Part 1

Soln1)

```
From sklearn.datasets import fetch_california_housing

From sklearn.linear_model import LinearRegression

Import matplotlib.pyplot as plt

# Load dataset

Housing = fetch_california_housing()

Rooms = housing.data[:, 3].reshape(-1, 1)  # AveRooms

Prices = housing.target  # Median House Value


# Train linear regression model

Model = LinearRegression()

Model.fit(rooms, prices)

# Predict and plot

Predicted_prices = model.predict(rooms)


Plt.scatter(rooms, prices, color=’skyblue’, alpha=0.5, label=’Actual Prices’)

Plt.plot(rooms, predicted_prices, color=’crimson’, linewidth=2, label=’Regression Line’)

Plt.xlabel(‘Average Rooms per Household’)

Plt.ylabel(‘Median House Value’)

Plt.title(‘Simple Linear Regression’)

Plt.legend()

Plt.show()
```

Soln2)

```
From sklearn.model_selection import train_test_split
From sklearn.metrics import r2_score, mean_squared_error
Import numpy as np

# Select numeric features: MedInc, HouseAge, AveRooms, AveOccup
Features = housing.data[:, [0, 1, 3, 5]]
Target = housing.target
# Split dataset
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2,
random_state=42)

# Train model
Model = LinearRegression()
Model.fit(X_train, y_train)

# Predict and evaluate
Y_pred = model.predict(X_test)
R2 = r2_score(y_test, y_pred)
Mse = mean_squared_error(y_test, y_pred)
Rmse = np.sqrt(mse)
Print("R-squared:", r2)
Print("Mean Squared Error:", mse)
Print("Root Mean Squared Error:", rmse)
Print("Feature Coefficients:", model.coef_)
```

Soln 3)

```
from sklearn.preprocessing import StandardScaler
model_unscaled = LinearRegression()
model_unscaled.fit(X_train, y_train)
r2_unscaled = model_unscaled.score(X_test, y_test)
# With scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
model_scaled = LinearRegression()
model_scaled.fit(X_train_scaled, y_train)
r2_scaled = model_scaled.score(X_test_scaled, y_test)
print("R² without Scaling:", r2_unscaled)
print("R² with Scaling:", r2_scaled)
```

soln 4)

```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Convert data to DataFrame for easier analysis
Df = pd.DataFrame(housing.data, columns=housing.feature_names)
Df['MedHouseVal'] = housing.target  # Add target column
# Compute correlation matrix
Corr_matrix = df.corr()

# Plot heatmap
```

Plt.figure(figsize=(10, 8))

Sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', fmt=".2f")

Plt.title('Correlation Matrix Heatmap')

Plt.show()

Part 2

Soln 5)

From sklearn.datasets import load_breast_cancer

From sklearn.linear_model import LogisticRegression

From sklearn.model_selection import train_test_split

From sklearn.metrics import accuracy_score, confusion_matrix, classification_report, roc_auc_score, roc_curve

Import matplotlib.pyplot as plt

Load dataset

Data = load_breast_cancer()

X = data.data

Y = data.target

Split dataset

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

Train logistic regression model

Model = LogisticRegression(max_iter=1000)

Model.fit(X_train, y_train)

```
# Predict and evaluate

Y_pred = model.predict(X_test)

Y_probs = model.predict_proba(X_test)[:, 1]


# Evaluation metrics

Print("Accuracy:", accuracy_score(y_test, y_pred))

Print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

Print("Classification Report:\n", classification_report(y_test, y_pred))

Print("ROC-AUC Score:", roc_auc_score(y_test, y_probs))


# Plot ROC Curve

Fpr, tpr, thresholds = roc_curve(y_test, y_probs)

Plt.plot(fpr, tpr, label="ROC Curve")

Plt.plot([0, 1], [0, 1], 'k—')  # Diagonal

Plt.xlabel("False Positive Rate")

Plt.ylabel("True Positive Rate")

Plt.title("ROC Curve")

Plt.legend()

Plt.show()
```

Soln 6)

```
From sklearn.metrics import f1_score


For threshold in [0.3, 0.5, 0.7]:

   Custom_pred = (y_probs >= threshold).astype(int)

   Cm = confusion_matrix(y_test, custom_pred)
```

```
    F1 = f1_score(y_test, custom_pred)


    Print(f"Threshold: {threshold}")
    Print("Confusion Matrix:\n", cm)
    Print("F1 Score:", f1)
    Print("-" * 30)


# Plot ROC Curve with threshold marker
Plt.plot(fpr, tpr, label="ROC Curve")
Plt.scatter(fpr[np.argmax(tpr – fpr)], tpr[np.argmax(tpr – fpr)],
        Color='red', label='Optimal Threshold')
Plt.xlabel("False Positive Rate")
Plt.ylabel("True Positive Rate")
Plt.title("ROC Curve with Optimal Threshold")
Plt.legend()
Plt.show()
```

PART 3

1)Linear Regression Assumptions: Linearity, independence of errors, homoscedasticity, and normally distributed residuals.

2)Use Logistic Regression: When the target variable is categorical, especially binary (e.g., yes/no).

3) Coefficients in Logistic Regression: Indicate the change in log odds for a one-unit increase in the predictor.

4) Sigmoid vs. Softmax: Sigmoid outputs a single probability (binary); softmax gives a probability distribution across multiple classes.

5) R-squared in Logistic Models: It doesn’t apply because logistic regression predicts probabilities, not continuous values.