Name: Girish Nandanwar	Course: 22CT744 – Lab. Machine Learning
Roll No: A-56	Department: Computer Technology

Practical No.3

Aim: To Implement Bias-Variance Tradeoff Using Polynomial Regression Model.

Theory:

Bias-Variance Tradeoff
The bias-variance tradeoff explains how model complexity affects prediction accuracy. The total model error has three parts: **bias**, **variance**, and **irreducible error**.

- **Bias**: Error due to simplifying assumptions, leading to *underfitting*.
- **Variance**: Error from model sensitivity to small data changes, causing *overfitting*.

Bias Formula:
($\text{Bias}^2 = (\mathbb{E}[\hat{f}(x)] - f(x))^2$)

Variance Formula:
($\text{Variance} = \mathbb{E}[(\hat{f}(x) - \mathbb{E}[\hat{f}(x)])^2]$)

Polynomial Regression
Polynomial regression models non-linear relationships by adding polynomial terms like ($x, x^2, x^3, \ldots, x^d$). It helps capture patterns that linear regression cannot.

Model Complexity	Bias	Variance	Result
Low Degree	High	Low	Underfitting
Moderate Degree	Balanced	Balanced	Best Generalization
High Degree	Low	High	Overfitting

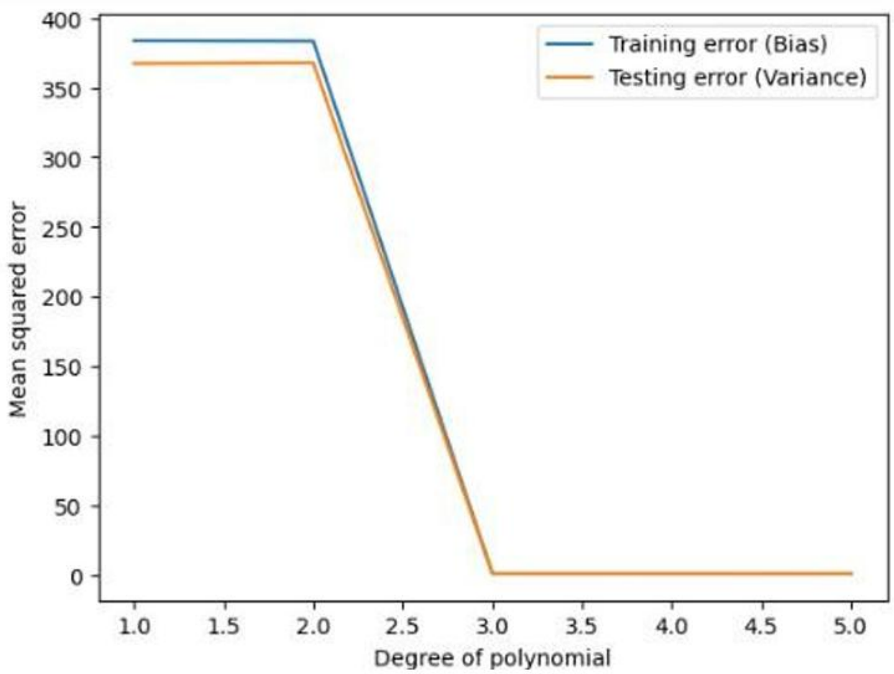
The goal is to find the degree that minimizes both training and test errors.

Code:

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

# Generate synthetic cubic data np.random.seed(0)
```

```
x = np.linspace(-5, 5, 100)
y = x**3 + np.random.normal(0, 20, 100)

# Split data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=0)

degrees = [1, 2, 3, 4, 5, 10, 15]
train_errors, test_errors, train_r2, test_r2 = [], [], [], []

for d in degrees:
poly = PolynomialFeatures(d)
X_train_poly = poly.fit_transform(x_train.reshape(-1, 1)) X_test_poly = poly.transform(x_test.reshape(-1, 1))
model = LinearRegression() model.fit(X_train_poly, y_train)
y_train_pred = model.predict(X_train_poly) y_test_pred = model.predict(X_test_poly)
train_errors.append(mean_squared_error(y_train, y_train_pred))
test_errors.append(mean_squared_error(y_test, y_test_pred)) train_r2.append(r2_score(y_train,
y_train_pred)) test_r2.append(r2_score(y_test, y_test_pred))

optimal_degree = degrees[np.argmin(test_errors)] print(f"Optimal Degree: {optimal_degree}")

# Plot Bias-Variance Tradeoff plt.figure(figsize=(12, 5)) plt.subplot(1,
2, 1)
plt.plot(degrees, train_errors, marker='o', label='Train Error') plt.plot(degrees, test_errors, marker='s',
label='Test Error') plt.axvline(optimal_degree, color='red', linestyle='--', label=f'Optimal
Degree={optimal_degree}')
plt.title('Bias-Variance Tradeoff') plt.xlabel('Polynomial Degree') plt.ylabel('Mean Squared Error')
plt.legend()
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(degrees, train_r2, marker='o', label='Train R²') plt.plot(degrees, test_r2, marker='s', label='Test R²')

plt.title('R² Score vs Polynomial Degree') plt.xlabel('Polynomial Degree') plt.ylabel('R² Score') plt.legend()
plt.grid(True) plt.tight_layout() plt.show()
```

Result:



Conclusion:

The bias-variance tradeoff was successfully demonstrated using polynomial regression. Low-degree models underfit due to high bias, while high-degree models overfit due to high variance. The optimal polynomial degree was 3, giving the best balance between bias and variance with the lowest test error.