# Polynomial Regression using Scikit-Learn

## Project Description

This project demonstrates the use of **Polynomial Regression** to model **non-linear** relationships between features and the target variable. Polynomial regression is an extension of linear regression that incorporates polynomial features to capture complex patterns in the data. The workflow includes **data preprocessing, model training, evaluation, and visualization**.

---

## Why Polynomial Regression?

Polynomial regression is useful when:

- The relationship between the independent and dependent variables is **non-linear**.
- You want to improve model performance over linear regression by introducing polynomial terms.
- Understanding **residuals and prediction trends** is critical.

---

## Prerequisites

### Required Libraries

- **Python 3.7 or later**
- `pandas` : For data manipulation and analysis.
- `numpy` : For numerical computations.
- `scikit-learn` : For machine learning tools and evaluation metrics.
- `matplotlib` : For data visualization.

### Installation

Run the following command to install the necessary libraries:

```
pip install pandas numpy scikit-learn matplotlib
```

---

## Files Included

- `your_dataset.csv` : A placeholder dataset (replace with your actual dataset file).
- **Python code for polynomial regression**.

---

## Code Description

### Steps in the Code

### 1. Dataset Loading

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

```
from sklearn.metrics import mean_squared_error, r2_score

# Load dataset
data = pd.read_csv('your_dataset.csv')  # Replace with actual dataset file
print(data.head())
```

The dataset is loaded using pandas, and the first few rows are displayed to understand the structure.

---

## 2. Handling Missing Values

```
# Fill missing values with column-wise mean
data.fillna(data.mean(), inplace=True)
```

Missing values are replaced with the column mean to avoid disrupting model training.

---

## 3. Splitting Features and Target

```
X = data.iloc[:, :-1]  # All columns except the last as features
y = data.iloc[:, -1]   # The last column as the target
```

---

## 4. Splitting Training and Test Sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=4
```

The dataset is split into **80% training** and **20% testing**.

---

## 5. Polynomial Feature Transformation

```
# Transforming features into polynomial terms (degree 2)
poly = PolynomialFeatures(degree=2)  # Adjust degree as needed
X_train_poly = poly.fit_transform(X_train)
```

This step **converts features** into polynomial terms.

---

## 6. Model Training

```
# Train the regression model
model = LinearRegression()
model.fit(X_train_poly, y_train)
```

A **Linear Regression model** is trained on polynomial-transformed features.

---

## 7. Model Predictions

```
X_test_poly = poly.transform(X_test)
y_pred = model.predict(X_test_poly)
```

---

### 8. Evaluation Metrics

#### Mean Squared Error (MSE)

```
mse = mean_squared_error(y_test, y_pred)
print(f"Mean Squared Error: {mse:.2f}")
```

#### $R^2$ Score

```
r2 = r2_score(y_test, y_pred)
print(f"R² Score: {r2:.2f}")
```

---

### 9. Visualization

#### Scatter Plot: Actual vs Predicted

```
plt.scatter(y_test, y_pred, color='blue', alpha=0.6)
plt.title('Actual vs Predicted')
plt.xlabel('Actual Values')
plt.ylabel('Predicted Values')
plt.show()
```

#### Residual Plot

```
residuals = y_test - y_pred
plt.scatter(y_pred, residuals, color='green', alpha=0.6)
plt.axhline(y=0, color='red', linestyle='--')
plt.title('Residuals vs Predicted')
plt.xlabel('Predicted Values')
plt.ylabel('Residuals')
plt.show()
```

- **Scatter Plot:** Shows how well the model's predictions align with actual values.
- **Residuals Plot:** Highlights potential biases or patterns in the residuals.

---

# Outputs

## Metrics:

- **Mean Squared Error (MSE):** Measures the average squared difference between actual and predicted values.
- **$R^2$ Score:** Represents the proportion of variance explained by the model.

## Visualizations:

- **Actual vs Predicted**: Compares real vs predicted values.
- **Residuals vs Predicted**: Identifies any systematic patterns.

---

## Example Output

```
 Mean Squared Error: 2.35
R² Score: 0.89
```

---

## Use Cases

This project is applicable for:

- **Predictive modeling** in scenarios with **non-linear relationships**.
- **Improving regression models** by incorporating polynomial features.
- **Understanding model performance** and residual trends.

---

## Future Enhancements

- **Experiment with different polynomial degrees** for a better model fit.
- **Apply cross-validation** for a more robust performance evaluation.
- **Use regularization techniques** (e.g., Ridge, Lasso) to prevent overfitting.

---