PROGRAM – 6

Program to Learn bias-variance trade-off

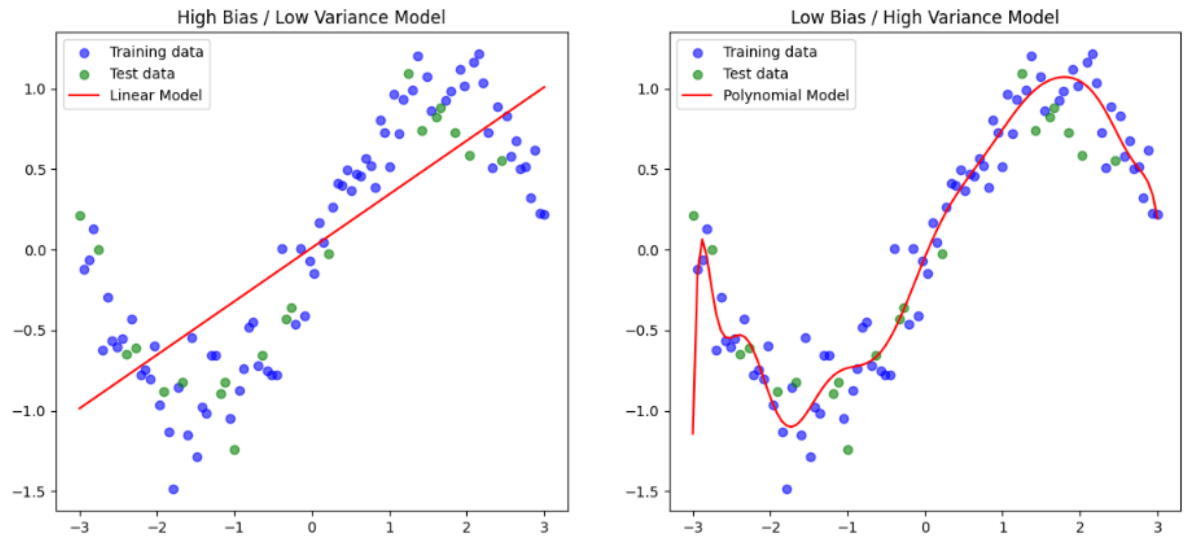
# estimate the bias and variance for a regression model

# Exhibiting low bias, high variance model and high bias, low variance model

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error
from sklearn.pipeline import make_pipeline
from sklearn.preprocessing import PolynomialFeatures

# Generate a synthetic dataset
np.random.seed(0)
X = np.linspace(-3, 3, 100)
y = np.sin(X) + np.random.normal(0, 0.2, 100)  # Adding noise
X = X[:, np.newaxis]  # Reshape for sklearn

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# High Bias / Low Variance Model (Linear Regression)
linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_train_linear = linear_model.predict(X_train)
y_pred_test_linear = linear_model.predict(X_test)

# Low Bias / High Variance Model (Polynomial Regression with high degree)
poly_model = make_pipeline(PolynomialFeatures(degree=15), LinearRegression())
poly_model.fit(X_train, y_train)
y_pred_train_poly = poly_model.predict(X_train)
y_pred_test_poly = poly_model.predict(X_test)

# Plotting
plt.figure(figsize=(14, 6))

# Linear Model Plot
plt.subplot(1, 2, 1)
plt.scatter(X_train, y_train, label='Training data', color='blue', alpha=0.6)
plt.scatter(X_test, y_test, label='Test data', color='green', alpha=0.6)
plt.plot(X, linear_model.predict(X), color='red', label='Linear Model')
plt.title('High Bias / Low Variance Model')
plt.legend()

# Polynomial Model Plot
```

```
plt.subplot(1, 2, 2)
plt.scatter(X_train, y_train, label='Training data', color='blue', alpha=0.6)
plt.scatter(X_test, y_test, label='Test data', color='green', alpha=0.6)
plt.plot(np.linspace(-3, 3, 100), poly_model.predict(np.linspace(-3, 3, 100)[:,
np.newaxis]), color='red', label='Polynomial Model')
plt.title('Low Bias / High Variance Model')
plt.legend()

plt.show()

# Print Mean Squared Error for both models
print(f"Linear Model Training MSE: {mean_squared_error(y_train,
y_pred_train_linear)}")
print(f"Linear Model Testing MSE: {mean_squared_error(y_test, y_pred_test_linear)}")
print(f"Polynomial Model Training MSE: {mean_squared_error(y_train,
y_pred_train_poly)}")
print(f"Polynomial Model Testing MSE: {mean_squared_error(y_test,
y_pred_test_poly)}")
```

OUTPUT



```
Linear Model Training MSE: 0.19743335485331392
Linear Model Testing MSE: 0.24695224333039673
Polynomial Model Training MSE: 0.029514806989774718
Polynomial Model Testing MSE: 0.1425297085836628
```

```
# Calculate Mean Squared Error (MSE) for training and testing sets
mse_train_linear = mean_squared_error(y_train, y_pred_train_linear)
mse_test_linear = mean_squared_error(y_test, y_pred_test_linear)
mse_train_poly = mean_squared_error(y_train, y_pred_train_poly)
mse_test_poly = mean_squared_error(y_test, y_pred_test_poly)

# Display MSE for both models
print("Linear Regression Model (High Bias/Low Variance)")
print(f"Training MSE: {mse_train_linear}")
print(f"Testing MSE: {mse_test_linear}\n")

print("Polynomial Regression Model (Low Bias/High Variance)")
print(f"Training MSE: {mse_train_poly}")
print(f"Testing MSE: {mse_test_poly}")
```

OUTPUT:

```
Linear Regression Model (High Bias/Low Variance)
Training MSE: 0.19743335485331392
Testing MSE: 0.24695224333039673

Polynomial Regression Model (Low Bias/High Variance)
Training MSE: 0.029514806989774718
Testing MSE: 0.1425297085836628
```

```
from mlxtend.evaluate import bias_variance_decomp
mse, bias, var = bias_variance_decomp(linear_model, X_train, y_train, X_test, y_test,
loss='mse', num_rounds=200, random_seed=1)
# summarize results
print('MSE: %.3f' % mse)
print('Bias: %.3f' % bias)
print('Variance: %.3f' % var)
```

OUTPUT:

```
MSE: 0.253
Bias: 0.247
Variance: 0.006
```

```
mse, bias, var = bias_variance_decomp(poly_model, X_train, y_train, X_test, y_test,
loss='mse', num_rounds=200, random_seed=1)
# summarize results
print('MSE: %.3f' % mse)
print('Bias: %.3f' % bias)
print('Variance: %.3f' % var)
```

OUTPUT:

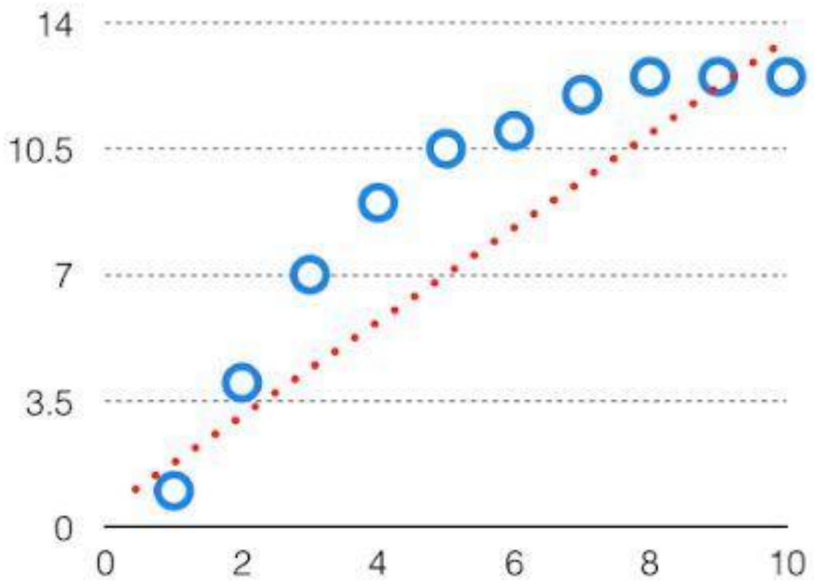
```
MSE: 2.609
Bias: 0.140
Variance: 2.468
```

Theory:

It is important to understand prediction errors (bias and variance) when it comes to accuracy in any machine-learning algorithm. There is a trade-off between a model's ability to minimize bias and variance which is referred to as the best solution for selecting a value of **Regularization** constant. A proper understanding of these errors would help to avoid the overfitting and underfitting of a data set while training the algorithm.

What is Bias?

The bias is known as the difference between the prediction of the values by the Machine Learning model and the correct value. Being high in biasing gives a large error in training as well as testing data. It recommended that an algorithm should always be low-biased to avoid the problem of underfitting. By high bias, the data predicted is in a straight line format, thus not fitting accurately in the data in the data set. Such fitting is known as the **Underfitting of Data**. This happens when the hypothesis is too simple or linear in nature. Refer to the graph given below for an example of such a situation.

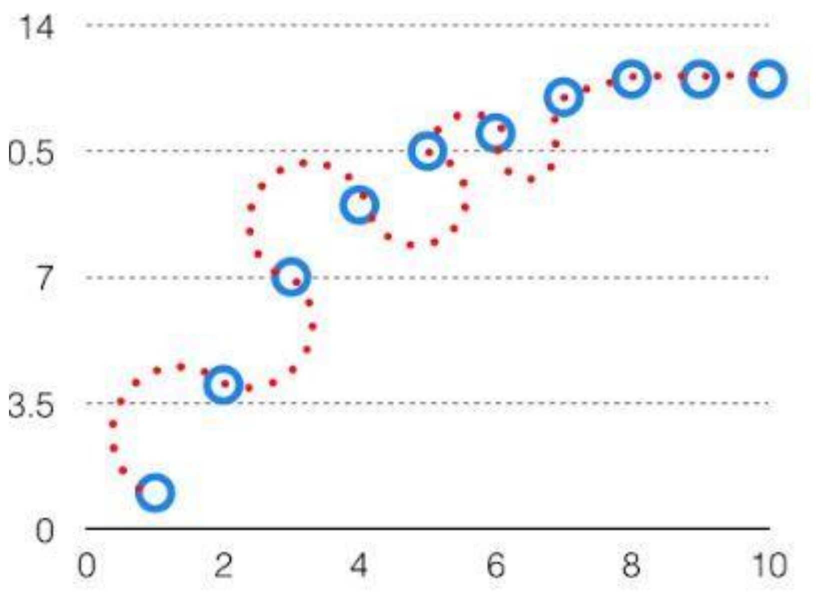


In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g(\theta_0 + \theta_1 x_1 + \theta_2 x_2)$$

What is Variance?

The variability of model prediction for a given data point which tells us the spread of our data is called the variance of the model. The model with high variance has a very complex fit to the training data and thus is not able to fit accurately on the data which it hasn't seen before. As a result, such models perform very well on training data but have high error rates on test data. When a model is high on variance, it is then said to as **Overfitting of Data**. Overfitting is fitting the training set accurately via complex curve and high order hypothesis but is not the solution as the error with unseen data is high. While training a data model variance should be kept low. The high variance data looks as follows.

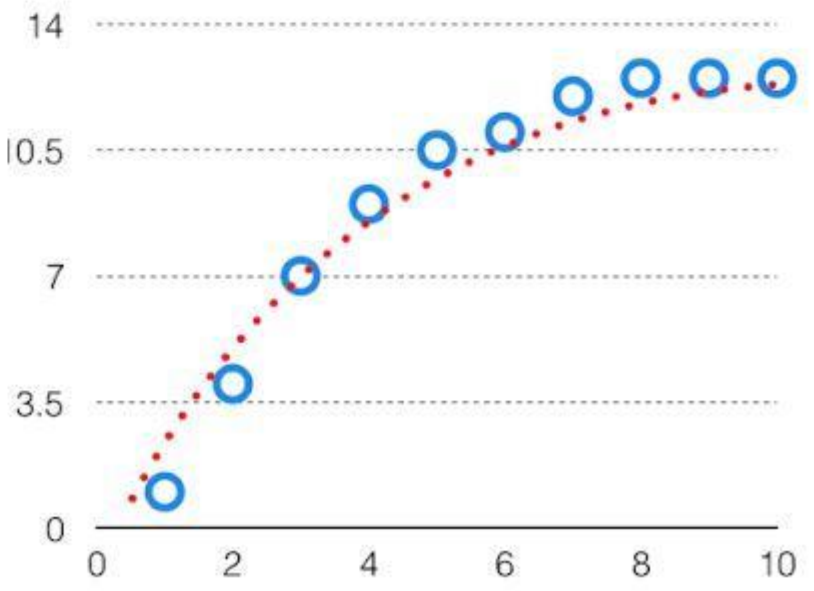


In such a problem, a hypothesis looks like follows.

$$h_\theta(x) = g\left(\theta_0 + \theta_1 x + \theta_2 x^2 + \theta_3 x^3 + \theta_4 x^4\right)$$

Bias Variance Trade-off

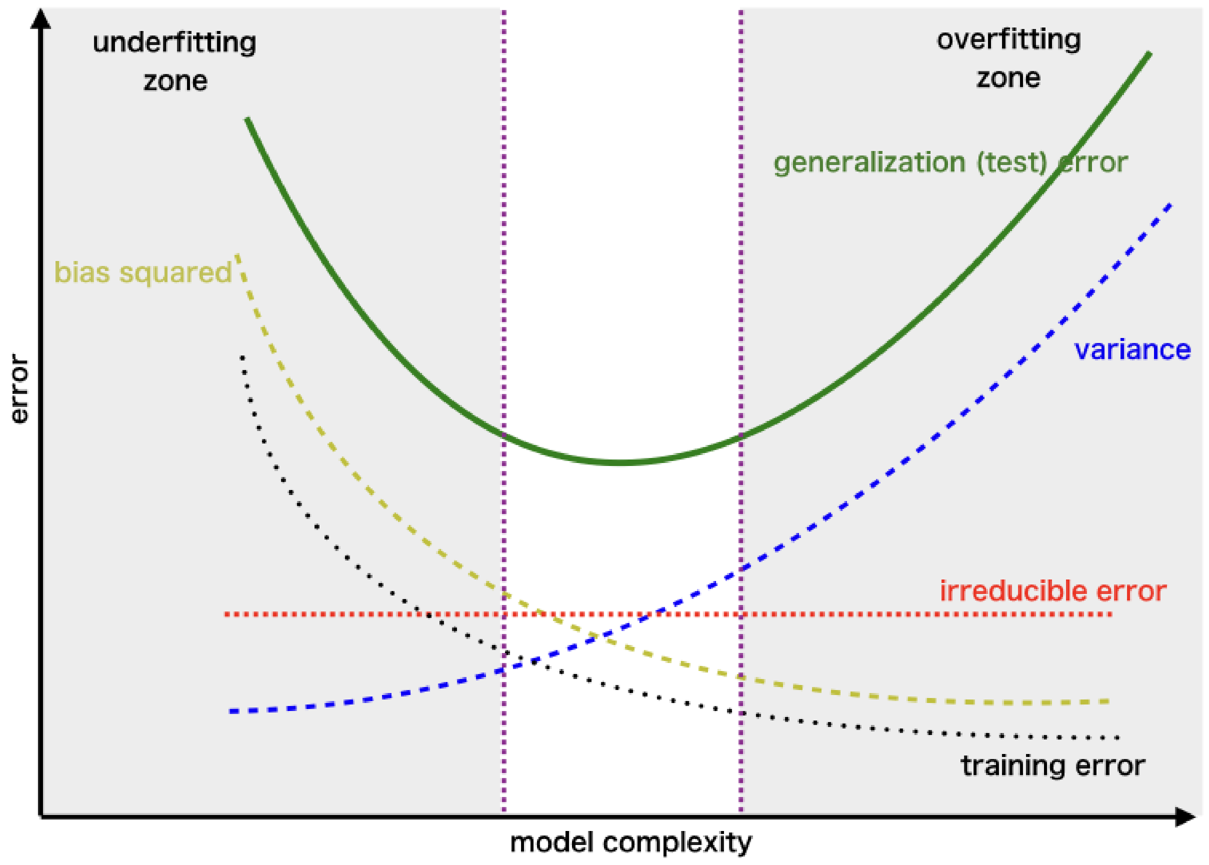
If the algorithm is too simple (hypothesis with linear equation) then it may be on high bias and low variance condition and thus is error-prone. If algorithms fit too complex (hypothesis with high degree equation) then it may be on high variance and low bias. In the latter condition, the new entries will not perform well. Well, there is something between both of these conditions, known as a Trade-off or Bias Variance Trade-off. This tradeoff in complexity is why there is a tradeoff between bias and variance. An algorithm can't be more complex and less complex at the same time. For the graph, the perfect tradeoff will be like this.



We try to optimize the value of the total error for the model by using the Bias-Variance Trade-off.

$$\text{Total Error} = \text{Bias}^2 + \text{Variance} + \text{Irreducible Error}$$

The best fit will be given by the hypothesis on the trade-off point. The error to complexity graph to show trade-off is given as –



Region for the Least Value of Total Error

This is referred to as the best point chosen for the training of the algorithm which gives low error in training as well as testing data.