A Masterclass in Regression Analysis: From Fundamentals to Advanced Applications

Rajendra K Pandey
rajendrapandey.95@live.com

Contents

1 Introduction to Regression Analysis **3**
1.1 Core Concept Explanation . . . 3
1.1.1 What is Regression? . . . 3
1.1.2 Purpose of Regression Analysis . . . 3
1.1.3 Intuitive Analogies for Beginners . . . 3
1.2 Mathematical Foundations . . . 4
1.2.1 Basic Notations . . . 4
1.2.2 Core Statistical Principles . . . 4
1.3 Role in Machine Learning . . . 5
1.4 Mini-Quiz . . . 5

2 Simple Linear Regression (SLR) **5**
2.1 Core Concept Explanation . . . 5
2.1.1 Definition of Simple Linear Regression . . . 5
2.1.2 The Line of Best Fit . . . 6
2.2 Mathematical Foundations . . . 6
2.2.1 Equation of SLR . . . 6
2.2.2 Derivation of OLS Estimators . . . 6
2.2.3 Assumptions for SLR . . . 6
2.3 Solved Example . . . 6
2.4 Visual Explanations . . . 7
2.5 Python Code Implementation . . . 7
2.5.1 Using scikit-learn . . . 7
2.5.2 Using statsmodels . . . 8
2.6 Real-World Dataset Example . . . 8
2.7 Model Evaluation . . . 9

2.8 Common Pitfalls and Fixes 10

3 Multiple Linear Regression (MLR) 10

3.1 Core Concept Explanation 10

3.1.1 Definition of MLR 10

3.1.2 Interpreting Coefficients 10

3.2 Mathematical Foundations 10

3.2.1 Equation of MLR 10

3.2.2 OLS in Matrix Form 10

3.3 Solved Example 10

3.4 Python Code Implementation 10

3.5 Model Evaluation 12

4 Polynomial Regression 12

4.1 Core Concept Explanation 12

4.2 Mathematical Foundations 12

4.3 Solved Example 13

4.4 Python Code Implementation 13

5 Ridge Regression (L2 Regularization) 14

5.1 Core Concept Explanation 14

5.2 Mathematical Foundations 14

1 Introduction to Regression Analysis

1.1 Core Concept Explanation

1.1.1 What is Regression?

Regression analysis is a cornerstone of statistical modeling and machine learning, designed to understand and quantify the relationship between a dependent variable, often denoted as Y, and one or more independent variables, denoted as $X_1, X_2, \ldots, X_p$. The primary goal is to model this relationship to predict future outcomes or understand the strength and nature of these associations.

For example, to predict a student's exam score (Y), factors such as hours spent studying (X_1), previous grades (X_2), and attendance (X_3) can be used as predictors. The relationship is often visualized as fitting a line or curve through a scatter plot, representing the "best fit" that summarizes the trend.

1.1.2 Purpose of Regression Analysis

Regression analysis serves multiple purposes:

- **Understanding Relationships**: Identifies and quantifies the strength and direction of relationships between variables.
- **Prediction and Forecasting**: Predicts the dependent variable for new values of independent variables.
- **Determining Influence/Impact**: Compares the effects of variables measured on different scales.
- **Causal Inference (with caution)**: Provides evidence supporting causal hypotheses in carefully designed studies.

1.1.3 Intuitive Analogies for Beginners

- **Adjusting Knobs**: Independent variables are like knobs on a machine, with regression coefficients indicating their sensitivity.
- **Drawing the Best Line Through Stars**: Regression draws a line through data points to represent the underlying trend.
- **Predicting Weight from Height**: Models the general trend that taller people tend to weigh more, though other factors influence weight.

1.2 Mathematical Foundations

1.2.1 Basic Notations

- Y: Dependent variable (outcome to predict).
- $X_1, X_2, \ldots, X_p$: Independent variables (predictors).
- y_i: Observed value of Y for the i-th observation.
- x_{ij}: Value of the j-th predictor for the i-th observation.
- β_0: Intercept, expected Y when all $X_j = 0$.
- $\beta_1, \beta_2, \ldots, \beta_p$: Regression coefficients, representing the change in Y per unit change in X_j.
- ϵ_i: Error term, capturing unmodeled factors.
- $\hat{y}_i$: Predicted value of Y for the i-th observation.
- n: Number of observations.
- p: Number of predictors.

The general linear regression model is:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

For the i-th observation:

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + \cdots + \beta_p x_{ip} + \epsilon_i$$

The predicted value is:

$$\hat{y}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{i1} + \hat{\beta}_2 x_{i2} + \cdots + \hat{\beta}_p x_{ip}$$

1.2.2 Core Statistical Principles

- **Relationship Modeling**: Quantifies statistical relationships, with ϵ acknowledging imperfections.
- **Ordinary Least Squares (OLS)**: Minimizes the Residual Sum of Squares (RSS):

$$\text{RSS} = \sum_{i=1}^{n} (y_i - \hat{y}_i)^2 = \sum_{i=1}^{n} \epsilon_i^2$$

- **Assumptions of Linear Regression**:
  1. Linearity in parameters.
  2. Independence of errors.
  3. Homoscedasticity (constant error variance).
  4. Normality of errors for inference.
  5. No perfect multicollinearity (for multiple regression).
  6. Strict exogeneity: $\text{Cov}(X_j, \epsilon) = 0$.
- **Gauss-Markov Theorem**: Under CLRM assumptions, OLS estimators are BLUE (Best Linear Unbiased Estimators).

1.3 Role in Machine Learning

Regression is a fundamental supervised learning technique used for:

- Predicting continuous target variables.
- Assessing feature importance via coefficients.
- Serving as a baseline for complex models.
- Forming the foundation for models like logistic regression.

1.4 Mini-Quiz

1. **Define dependent and independent variables.**
2. **Primary goal of OLS?**
3. **List three CLRM assumptions.**
4. **What does the Gauss-Markov theorem state?**

2 Simple Linear Regression (SLR)

2.1 Core Concept Explanation

2.1.1 Definition of Simple Linear Regression

SLR models the linear relationship between one independent variable (X) and one dependent variable (Y):

$$Y = \beta_0 + \beta_1 X + \epsilon$$

The goal is to find the line of best fit using OLS.

2.1.2 The Line of Best Fit

The OLS method minimizes:

$$\text{RSS} = \sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \sum_{i=1}^{n}(y_i - (\hat{\beta}_0 + \hat{\beta}_1 x_i))^2$$

2.2 Mathematical Foundations

2.2.1 Equation of SLR

$$Y = \beta_0 + \beta_1 X + \epsilon$$

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X$$

2.2.2 Derivation of OLS Estimators

The OLS estimators are:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2} = \frac{S_{xy}}{S_{xx}}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x}$$

2.2.3 Assumptions for SLR

Same as CLRM, applied to one predictor.

2.3 Solved Example

Consider the dataset:

Hours Studied (X)	Exam Score (Y)
2	65
3	70
4	75
5	85
6	90

Table 1: Dataset for SLR Example

Step 1: Calculate sums and means

$$n = 5, \quad \sum x_i = 20, \quad \sum y_i = 385, \quad \bar{x} = 4, \quad \bar{y} = 77$$

$$\sum x_i^2 = 90, \quad \sum x_i y_i = 1605$$

Step 2: Calculate $\hat{\beta}_1$

$$\hat{\beta}_1 = \frac{5 \cdot 1605 - 20 \cdot 385}{5 \cdot 90 - 20^2} = \frac{325}{50} = 6.5$$

Step 3: Calculate $\hat{\beta}_0$

$$\hat{\beta}_0 = 77 - 6.5 \cdot 4 = 51$$

Regression Equation

$$\hat{y} = 51 + 6.5 \cdot \text{Hours Studied}$$

2.4 Visual Explanations

- **Scatter Plot with Regression Line**: Displays data points with the line $\hat{y} = 51 + 6.5X$.
- **Residual Plot**: Residuals vs. x_i or $\hat{y}_i$, ideally showing random scatter.
- **Normal Q-Q Plot**: Checks normality of residuals.

2.5 Python Code Implementation

2.5.1 Using scikit-learn

```
import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
import matplotlib.pyplot as plt

# Dataset
data = {'Hours_Studied': [2, 3, 4, 5, 6], 'Exam_Score': [65, 70,
    75, 85, 90]}
df = pd.DataFrame(data)
X = df[['Hours_Studied']]
y = df['Exam_Score']

# Fit model
model = LinearRegression()
model.fit(X, y)

# Coefficients
print(f"Intercept (beta_0): {model.intercept_}")
print(f"Slope (beta_1): {model.coef_[0]}")
```

```
# Predictions
y_pred = model.predict(X)

# Plotting
plt.scatter(X, y, color='blue', label='Actual Scores')
plt.plot(X, y_pred, color='red', label='Regression Line')
plt.xlabel('Hours Studied')
plt.ylabel('Exam Score')
plt.title('Simple Linear Regression')
plt.legend()
plt.show()

# Residual Plot
residuals = y - y_pred
plt.scatter(y_pred, residuals)
plt.axhline(y=0, color='red', linestyle='--')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.title('Residual Plot')
plt.show()
```

2.5.2 Using statsmodels

```
import statsmodels.api as sm

# Add constant
X_sm = sm.add_constant(X)

# Fit model
model_sm = sm.OLS(y, X_sm).fit()
print(model_sm.summary())
```

2.6 Real-World Dataset Example

Using the `diabetes` dataset from scikit-learn:

```
from sklearn.datasets import load_diabetes
from sklearn.model_selection import train_test_split

# Load data
```

```
diabetes = load_diabetes()
X_diabetes = diabetes.data[:, np.newaxis, 2] # BMI
y_diabetes = diabetes.target

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_diabetes,
    y_diabetes, test_size=0.2, random_state=42)

# Fit model
slr_model = LinearRegression()
slr_model.fit(X_train, y_train)

# Coefficients
print(f"Intercept: {slr_model.intercept_}")
print(f"Slope: {slr_model.coef_[0]}")

# Plotting
y_pred = slr_model.predict(X_test)
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.xlabel('BMI (Normalized)')
plt.ylabel('Disease Progression')
plt.title('SLR on Diabetes Dataset')
plt.legend()
plt.show()

# Statsmodels
X_train_sm = sm.add_constant(X_train)
model_sm = sm.OLS(y_train, X_train_sm).fit()
print(model_sm.summary())
```

2.7 Model Evaluation

- **MSE**: $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- **RMSE**: $\sqrt{\text{MSE}}$
- **MAE**: $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$
- **R-squared**: $R^2 = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$

2.8 Common Pitfalls and Fixes

- **Non-linearity**: Transform variables or use polynomial regression.
- **Outliers**: Investigate and consider robust regression.
- **Heteroscedasticity**: Use transformations or WLS.
- **Autocorrelation**: Include lagged variables or use time series models.

3 Multiple Linear Regression (MLR)

3.1 Core Concept Explanation

3.1.1 Definition of MLR

MLR models the relationship between Y and multiple predictors $X_1, X_2, \ldots, X_p$:

$$Y = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_p X_p + \epsilon$$

3.1.2 Interpreting Coefficients

β_j represents the change in Y per unit change in X_j, holding other predictors constant.

3.2 Mathematical Foundations

3.2.1 Equation of MLR

$$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 X_1 + \hat{\beta}_2 X_2 + \cdots + \hat{\beta}_p X_p$$

3.2.2 OLS in Matrix Form

$$\mathbf{Y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\epsilon}$$

$$\hat{\beta} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{Y}$$

3.3 Solved Example

Estimated equation:

$$\hat{y} = 50 + 0.15 \cdot \text{Size} - 2.5 \cdot \text{Age}$$

3.4 Python Code Implementation

Price (Y, \$000)	Size (X_1, sq ft)	Age (X_2, years)
300	1500	10
450	2000	5
250	1200	20
550	2500	2
350	1800	15

Table 2: MLR Example Dataset

```
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.datasets import fetch_california_housing
import statsmodels.api as sm
from statsmodels.stats.outliers_influence import
   variance_inflation_factor
import matplotlib.pyplot as plt
import seaborn as sns

# Load data
housing = fetch_california_housing()
X_df = pd.DataFrame(housing.data, columns=housing.feature_names)
y_series = pd.Series(housing.target, name='MedHouseValue')
X_selected = X_df[['MedInc', 'HouseAge', 'AveRooms', 'AveBedrms',
   'Population']]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X_selected,
   y_series, test_size=0.2, random_state=42)

# Fit model
mlr_model = LinearRegression()
mlr_model.fit(X_train, y_train)

# Coefficients
print("Intercept:", mlr_model.intercept_)
for feature, coef in zip(X_selected.columns, mlr_model.coef_):
    print(f"{feature}: {coef:.4f}")

# VIF
X_with_constant = sm.add_constant(X_selected)
```

```
vif_data = pd.DataFrame()
vif_data["feature"] = X_with_constant.columns
vif_data["VIF"] =
   [variance_inflation_factor(X_with_constant.values, i) for i in
   range(X_with_constant.shape[1])]
print(vif_data)

# Correlation Heatmap
plt.figure(figsize=(8, 6))
sns.heatmap(X_selected.corr(), annot=True, cmap='coolwarm',
   fmt=".2f")
plt.title('Correlation Heatmap')
plt.show()

# Statsmodels
X_train_sm = sm.add_constant(X_train)
sm_model = sm.OLS(y_train, X_train_sm).fit()
print(sm_model.summary())
```

3.5 Model Evaluation

- **Adjusted R-squared**: Penalizes for additional predictors.
- **F-statistic**: Tests if at least one $\beta_j \neq 0$.
- **VIF**: Detects multicollinearity.

4 Polynomial Regression

4.1 Core Concept Explanation

Polynomial regression models non-linear relationships using:

$$Y = \beta_0 + \beta_1 X + \beta_2 X^2 + \cdots + \beta_d X^d + \epsilon$$

4.2 Mathematical Foundations

The model is linear in parameters, treated as MLR with features $X, X^2, \ldots, X^d$.

4.3 Solved Example

For points (0,1), (1,4), (2,9):

$$\hat{y} = 1 + 2X + X^2$$

4.4 Python Code Implementation

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

# Generate data
np.random.seed(0)
X_poly_ex = 6 * np.random.rand(100, 1) - 3
y_poly_ex = 0.8 * X_poly_ex**2 + 0.9 * X_poly_ex + 2 +
   np.random.randn(100, 1)
y_poly_ex = y_poly_ex.ravel()

plt.figure(figsize=(18, 5))
for i, degree in enumerate([1, 2, 15]):
    ax = plt.subplot(1, 3, i + 1)
    model = Pipeline([
        ("poly_features", PolynomialFeatures(degree=degree,
           include_bias=False)),
        ("lin_reg", LinearRegression())
    ])
    model.fit(X_poly_ex, y_poly_ex)
    y_pred = model.predict(X_poly_ex)
    X_plot = np.linspace(X_poly_ex.min(), X_poly_ex.max(),
       100).reshape(-1, 1)
    y_plot = model.predict(X_plot)
    plt.plot(X_plot, y_plot, color='r', label='Model')
    plt.scatter(X_poly_ex, y_poly_ex, color='b', label='Samples')
    plt.xlabel('X')
    plt.ylabel('Y')
    plt.legend()
    plt.title(f'Degree {degree}')
plt.show()
```

5 Ridge Regression (L2 Regularization)

5.1 Core Concept Explanation

Ridge regression addresses multicollinearity and overfitting by adding an L2 penalty:

$$J(\boldsymbol{\beta}) = (\mathbf{Y} - \mathbf{X}\boldsymbol{\beta})^T(\mathbf{Y} - \mathbf{X}\boldsymbol{\beta}) + \lambda\boldsymbol{\beta}^T\boldsymbol{\beta}$$

5.2 Mathematical Foundations

$$\hat{\beta}_{\text{Ridge}} = (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{Y}$$