1、问题描述

根据给定数据，选定模型进行拟合，利用平方误差：mean squared error(MSE)作为模型的度量，重点指出最后模型的测试误差。

2、数据集

数据分成两个部分：complex_nonlinear_data.csv 训练数据用来训练模型，用 new_complex_nonlinear_data.csv 测试数据验证模型的性能。

3、数据预处理

3.1、Python 库的选择：

pandas:读取 csv 文件数据，剔除异常值

numpy: 在拟合模型中，将数据转换为 numpy 数组进行高效的数学运算。

seaborn:用于生成箱状图，分析数据的异常值。

matplotlib:用于数据的可视化分析。

sklearn:用于机器学习，训练回归模型并进行相应的评估。

3.2、数据的导入

```
file = "complex_nonlinear_data.csv"
df = pd.read_csv(file)
data = df.iloc[:, 0:2]
train_x = data["x"]
train_y = data["y_complex"]
```

3.3、异常值的处理

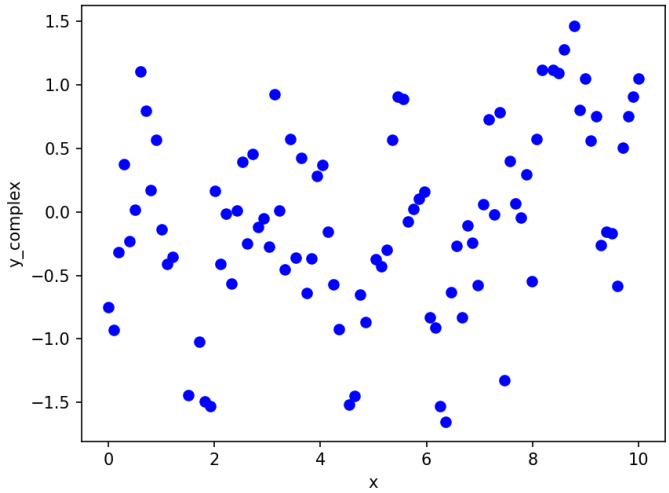

图 1 complex_nonlinear_data.csv 数据的散点图

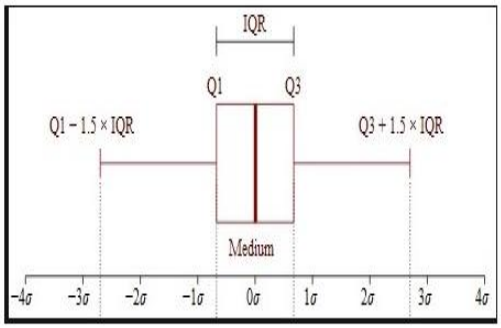

图 2 IQR 箱状图理论示意图

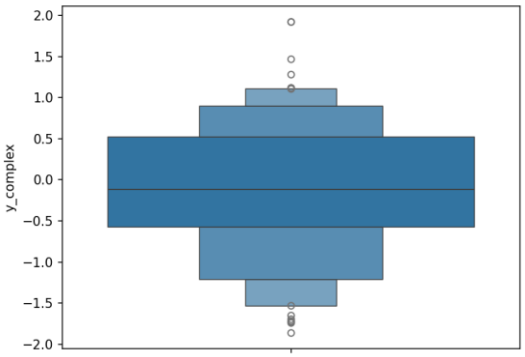

图 3 y_complex 的箱状图

根据对相应散点图以及箱状图的分析可以发现，在$[Q1-IQR，Q3+IQR]$的范围以外存在着一定的离群值数据。

利用四分位距方法（interquartile range，IQR）对异常数据进行处理，剔除异常值，并对 x 进行 numpy 数据转化：

```
# IQR
sns.boxenplot(train_y)
Q1 = np.percentile(train_y, 25, interpolation="midpoint")
Q3 = np.percentile(train_y, 75, interpolation="midpoint")
IQR = Q3 - Q1
upper_bound = Q3 + 1 * IQR
lower_bound = Q1 - 1 * IQR
upper_indices = np.where(train_y > upper_bound)[0]
lower_indices = np.where(train_y < lower_bound)[0]
train_x = train_x.drop(list(np.concatenate([upper_indices,
lower_indices])))
train_y = train_y.drop(list(np.concatenate([upper_indices,
lower_indices])))
train_x = train_x.dropna().reset_index(drop=True)
train_y = train_y.dropna().reset_index(drop=True)
X = train_x.values.reshape(-1, 1)
```

3.4、K 折交叉验证：

将原始训练数据集划分为训练集和测试集，避免了为了追求高准确率而在训练集上产生过拟合，从而使得模型在样本外的数据上预测准确率高。

图 4 十折交叉验证原理图

由于数据规模较小，本模型选用十折交叉验证进行分析，此时训练集和测试集的数据比为 9:1。

```
Kfold = KFold(n_splits=10, shuffle=True)
for train_index, test_index in Kfold.split(X):
    X_train, X_test = X[train_index], X[test_index]
    Y_train, Y_test = train_y[train_index], train_y[test_index]
```

4、模型的选择

根据对散点图的分析，以及图中出现多次类似谷峰谷底的数据情况，该数据呈现明显的非线性关系，因此选用多项式模型进行数据的拟合：

```
def PolynomialRegression(degree):
    return Pipeline(
        [
            ("poly", PolynomialFeatures(degree)),
            ("std_scaler", StandardScaler()),
            ("line_reg", LinearRegression()),
        ]
    )
```

5、模型的训练与评估

5.1、学习曲线的建立和模型训练

通过对多项式次数 degree 进行一定范围的搜索，得出符合一定规律的 Mse_train 和 Mse_test 曲线：

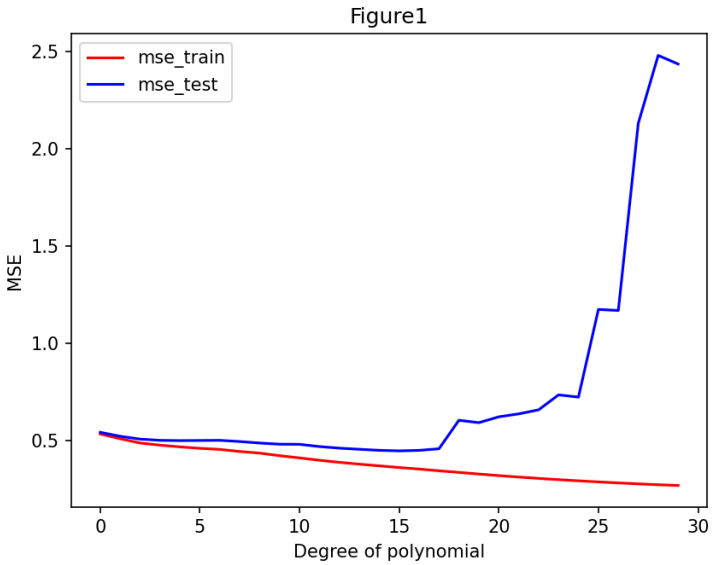

图 5 学习曲线

由曲线观察分析，当 Degree of polynomoial 逐渐增加时，mse_train 曲线一直下降,而 mse_test 曲线先下降后明显上升，说明出现过拟合现象。因此本模型选择 Mse_test 最小的时刻的 Degree of polynomial = 15 作为多项式回归模型最优的 degree 参数。

同时获得训练数据集最优的 Mse= 0.27359363929663677。

```
for degree in range(30):
    for train_index, test_index in Kfold.split(X):
        X_train, X_test = X[train_index], X[test_index]
        Y_train, Y_test = train_y[train_index], train_y[test_index]
        # 回归模型
        poly_reg = PolynomialRegression(degree)
        poly_reg.fit(X_train, Y_train)
        y_predict_train = poly_reg.predict(X_train)
        y_predict_test = poly_reg.predict(X_test)
        scores_train.append(mean_squared_error(Y_train,
y_predict_train))
        scores_test.append(mean_squared_error(Y_test, y_predict_test))
    mse_test.append(np.mean(scores_test))
    mse_train.append(np.mean(scores_train))
# 最优次项
print(mse_test.index(min(mse_test)))
```

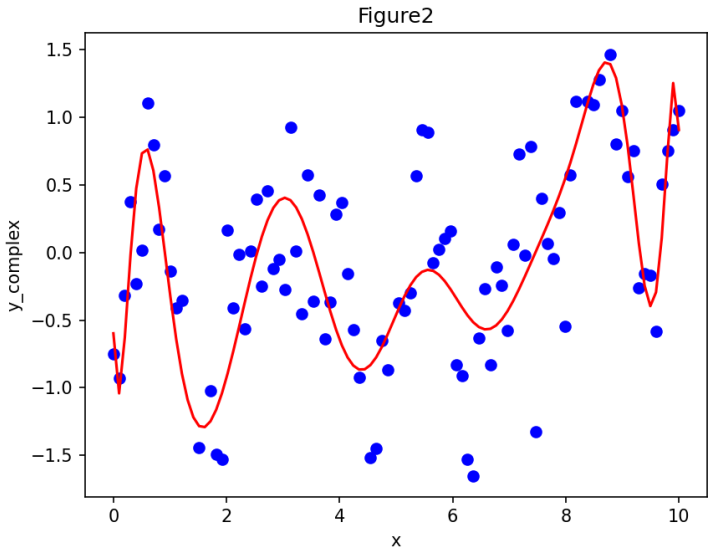

图 6 训练数据的拟合曲线

```
# 回归模型
        poly_reg = PolynomialRegression(degree)
        poly_reg.fit(X_train, Y_train)
        y_predict_train = poly_reg.predict(X_train)
        y_predict_test = poly_reg.predict(X_test)
```

5.2、模型的评估

将最终的测试数据带入训练好的模型之中：

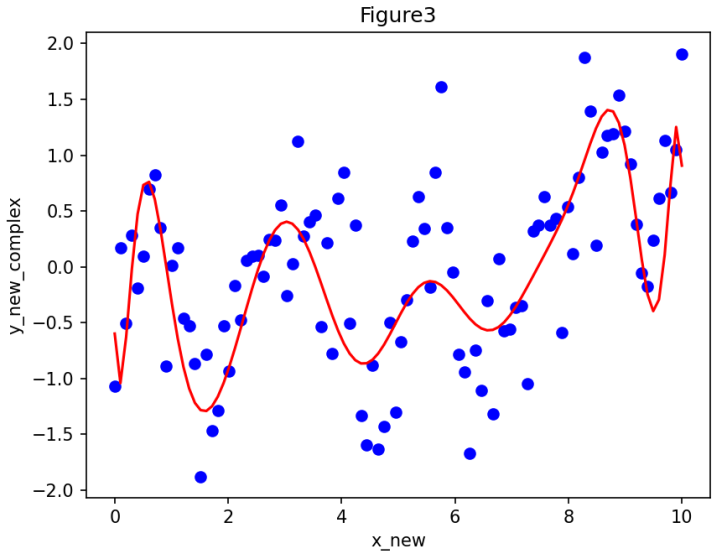

图 7 测试数据的拟合曲线

最终所得测试数据的 Mse= 0.3384791555416797。由测试数据的回归曲线可以看出，degree=15 时的多项式回归模型可以较好的回归出数据的趋势，Mse 的范围也比较精确。由于训练和测试数据存在较大的噪声波动等影响，degree=15 时的多项式回归模型已经可以较好的拟合出相应的数据。

```
# 训练测试
poly_reg = PolynomialRegression(mse_test.index(min(mse_test)))
```

```
poly_reg.fit(data["x"].values.reshape(-1, 1), data["y_complex"])
Y_predict = poly_reg.predict(data["x"].values.reshape(-1, 1))
print(mean_squared_error(data["y_complex"], Y_predict))

Y_new_predict = poly_reg.predict(test_x.values.reshape(-1, 1))
print(mean_squared_error(test_y, Y_new_predict))
```

6、总结归纳

本次作业，通过对数据集提取、数据清洗和处理，利用十折交叉验证对训练数据进行验证，避免了过拟合现象的发生。同时根据对数据的分析，选用多项式模型进行回归，并对测试数据进行验证，鉴于数据存在一定的噪声误差，利用 Mse 对所得回归模型进行评估时，本次模型验证较为良好，具有一定的参考意义。