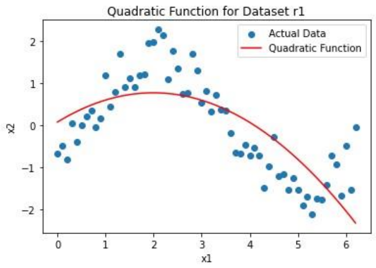
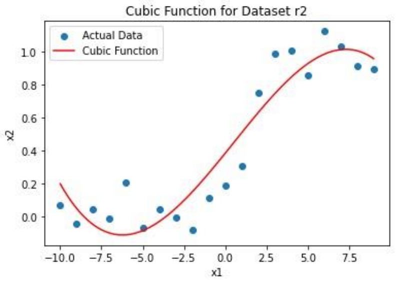
Name: Rutvi shah
Matriculation:23159043
Idm :to20raje



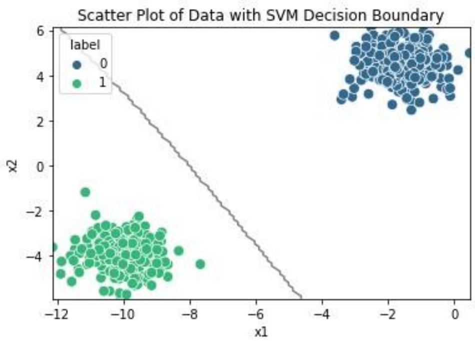

```
r1 = pd.read_csv("C:\\Users\\rutvi\\OneDrive\\Desktop\\Masters\\Semester 3\\Data science survival skills\\Exercise\\Assign

# Display the first few rows of the dataframe to understand the structure
print(r1.head())
# Display basic information about the dataframe
print(r1.info())

# Summary statistics
print(r1.describe())

# Check for missing values
print(r1.isnull().sum())

# Assuming 'x1' as the feature and 'x2' as the target
X = r1[['x1']]
y = r1['x2']

# Transform features to quadratic
poly_features = PolynomialFeatures(degree=2)
X_poly = poly_features.fit_transform(X)

# Fit linear regression with quadratic features
model = LinearRegression()
model.fit(X_poly, y)

# Generate predictions
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
X_range_poly = poly_features.transform(X_range)
predictions = model.predict(X_range_poly)

# Plot the actual data points and the fitted quadratic function
plt.scatter(X, y, label='Actual Data')
plt.plot(X_range, predictions, color='red', label='Quadratic Function')
plt.title('Quadratic Function for Dataset r1')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

● Values: The values include 'x1' (feature) and 'x2' (target) from the r1 dataset.
● Slope: In the context of polynomial regression, the slopes represent the Coefficients of the quadratic terms.
● Function: The fitted quadratic function is visualized, representing the Relationship between 'x1' and 'x2' in the dataset.




```
# Display the first few rows of the dataframe to understand the structure
print(r2.head())
# Display basic information about the dataframe
print(r2.info())

# Summary statistics
print(r2.describe())

# Check for missing values
print(r2.isnull().sum())


# Assuming 'x1' as the feature and 'x2' as the target
X = r2[['x1']]
y = r2['x2']

# Transform features to cubic
poly_features = PolynomialFeatures(degree=3)
X_poly = poly_features.fit_transform(X)

# Fit linear regression with cubic features
model = LinearRegression()
model.fit(X_poly, y)

# Generate predictions
X_range = np.linspace(X.min(), X.max(), 100).reshape(-1, 1)
X_range_poly = poly_features.transform(X_range)
predictions = model.predict(X_range_poly)

# Plot the actual data points and the fitted cubic function
plt.scatter(X, y, label='Actual Data')
plt.plot(X_range, predictions, color='red', label='Cubic Function')
plt.title('Cubic Function for Dataset r2')
plt.xlabel('x1')
plt.ylabel('x2')
plt.legend()
plt.show()
```

● Values: The values include 'x1' (feature) and 'x2' (target) from the r2 dataset.
● Slope: In the context of polynomial regression, the slopes represent the coefficients of the cubic terms.
● Function: The fitted cubic function is visualized, representing the relationship between 'x1' and 'x2' in the dataset.




```
# Split the data 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)

# Create and train an SVM classifier with a polynomial kernel
# You can adjust the degree parameter to control the polynomial degree
classifier = SVC(kernel='poly', degree=3)  # 'poly' for polynomial kernel
classifier.fit(X_train, y_train)

# Make predictions on the test set
y_pred = classifier.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)

# Print evaluation metrics
print(f'Accuracy: {accuracy}')
print('Confusion Matrix:')
print(conf_matrix)

# Scatter plot of the data points
sns.scatterplot(x='x1', y='x2', hue='label', data=c, palette='viridis', marker='o', s=75)
plt.title('Scatter Plot of Data with SVM Decision Boundary')
plt.xlabel('x1')
plt.ylabel('x2')

# Create a meshgrid to plot the decision boundary
xx, yy = np.meshgrid(np.linspace(c['x1'].min(), c['x1'].max(), 100),
                     np.linspace(c['x2'].min(), c['x2'].max(), 100))

# Predict labels for each point in the meshgrid
Z = classifier.predict(np.c_[xx.ravel(), yy.ravel()])
Z = Z.reshape(xx.shape)

# Plot the decision boundary
plt.contour(xx, yy, Z, colors='k', levels=[-1, 0, 1], alpha=0.5, linestyles=['--', '-', '--'])

plt.show()
```

Values: The values include 'x1' and 'x2' as features, and 'label' as the target variable.
● SVM Polynomial Kernel: The SVM classifier uses a polynomial kernel of degree 3.
● Visualization: The scatter plot visualizes the data points, and the decision boundary of the SVM is plotted.