Lab 6: Polynomial Regression

Objective:

This lab aims to understand and implement Polynomial Regression, an extension of linear regression that models the relationship between independent and dependent variables as an nth-degree polynomial. You will:

1. Learn the mathematical formulation of Polynomial Regression.
2. Explore a dataset to understand its trends.
3. Implement Polynomial Regression using Python.
4. Compare Polynomial Regression with Linear Regression to evaluate performance.

Prerequisites:

- **Python Libraries:** Pandas, NumPy, Matplotlib, Seaborn, Scikit-learn.
- **Conceptual Understanding:** Linear Regression, polynomial equations, basic machine learning principles, and visualization.

Equation of Polynomial Regression:

The general equation for Polynomial Regression is:

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \cdots + \beta_n x^n$$

y: Dependent variable
x: Independent variable
β0,β1,β2,…,βn: Coefficients
n: Degree of the polynomial

In Polynomial Regression, the coefficients β are determined using least squares or maximum likelihood estimation. The degree n determines the complexity of the model.

Step 1: Import Required Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import PolynomialFeatures
from sklearn.linear_model import LinearRegression
```

Step 2: Load the Dataset

```
# Load dataset
df_sal = pd.read_csv('Position_Salaries.csv')
df_sal.head()
```

Step 3: Data Analysis

```
# Describe data
df_sal.describe()

# Salary distribution
plt.title('Salary Distribution Plot')
sns.distplot(df_sal['Salary'])
plt.show()


# Scatter plot
plt.scatter(df_sal['Level'], df_sal['Salary'], color='lightcoral')
plt.title('Salary vs Level')
plt.xlabel('Level')
plt.ylabel('Salary')
plt.box(False)
plt.show()
```

Step 4: Split the Dataset into Independent and Dependent Variables

```
# Splitting into independent (X) and dependent (y) variables
X = df_sal.iloc[:, 1:-1].values  # Independent variable
y = df_sal.iloc[:, -1].values  # Dependent variable
```

Step 5: Train the Regression Models

Linear Regression Model:

```
# Train linear regression model
lr = LinearRegression()
lr.fit(X, y)
```

Polynomial Regression Model:

```
# Train polynomial regression model
pr = PolynomialFeatures(degree=4)  # Degree of polynomial
X_poly = pr.fit_transform(X)
lr_2 = LinearRegression()
lr_2.fit(X_poly, y)
```

Step 6: Predict the Results

```
# Predict using linear regression
y_pred_lr = lr.predict(X)

# Predict using polynomial regression
y_pred_poly = lr_2.predict(X_poly)
```

Step 7: Visualization of Predictions

```
plt.scatter(X, y, color='lightcoral')
plt.plot(X, lr.predict(X), color='firebrick')
plt.title('Real Data with Linear Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.legend(['Predicted Salary (Linear)', 'Actual Salary'], loc='best', facecolor='white')
plt.box(False)
plt.show()
```

Polynomial Regression:

```
# Finer grid for smooth curve
X_grid = np.arange(min(X), max(X), 0.1)
X_grid = X_grid.reshape((len(X_grid), 1))

# Plot polynomial regression
plt.scatter(X, y, color='lightcoral')
plt.plot(X_grid, lr_2.predict(pr.fit_transform(X_grid)), color='firebrick')
plt.title('Real Data with Polynomial Regression')
plt.xlabel('Position Level')
plt.ylabel('Salary')
plt.legend(['Predicted Salary (Polynomial)', 'Actual Salary'], loc='best', facecolor='white')
plt.box(False)
plt.show()
```

Step 8: Test Predictions with Specific Input

Let's check the prediction for Level = 7.5 using both models:

```
# Predict with Linear Regression
print(f'Linear Regression Prediction for Level 7.5: {lr.predict([[7.5]])}')

# Predict with Polynomial Regression
print(f'Polynomial Regression Prediction for Level 7.5: {lr_2.predict(pr.fit_transform([[7.5]]))}')
```

LAB TASKS

- **Task 1:** Visualize the dataset using scatter plots and regression lines to compare Linear Regression and Polynomial Regression (degree = 4).
- **Task 2:** Train and evaluate Polynomial Regression models with degrees 2, 3, and 5. Compare their R-squared scores to identify the best-fitting model.
- **Task 3:** Implement and plot predictions for Polynomial Regression with Ridge regularization (degree = 4).
- **Task 4:** Calculate and compare Mean Absolute Error (MAE) and Mean Squared Error (MSE) for both Linear and Polynomial Regression models.
- **Task 5:** Use the trained Polynomial Regression model to predict salaries for Levels 6.5, 8.0, and 9.0. Visualize these predictions on the original dataset plot.