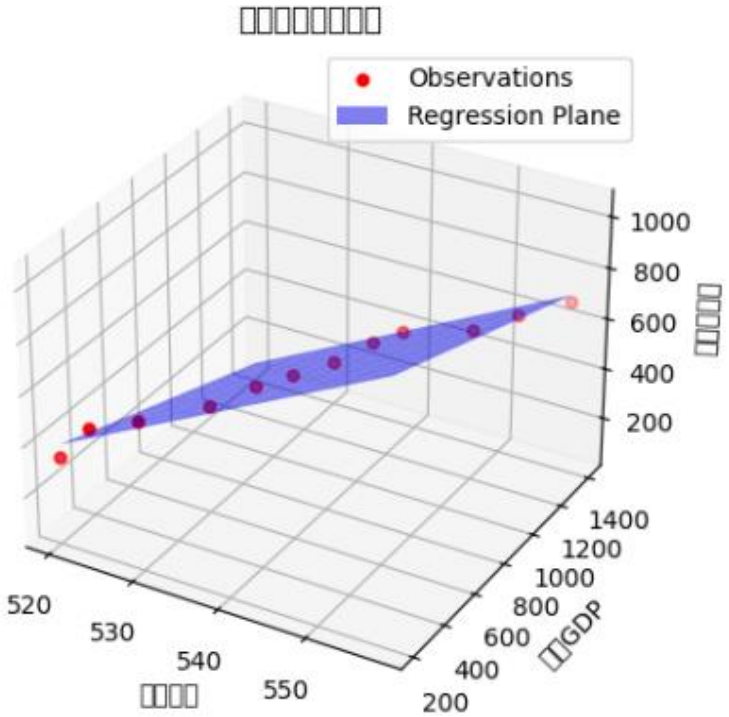
四、 实验结果

实验代码

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D


# 原始数据
data = np.array([
    [1991, 342, 520, 211.9],
    [1992, 466, 522.9, 244.6],
    [1993, 492, 527.1, 325.1],
    [1994, 483, 531.5, 528.1],
    [1995, 530, 534.7, 645.1],
    [1996, 553, 537.4, 733.1],
    [1997, 581.5, 540.4, 829.7],
    [1998, 634.8, 543.2, 926.3],
    [1999, 656.1, 545.3, 1003.1],
    [2000, 664.4, 551.5, 1110.8],
    [2001, 688.3, 554.6, 1235.6],
    [2002, 684.4, 557.93, 1406]
])


# 将数据分为自变量(X)和因变量(y)
X = data[:, 2:]  # 自变量包括市人口数和城市 GDP
y = data[:, 1]   # 因变量是水路客运量
```

```
# 计算参数 (β = (X^T * X)^(-1) * X^T * y)
X = np.column_stack((np.ones(X.shape[0]), X))
beta = np.linalg.inv(X.T.dot(X)).dot(X.T).dot(y)

# 绘制散点图和回归平面
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

x1 = data[:, 2]
x2 = data[:, 3]
ax.scatter(x1, x2, y, color='r', marker='o', label='Observations')

x1_surf = np.linspace(min(x1), max(x1), 10)                # 绘制回归
平面
x2_surf = np.linspace(min(x2), max(x2), 10)
x1_surf, x2_surf = np.meshgrid(x1_surf, x2_surf)
y_surf = beta[0] + beta[1] * x1_surf + beta[2] * x2_surf
ax.plot_surface(x1_surf,  x2_surf,  y_surf,  color='b',  alpha=0.5,
label='Regression Plane')

ax.set_xlabel('市人口数')
ax.set_ylabel('城市 GDP')
ax.set_zlabel('水路客运量')

plt.title('多元线性回归模型')
plt.legend()
plt.show()
```

输出结果



五、 实验总结

在这个实验中，使用了 12 年的市人口数、城市 GDP 和水路客运量数据，并使用多元线性回归模型进行分析。通过最小二乘法，建立了一个模型来描述市人口数和城市 GDP 对水路客运量的影响关系，并通过绘制数据的散点图和回归平面进行可视化。结果显示，市人口数和城市 GDP 对水路客运量都呈现出正向影响，即随着市人口数和城市 GDP 的增加，水路客运量也增加。

实验三：非线性回归模型

一、 实验目的

学习非线性回归模型

二、 实验任务

从五种非线性回归模型中选两种完成实验

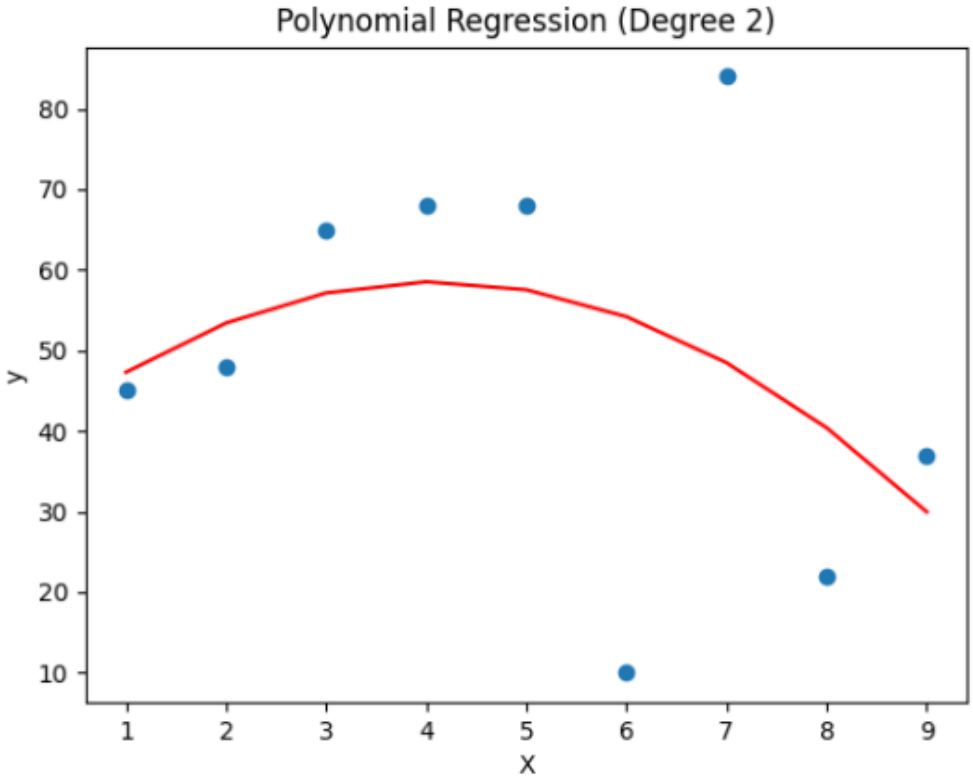
三、 实验内容

幂函数模型

```
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
import matplotlib.pyplot as plt

# 生成随机数据
np.random.seed(0)
X = np.arange(1, 10, 1).reshape(-1, 1)
y = np.random.randint(1, 100, size=(X.shape[0],))

# 使用多项式特征将输入数据转换为幂函数形式
poly_features = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly_features.fit_transform(X)

# 创建线性回归模型
model = LinearRegression()

# 拟合模型
model.fit(X_poly, y)

# 打印模型系数
```

```
print("Model Coefficients:", model.coef_)

# 绘制数据和拟合的幂函数曲线
plt.scatter(X, y)
plt.plot(X, model.predict(X_poly), color='red')
plt.title('Polynomial Regression (Degree 2)')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

输出结果

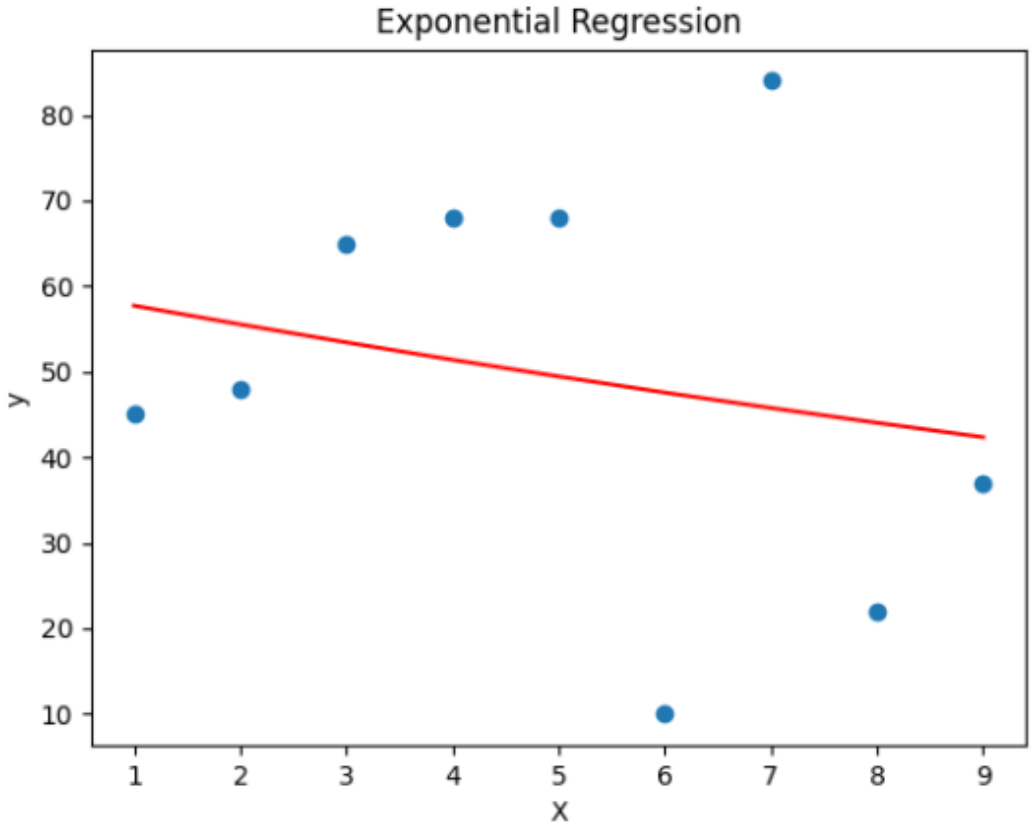


指数函数模型

```
import numpy as np
from scipy.optimize import curve_fit
import matplotlib.pyplot as plt

# 定义指数函数模型
```

```
def exponential_func(x, a, b):
    return a * np.exp(b * x)

# 生成随机数据
np.random.seed(0)
X = np.arange(1, 10)
y = np.random.randint(1, 100, size=X.shape[0])

# 拟合指数函数模型
params, covariance = curve_fit(exponential_func, X, y)

# 提取拟合参数
a, b = params

# 打印拟合参数
print("a:", a)
print("b:", b)

# 绘制数据和拟合的指数函数曲线
plt.scatter(X, y)
plt.plot(X, exponential_func(X, a, b), color='red')
plt.title('Exponential Regression')
plt.xlabel('X')
plt.ylabel('y')
plt.show()
```

输出结果




```
D:\Users\LJ189\PycharmProjec
a: 60.00846078708002
b: -0.03869849426280258
```

四、 实验总结

这次实验利用 Scikit-learn 和 SciPy 库，展示了如何拟合幂函数模型和指数函数模型。通过随机生成的数据，我们使用多项式回归实现了幂函数模型的拟合，同时使用曲线拟合方法实现了指数函数模型的拟合。这次实验帮助我们了解了如何利用不同的工具和技术来处理非线性回归问题，并通过数据可视化来展示拟合结果。

实验四：趋势外推模型

一、 实验目的

学习趋势外推模型

二、 实验任务

三种趋势外推模型任选一种实现

三、 实验内容

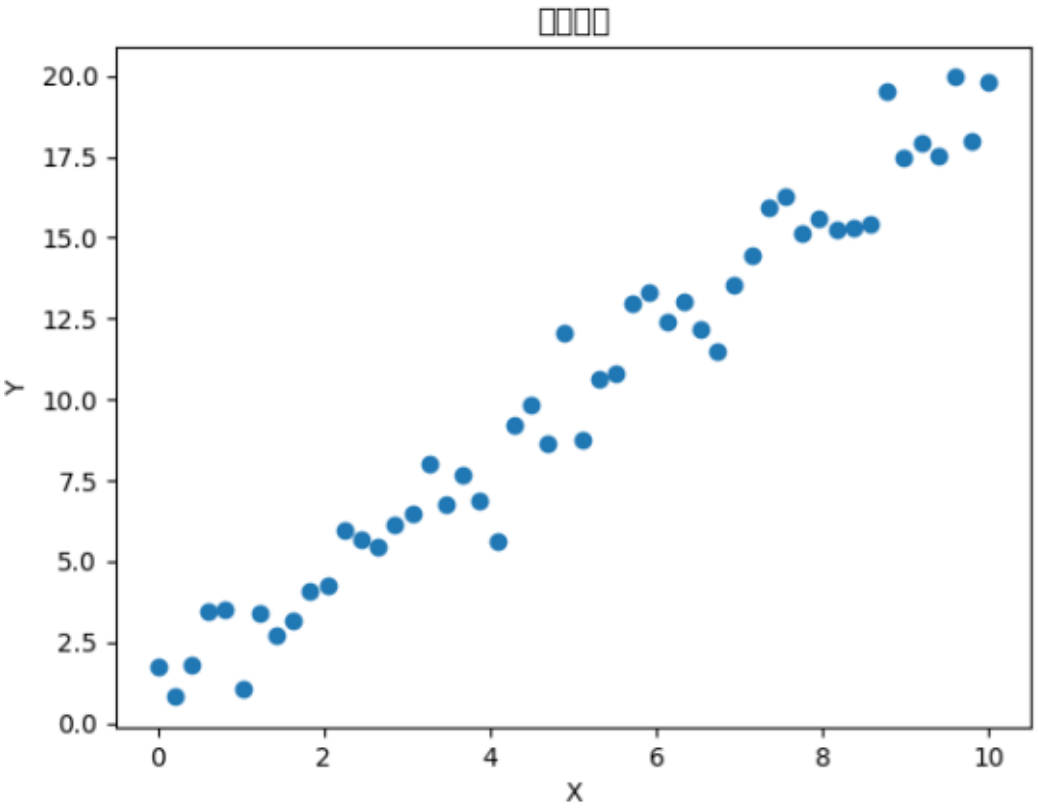
实验代码

```
import numpy as np
import matplotlib.pyplot as plt

# 生成示例数据
np.random.seed(0)  # 设置随机种子以便结果可复现
num_samples = 50  # 数据点数量
x = np.linspace(0, 10, num_samples)
y = 2 * x + np.random.normal(loc=0, scale=1, size=num_samples)  # 添加噪声

# 绘制示例数据
plt.scatter(x, y)
plt.xlabel('X')
plt.ylabel('Y')
plt.title('示例数据')
plt.show()

# 皮尔预测模型
def pearson_correlation(x, y):
    """
    计算两个变量之间的皮尔逊相关系数
```

```
    """
    mean_x = np.mean(x)
    mean_y = np.mean(y)
    std_x = np.std(x)
    std_y = np.std(y)
    correlation = np.mean((x - mean_x) * (y - mean_y)) / (std_x * std_y)
    return correlation

def pear_forecast_model(x_train, y_train, x_test):
    """
    皮尔预测模型：根据训练数据预测测试数据的结果
    """
    predictions = []
    for test_point in x_test:
        correlations = []
        for i, train_point in enumerate(x_train):
            correlation = pearson_correlation(train_point, y_train[i])
            correlations.append(correlation)
        max_corr_index = np.argmax(correlations)
        prediction = y_train[max_corr_index][-1]  # 使用最相关的训练数据的最后一个点作为预测值
        predictions.append(prediction)
    return predictions

# 将数据转换为适合模型的格式
x_train = x.reshape(-1, 1)
y_train = y.reshape(-1, 1)
x_test = np.array([[10]])  # 需要预测的未来数据点
```

```
# 使用皮尔预测模型预测结果
predictions = pear_forecast_model(x_train, y_train, x_test)
print("预测结果:", predictions[0])
```

实验结果




```
预测结果: 1.7640523459676644
```

四、 实验总结

在这个实验中，我们使用了皮尔预测模型来预测具有线性关系的示例数据的未来结果。首先，我们生成了一组示例数据，其中 X 和 Y 之间存在线性关系，并且添加了一些噪声。然后，我们实现了皮尔预测模型，该模型根据历史数据中变量之间的相关性来预测未来的结果。最后，我们使用该模型来预测了一个未来数据点的结果。通过这个实验，我们展示了皮尔预测模型的基本原理和应用，以及如何使用 Python 实现该模型进行预测。