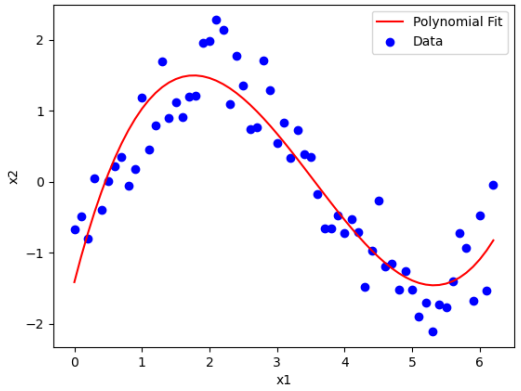
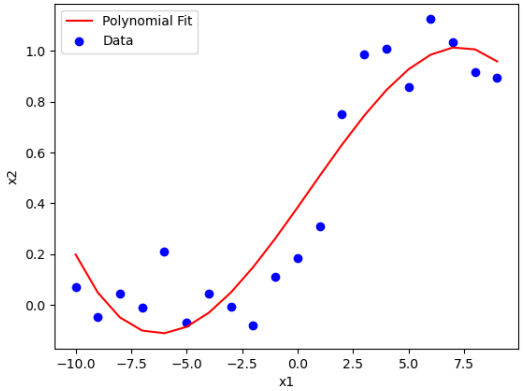
a) The data is a noisy sinusoidal signal.




```
# load datasets
regression_1 = pd.read_csv('regression_1.csv')

# regression 1
X1 = regression_1['x1'].values.reshape(-1, 1)
y1 = regression_1['x2'].values

poly_reg_1 = make_pipeline(PolynomialFeatures(degree=3), LinearRegression())
poly_reg_1.fit(X1, y1)

# plot regression 1
plt.scatter(X1, y1, color='blue', label='Data')
plt.plot(np.sort(X1, axis=0), poly_reg_1.predict(np.sort(X1, axis=0)), color='red', label='Polynomial Fit')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

I fitted a polynomial function of degree 3 to the data to capture its nonlinear trend. The result is a smooth curve that minimizes the distance (residuals) between the predicted values and the given data points, effectively modeling the data's sinusoidal-like behavior.

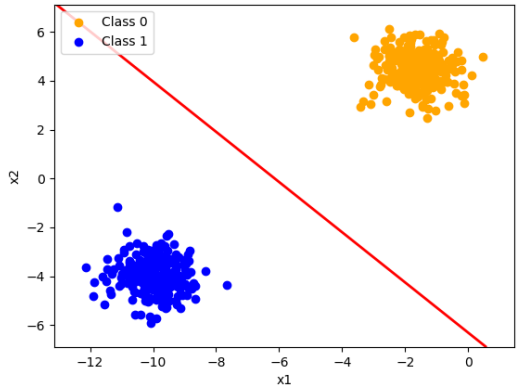
b) The data is a noisy smooth signal.




```
# load datasets
regression_2 = pd.read_csv('regression_2.csv')

# regression 2
X2 = regression_2['x1'].values.reshape(-1, 1)
y2 = regression_2['x2'].values

poly_model = make_pipeline(PolynomialFeatures(degree=3), LinearRegression())
poly_model.fit(X2, y2)

# plot regression 2
plt.scatter(X2, y2, color='blue', label='Data')
plt.plot(np.sort(X2, axis=0), poly_model.predict(np.sort(X2, axis=0)), color='red', label='Polynomial Fit')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

The data exhibits a noisy signal with an underlying smooth non-linear trend. By fitting a cubic polynomial function, the model successfully captures the general pattern while minimizing the distance between the curve and the data points, despite the noise. This approach achieves a balance between accurately representing the trend and avoiding overfitting.

c) The data is a separable binary classification.




```
# Load dataset
classification = pd.read_csv('classification.csv')

# Prepare data
X3 = classification[['x1', 'x2']].values
y3 = classification['label'].values

# Train SVM Classifier
svc = SVC(kernel='linear')
svc.fit(X3, y3)

# Plot Classification
plt.scatter(X3[y3 == 0][:, 0], X3[y3 == 0][:, 1], color='orange', label='Class 0')
plt.scatter(X3[y3 == 1][:, 0], X3[y3 == 1][:, 1], color='blue', label='Class 1')

# Decision Boundary
xx, yy = np.meshgrid(np.linspace(X3[:, 0].min() - 1, X3[:, 0].max() + 1, 100),
                     np.linspace(X3[:, 1].min() - 1, X3[:, 1].max() + 1, 100))
Z = svc.decision_function(np.c_[xx.ravel(), yy.ravel()]).reshape(xx.shape)
plt.contour(xx, yy, Z, levels=[0], linewidths=2, colors='red')
plt.xlim(X3[:, 0].min() - 1, X3[:, 0].max() + 1)
plt.ylim(X3[:, 1].min() - 1, X3[:, 1].max() + 1)
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

The data is linearly separable with two distinct clusters for Class 0 and Class 1. The SVM model with a linear kernel successfully separates the two classes using a clear decision boundary. This shows the dataset is simple and suitable for linear classification.

I-En Lee, 23365648, di65haji