Lab No 05

Name Maaz Ahamd
Reg No B23F0722AI170
Section Ai Yellow
Submitted to Sir Abdullah Sajid

Task 1

Ridge Regression with Feature Scaling (California Housing Dataset)

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import fetch_california_housing
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, Ridge
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer

california = fetch_california_housing()
X = pd.DataFrame(california.data, columns=california.feature_names)
y = pd.Series(california.target, name='MedHouseVal')

imputer = SimpleImputer(strategy='mean')
X_imputed = pd.DataFrame(imputer.fit_transform(X), columns=X.columns)

X_train, X_test, y_train, y_test = train_test_split(X_imputed, y, test_size=0.2, random_state=42)

linear_model = LinearRegression()
linear_model.fit(X_train, y_train)
y_pred_linear = linear_model.predict(X_test)

mse_linear = mean_squared_error(y_test, y_pred_linear)
r2_linear = r2_score(y_test, y_pred_linear)
print("Linear Regression Model:")
print(f"Mean Squared Error (MSE): {mse_linear:.4f}")
print(f"R-squared: {r2_linear:.4f}")

scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train_scaled, y_train)
y_pred_ridge = ridge_model.predict(X_test_scaled)

mse_ridge = mean_squared_error(y_test, y_pred_ridge)
r2_ridge = r2_score(y_test, y_pred_ridge)
print("\nRidge Regression Model (with Feature Scaling):")
print(f"Mean Squared Error (MSE): {mse_ridge:.4f}")
print(f"R-squared: {r2_ridge:.4f}")

print("\nModel Performance Comparison:")
print(f"Linear Regression MSE: {mse_linear:.4f}, Ridge Regression MSE: {mse_ridge:.4f}")
print(f"Linear Regression R²: {r2_linear:.4f}, Ridge Regression R²: {r2_ridge:.4f}")

plt.figure(figsize=(12, 6))
linear_coef = pd.Series(linear_model.coef_, index=X.columns)
linear_coef = linear_coef.sort_values()
ridge_coef = pd.Series(ridge_model.coef_, index=X.columns)
ridge_coef = ridge_coef.sort_values()
x_pos = np.arange(len(X.columns))
width = 0.35
fig, ax = plt.subplots(figsize=(12, 8))
bars1 = ax.bar(x_pos - width/2, linear_coef, width, label='Linear Regression')
bars2 = ax.bar(x_pos + width/2, ridge_coef, width, label='Ridge Regression')
```

```
ax.set_xlabel('Features')
ax.set_ylabel('Coefficient Value')
ax.set_title('Comparison of Feature Coefficients: Linear vs Ridge Regression')
ax.set_xticks(x_pos)
ax.set_xticklabels(X.columns, rotation=45, ha='right')
ax.legend()

def add_value_labels(bars):
    for bar in bars:
        height = bar.get_height()
        ax.annotate(f'{height:.2f}', xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, 3), textcoords="offset points", ha='center', va='bottom', fontsize=8)

add_value_labels(bars1)
add_value_labels(bars2)
plt.tight_layout()
plt.savefig('feature_coefficients_comparison.png')
plt.show()

sample_data = X_test.iloc[:5]
sample_scaled = scaler.transform(sample_data)
linear_predictions = linear_model.predict(sample_data)
ridge_predictions = ridge_model.predict(sample_scaled)
print("\nExample Predictions:")
for i in range(5):
    print(f"Sample {i+1}: Actual = {y_test.iloc[i]:.2f}, Linear Prediction = {linear_predictions[i]:.2f}, Ridge Prediction = {ridge
```

Description

we implemented both Linear Regression and Ridge Regression models using the California Housing dataset to predict median house values. Missing data was handled using mean imputation, and feature scaling was applied to normalize the input features.
The models were compared using Mean Squared Error (MSE) and R-squared values to analyze performance. Ridge Regression showed improved generalization by reducing overfitting, and feature importance was visualized using coefficient comparison plots.

Task 2

Logistic Regression (Breast Cancer Dataset)

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, classification_report, accuracy_score, roc_curve, roc_auc_score
from matplotlib.colors import ListedColormap

cancer = load_breast_cancer()
X = cancer.data[:, :2]
y = cancer.target

df = pd.DataFrame(X, columns=cancer.feature_names[:2])
df['target'] = y
print("First 5 rows of the dataset:")
print(df.head())
print("\nTarget variable distribution:")
print(df['target'].value_counts())

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=0)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)

classifier = LogisticRegression(random_state=0)
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)
y_prob = classifier.predict_proba(X_test)[:, 1]
```

```
print("\nConfusion Matrix:")
cm = confusion_matrix(y_test, y_pred)
print(cm)

accuracy = accuracy_score(y_test, y_pred)
print(f"\nAccuracy: {accuracy:.4f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))

fpr, tpr, thresholds = roc_curve(y_test, y_prob)
auc = roc_auc_score(y_test, y_prob)
print(f"AUC: {auc:.4f}")

plt.figure(figsize=(10, 6))
plt.plot(fpr, tpr, color='orange', label=f'ROC Curve (AUC = {auc:.4f})')
plt.plot([0, 1], [0, 1], color='blue', linestyle='--')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver Operating Characteristic (ROC) Curve')
plt.legend()
plt.grid(True)
plt.savefig('roc_curve.png')
plt.show()

def plot_decision_boundary(X_set, y_set, title, filename):
    X1, X2 = np.meshgrid(np.arange(start=X_set[:, 0].min() - 1, stop=X_set[:, 0].max() + 1, step=0.01),
                         np.arange(start=X_set[:, 1].min() - 1, stop=X_set[:, 1].max() + 1, step=0.01))
    plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
                 alpha=0.75, cmap=ListedColormap(('red', 'green')))
    plt.xlim(X1.min(), X1.max())
    plt.ylim(X2.min(), X2.max())
    for i, j in enumerate(np.unique(y_set)):
        plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                    c=ListedColormap(('red', 'green'))(i), label=cancer.target_names[j])
    plt.title(title)
    plt.xlabel(cancer.feature_names[0])
    plt.ylabel(cancer.feature_names[1])
    plt.legend()
    plt.savefig(filename)
    plt.show()

plot_decision_boundary(X_train, y_train, 'Logistic Regression (Training set)', 'training_set_results.png')
plot_decision_boundary(X_test, y_test, 'Logistic Regression (Test set)', 'test_set_results.png')

sample_indices = [0, 5, 10, 15, 20]
print("\nPrediction Examples:")
for i in sample_indices:
    actual = y_test[i]
    predicted = y_pred[i]
    probability = y_prob[i]
    print(f"Sample {i}: Actual Class = {cancer.target_names[actual]}, Predicted Class = {cancer.target_names[predicted]}, Malignar
```

Description

we implemented both Linear Regression and Ridge Regression models using the California Housing dataset to predict median house values. Missing data was handled using mean imputation, and feature scaling was applied to normalize the input features. The models were compared using Mean Squared Error (MSE) and R-squared values to analyze performance. Ridge Regression showed improved generalization by reducing overfitting, and feature importance was visualized using coefficient comparison plots.

Summary

1. Implemented Linear Regression on the Student Performance dataset to predict students' performance index using multiple input features.
2. Encoded categorical data, split the dataset, and evaluated the model using Mean Squared Error (MSE) and R-squared to measure prediction accuracy.

3. Enhanced the model using Feature Scaling and Ridge Regression on the California Housing dataset to reduce overfitting and improve generalization.
4. Compared Linear vs Ridge Regression, showing that Ridge performs better when features are correlated or scaled differently.
5. Applied Logistic Regression on the Breast Cancer dataset for binary classification (malignant vs benign).
6. Evaluated the classifier using Confusion Matrix, Accuracy, Precision, Recall, F1-score, and ROC-AUC metrics.
7. Visualized decision boundaries and ROC curves to better understand model behavior and performance.