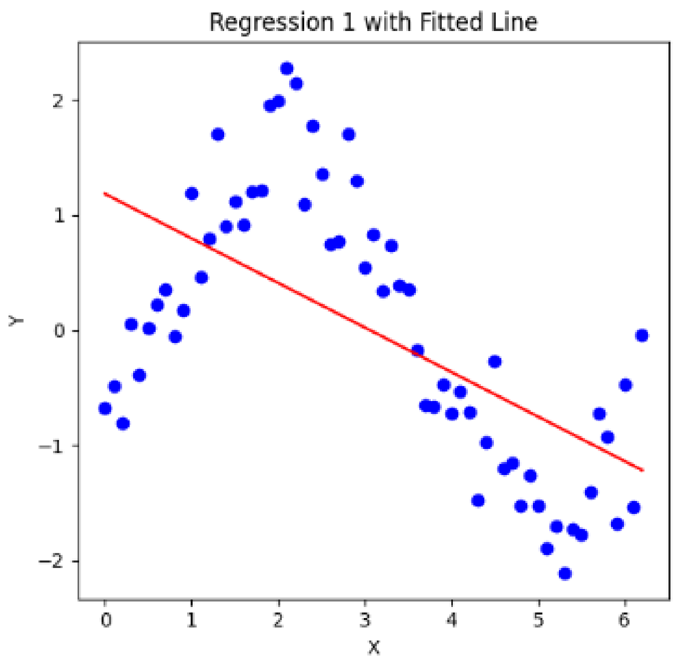
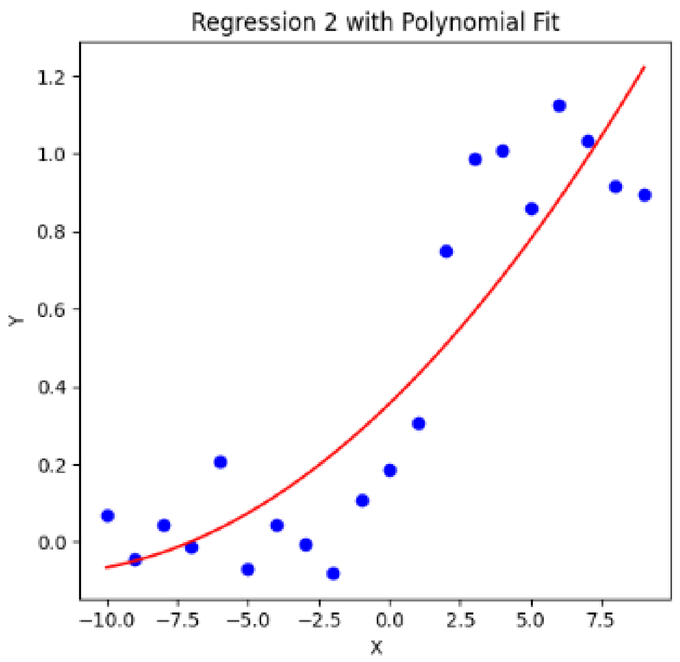
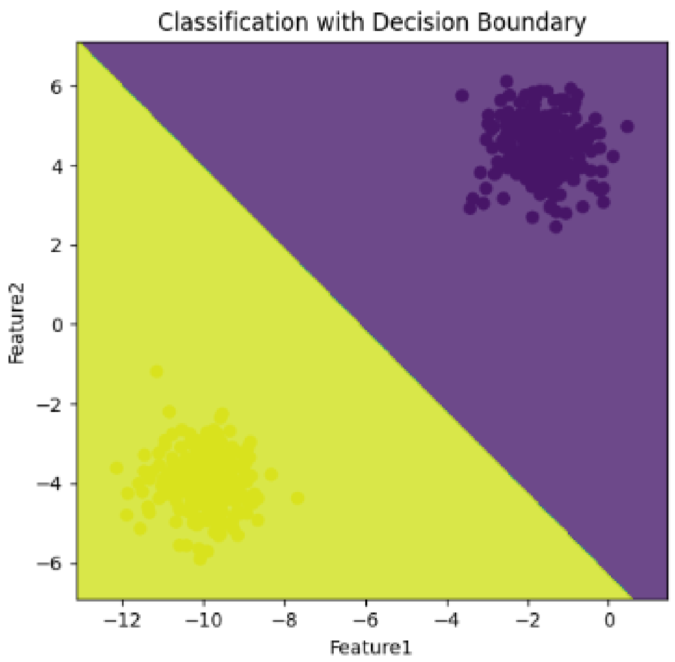
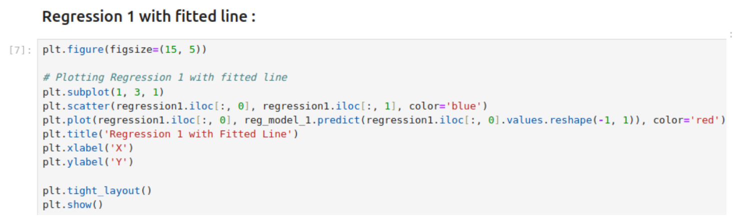
Homework 7



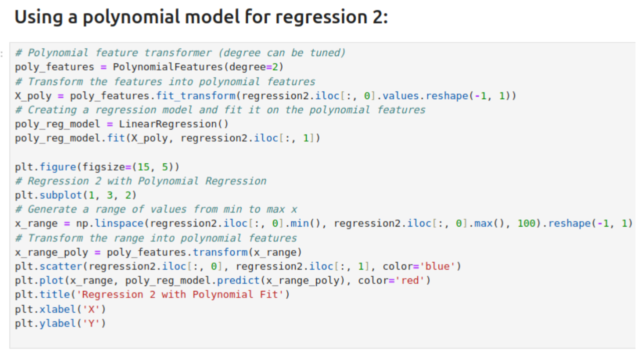






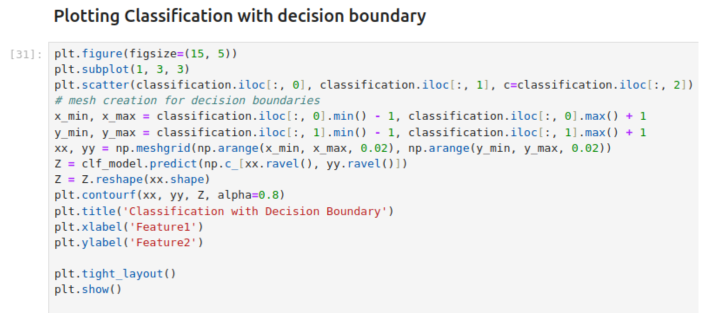
Regression 1 with fitted line :

```
[7]: plt.figure(figsize=(15, 5))

     # Plotting Regression 1 with fitted line
     plt.subplot(1, 3, 1)
     plt.scatter(regression1.iloc[:, 0], regression1.iloc[:, 1], color='blue')
     plt.plot(regression1.iloc[:, 0], reg_model_1.predict(regression1.iloc[:, 0].values.reshape(-1, 1)), color='red')
     plt.title('Regression 1 with Fitted Line')
     plt.xlabel('X')
     plt.ylabel('Y')

     plt.tight_layout()
     plt.show()
```



Using a polynomial model for regression 2:

```
# Polynomial feature transformer (degree can be tuned)
poly_features = PolynomialFeatures(degree=2)
# Transform the features into polynomial features
X_poly = poly_features.fit_transform(regression2.iloc[:, 0].values.reshape(-1, 1))
# Creating a regression model and fit it on the polynomial features
poly_reg_model = LinearRegression()
poly_reg_model.fit(X_poly, regression2.iloc[:, 1])

plt.figure(figsize=(15, 5))
# Regression 2 with Polynomial Regression
plt.subplot(1, 3, 2)
# Generate a range of values from min to max x
x_range = np.linspace(regression2.iloc[:, 0].min(), regression2.iloc[:, 0].max(), 100).reshape(-1, 1)
# Transform the range into polynomial features
x_range_poly = poly_features.transform(x_range)
plt.scatter(regression2.iloc[:, 0], regression2.iloc[:, 1], color='blue')
plt.plot(x_range, poly_reg_model.predict(x_range_poly), color='red')
plt.title('Regression 2 with Polynomial Fit')
plt.xlabel('X')
plt.ylabel('Y')
```



Plotting Classification with decision boundary

```
[31]: plt.figure(figsize=(15, 5))
      plt.subplot(1, 3, 3)
      plt.scatter(classification.iloc[:, 0], classification.iloc[:, 1], c=classification.iloc[:, 2])
      # mesh creation for decision boundaries
      x_min, x_max = classification.iloc[:, 0].min() - 1, classification.iloc[:, 0].max() + 1
      y_min, y_max = classification.iloc[:, 1].min() - 1, classification.iloc[:, 1].max() + 1
      xx, yy = np.meshgrid(np.arange(x_min, x_max, 0.02), np.arange(y_min, y_max, 0.02))
      Z = clf_model.predict(np.c_[xx.ravel(), yy.ravel()])
      Z = Z.reshape(xx.shape)
      plt.contourf(xx, yy, Z, alpha=0.8)
      plt.title('Classification with Decision Boundary')
      plt.xlabel('Feature1')
      plt.ylabel('Feature2')

      plt.tight_layout()
      plt.show()
```

(a)

The linear regression model was chosen due to the apparent linear trend in the data, with the fitted line closely following the central tendency of the data points.

(b)

I used a polynomial regression model to capture the non-linear trend observed in the data. The second-degree polynomial provide a good fit, modeling the curvature observed among the data points without appearing to overfit. I choose the degree of polynomial as 2 because it was the capturing the pattern keeping the model simple.

(c)

I used a linear SVM, and the resulting decision boundary shows a good separation of the two classes. This model choice is supported by initial value assessment that the classes are fairly linearly separable.