COMSATS UNIVERSITY ABBOTABAD DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING

Submitted By:

NAME	Registration
Syed M Usman	FA22-BCE-068

Machine learning

Assignment#: 2

Date: 26-05-2025

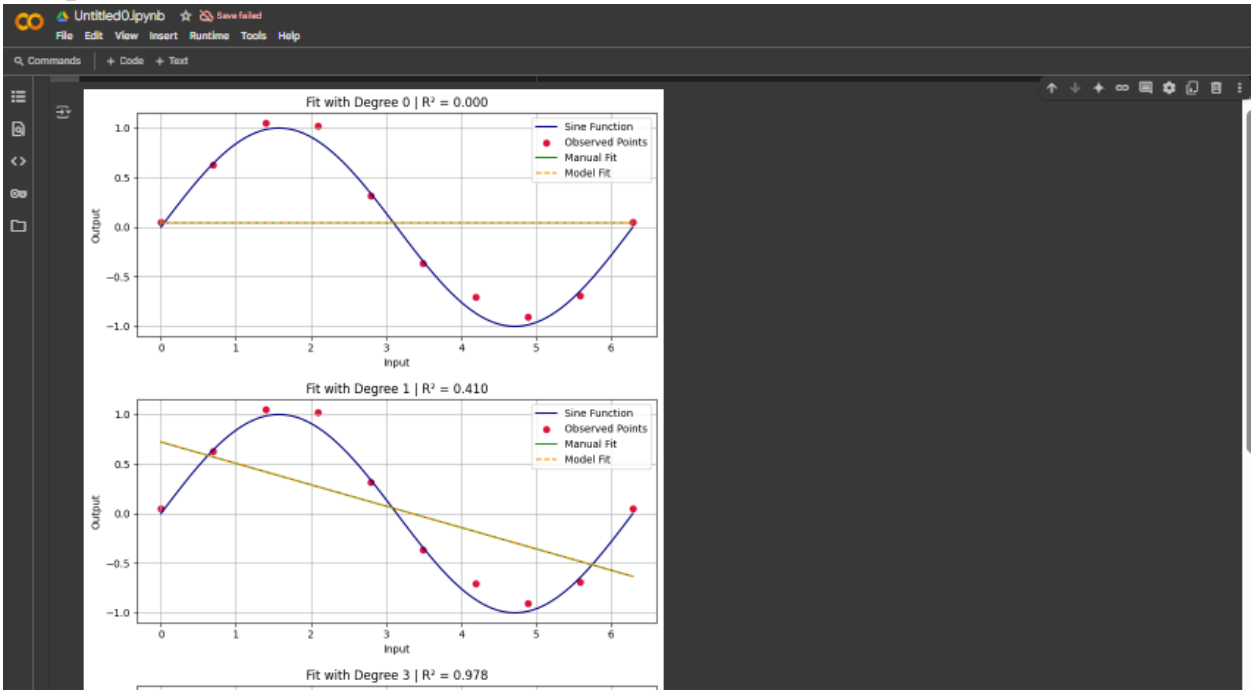
Question 1:

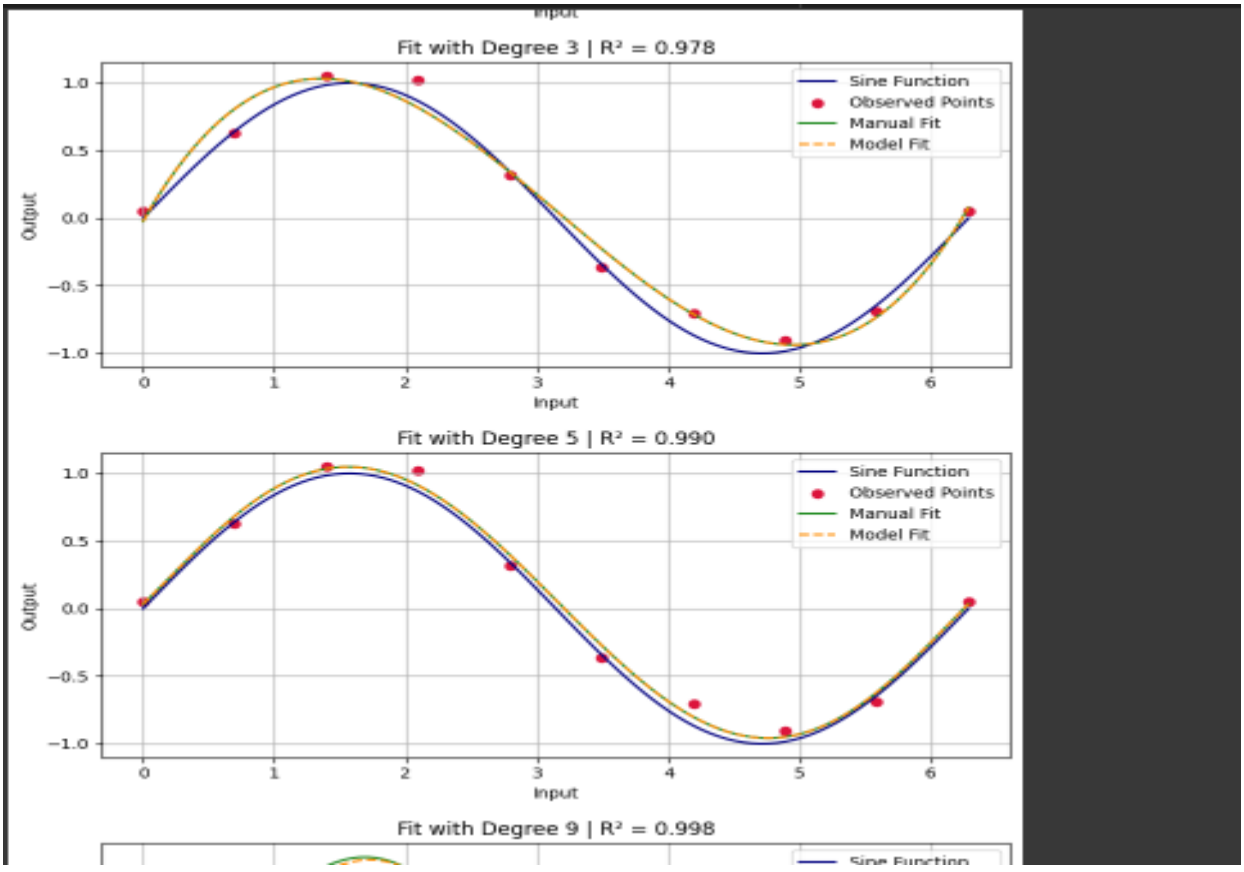
Code:

```
import numpy as np
import matplotlib.pyplot as plt
from numpy.linalg import pinv
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score

np.random.seed(0)
x = np.linspace(0, 2 * np.pi, 10)
y_true = np.sin(x)
noise = np.random.normal(0, 0.1, x.shape)
y_noisy = y_true + noise

degrees = [0, 1, 3, 5, 9]

for degree in degrees:
    X_poly = np.vander(x, degree + 1, increasing=True)
    theta = pinv(X_poly.T @ X_poly) @ X_poly.T @ y_noisy
    y_pred = X_poly @ theta

    model = LinearRegression()
    model.fit(X_poly, y_noisy)
    y_sklearn = model.predict(X_poly)

    x_dense = np.linspace(0, 2 * np.pi, 100)
    X_dense_poly = np.vander(x_dense, degree + 1, increasing=True)
    y_dense_pred = X_dense_poly @ theta
    y_dense_sklearn = model.predict(X_dense_poly)
    y_true_dense = np.sin(x_dense)

    plt.figure(figsize=(8, 4))
    plt.plot(x_dense, y_true_dense, label='True sin(x)', color='blue')
    plt.scatter(x, y_noisy, label='Noisy data', color='red')
```

```
    plt.plot(x_dense, y_dense_pred, label='Normal Eq. fit', color='green')
    plt.plot(x_dense, y_dense_sklearn, '--', label='Sklearn fit', color='purple')
    r2 = r2_score(y_noisy, y_pred)
    plt.title(f'Polynomial Degree {degree} | R² Score: {r2:.3f}')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.legend()
    plt.grid(True)
    plt.tight_layout()
    plt.show()

    if not np.allclose(y_pred, y_sklearn):
        print(f"Warning: Slight mismatch found at degree {degree}!")
```

Output:


Untitled0.ipynb
File Edit View Insert Runtime Tools Help
Commands + Code + Text
Fit with Degree 0 | R² = 0.000
Sine Function
Observed Points
Manual Fit
Model Fit
Output
Input
Fit with Degree 1 | R² = 0.410
Sine Function
Observed Points
Manual Fit
Model Fit
Output
Input
Fit with Degree 3 | R² = 0.978

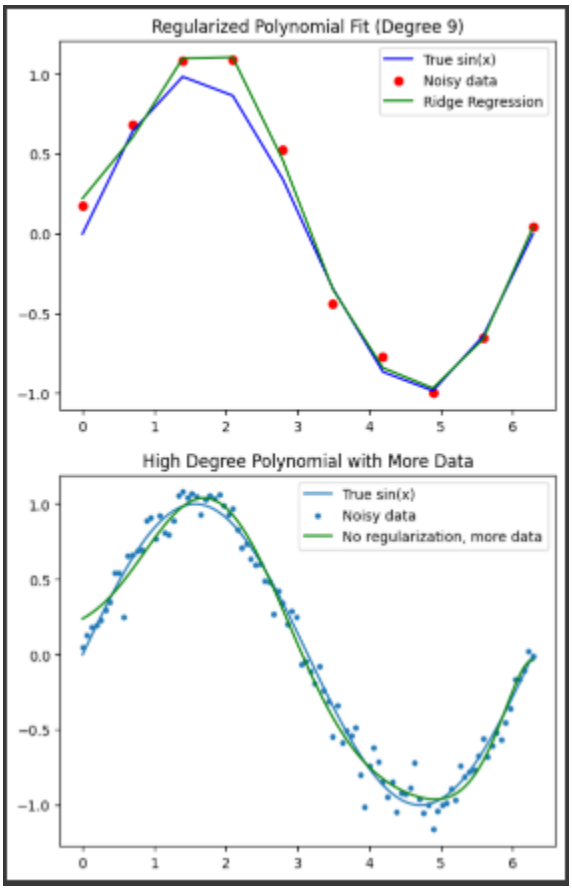

Fit with Degree 3 | R² = 0.978
Sine Function
Observed Points
Manual Fit
Model Fit
Output
Input
Fit with Degree 5 | R² = 0.990
Sine Function
Observed Points
Manual Fit
Model Fit
Output
Input
Fit with Degree 9 | R² = 0.998

QUESTION 2:

CODE:

```
# Regularized Normal Equation (Ridge)
def ridge_regression(X, y, lambda_):
    I = np.eye(X.shape[1])
    return pinv(X.T @ X + lambda_ * I) @ X.T @ y

# Observe overfitting at degree 9
degree = 9
X_poly = np.vander(x, degree + 1, increasing=True)
theta_ridge = ridge_regression(X_poly, y_noisy, lambda_=0.1)
y_ridge = X_poly @ theta_ridge

plt.plot(x, y_true, label='True sin(x)', color='blue')
plt.scatter(x, y_noisy, label='Noisy data', color='red')
plt.plot(x, y_ridge, label='Ridge Regression', color='green')
plt.title('Regularized Polynomial Fit (Degree 9)')
plt.legend()
plt.show()

# Increase data points
x_more = np.linspace(0, 2 * np.pi, 100)
y_more_true = np.sin(x_more)
noise_more = np.random.normal(0, 0.1, x_more.shape)
y_more_noisy = y_more_true + noise_more

X_more_poly = np.vander(x_more, degree + 1, increasing=True)
theta_more = pinv(X_more_poly.T @ X_more_poly) @ X_more_poly.T @ y_more_noisy
y_more_pred = X_more_poly @ theta_more

plt.plot(x_more, y_more_true, label='True sin(x)')
plt.scatter(x_more, y_more_noisy, label='Noisy data', s=10)
plt.plot(x_more, y_more_pred, label='No regularization, more data', color='green')
plt.title('High Degree Polynomial with More Data')
plt.legend()
plt.show()
```

OUTPUT:


Regularized Polynomial Fit (Degree 9)
True sin(x)
Noisy data
Ridge Regression
High Degree Polynomial with More Data
True sin(x)
Noisy data
No regularization, more data