Name: Shrey Santosh Rupnavar **Division:** SY CSE AI B **Roll No:** 33 **Batch:** 2 **PRN No:** 12310543

1] LINEAR REGRESSION

```
#importing neccesary libraries
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
from sklearn.metrics import mean_squared_error, r2_score
```

```
#Loading the dataset
california = fetch_california_housing()
df = pd.DataFrame(california.data, columns=california.feature_names)
df['MedHouseVal'] = california.target
```

```
#X variable is set to average rooms per household col & y is set to median house val
X = df[['AveRooms']]
y = df['MedHouseVal']
```

```
#Splliting into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Training the model
linear_regressor = LinearRegression()
linear_regressor.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

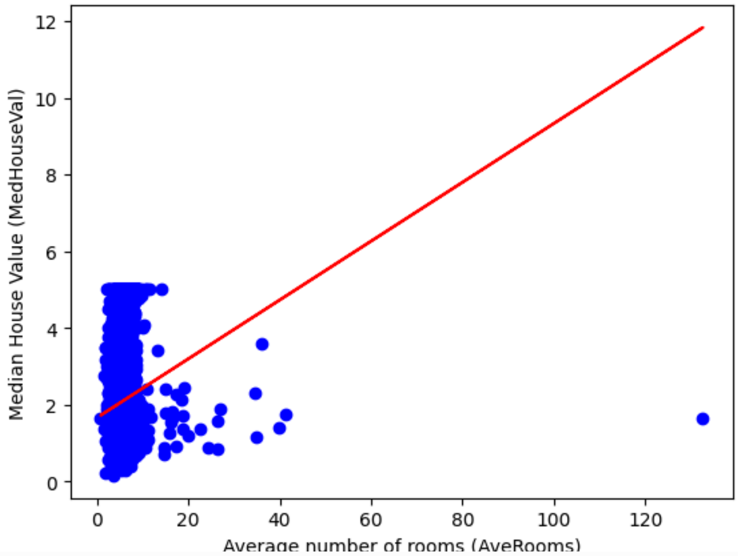
```
#Making predictions
y_pred = linear_regressor.predict(X_test)
```

```
#Model evaluation by calculating mean squared error and r2 score
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 1.2923314440807299
R-squared: 0.013795337532284901
```

```
#Plotting regression line
plt.scatter(X_test, y_test, color='blue')
plt.plot(X_test, y_pred, color='red')
plt.xlabel('Average number of rooms (AveRooms)')
plt.ylabel('Median House Value (MedHouseVal)')
plt.show()
```



2] MULTIPLE LINEAR REGRESSION

```
#Selecting multiple columns
X = df[['AveRooms', 'AveOccup', 'HouseAge']]
y = df['MedHouseVal']
```

```
#Again splliting into train and test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

```
#Training the model
multiple_linear_regressor = LinearRegression()
multiple_linear_regressor.fit(X_train, y_train)
```

```
▾ LinearRegression
LinearRegression()
```

```
#Making multiple predictions
y_pred = multiple_linear_regressor.predict(X_test)
```

```
#Evaluating the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)
```

```
#Printing errors
print(f"Mean Squared Error: {mse}")
print(f"R-squared: {r2}")
```

```
Mean Squared Error: 1.2699545224857287
R-squared: 0.030871625902225697
```

3] LOGISTIC REGRESSION USING SYNTHETIC DATASET

```
#Importing additional libraries
from sklearn.datasets import make_classification
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
import seaborn as sns
```

```
#Generating synthetic dataset
X, y = make_classification(n_samples=1000, n_features=3, n_informative=2, n_redundant=1, n_clusters_per_class=1, random_state=42)
```

```
#Converting synthetic dataset to dataframe for better handling
df = pd.DataFrame(X, columns=['Feature1', 'Feature2', 'Feature3'])
df['Target'] = y
```

```
#Splliting into train & test data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

#Training the logistic regression model
logistic_regressor = LogisticRegression()
logistic_regressor.fit(X_train, y_train)
```

```
▾ LogisticRegression
LogisticRegression()
```

```
#Making predictions on test data
y_pred = logistic_regressor.predict(X_test)


#Evaluating the model for results
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
class_report = classification_report(y_test, y_pred)


#The results:
print(f"Accuracy: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(class_report)
```

```
Accuracy: 0.91

Confusion Matrix:
[[94  8]
 [11 87]]

Classification Report:
              precision    recall  f1-score   support

           0       0.90      0.92      0.91       102
           1       0.92      0.89      0.90        98

    accuracy                           0.91       200
   macro avg       0.91      0.90      0.90       200
weighted avg       0.91      0.91      0.90       200
```

```
#Visualizing the confusion matrix
sns.heatmap(conf_matrix, annot=True, fmt='d', cmap='Blues')
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```