Tut 2: Apply Linear Regression,Multilinear regression Logistic regresssion for suitable data set.

Name - Tanishq Thuse

Year - SY

Branch - CSE(AI)

Div - B

Roll no. - 60

Importing Neccesary Libraries

```
#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
```

EDA (Exploratory Data Analysis)

```
#Loading the dataset
california = fetch_california_housing()
#The above method is very much helpfull as this helps us get the dataset without
#uploading it again and again on collab
df = pd.DataFrame(california.data, columns=california.feature_names)
```

```
df.head()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
0	8.3252	41.0	6.984127	1.023810	322.0	2.555556	37.88	-122.23	4.526
1	8.3014	21.0	6.238137	0.971880	2401.0	2.109842	37.86	-122.22	3.585
2	7.2574	52.0	8.288136	1.073446	496.0	2.802260	37.85	-122.24	3.521
3	5.6431	52.0	5.817352	1.073059	558.0	2.547945	37.85	-122.25	3.413
4	3.8462	52.0	6.281853	1.081081	565.0	2.181467	37.85	-122.25	3.422

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
df.tail()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
20635	1.5603	25.0	5.045455	1.133333	845.0	2.560606	39.48	-121.09	0.781
20636	2.5568	18.0	6.114035	1.315789	356.0	3.122807	39.49	-121.21	0.771
20637	1.7000	17.0	5.205543	1.120092	1007.0	2.325635	39.43	-121.22	0.923
20638	1.8672	18.0	5.329513	1.171920	741.0	2.123209	39.43	-121.32	0.847
20639	2.3886	16.0	5.254717	1.162264	1387.0	2.616981	39.37	-121.24	0.894

```
df.describe()
```

	MedInc	HouseAge	AveRooms	AveBedrms	Population	AveOccup	Latitude	Longitude	MedHouseVal
count	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000	20640.000000
mean	3.870671	28.639486	5.429000	1.096675	1425.476744	3.070655	35.631861	-119.569704	2.068558
std	1.899822	12.585558	2.474173	0.473911	1132.462122	10.386050	2.135952	2.003532	1.153956
min	0.499900	1.000000	0.846154	0.333333	3.000000	0.692308	32.540000	-124.350000	0.149990
25%	2.563400	18.000000	4.440716	1.006079	787.000000	2.429741	33.930000	-121.800000	1.196000
50%	3.534800	29.000000	5.229129	1.048780	1166.000000	2.818116	34.260000	-118.490000	1.797000
75%	4.743250	37.000000	6.052381	1.099526	1725.000000	3.282261	37.710000	-118.010000	2.647250
max	15.000100	52.000000	141.909091	34.066667	35682.000000	1243.333333	41.950000	-114.310000	5.000010

LINEAR REGRESSION

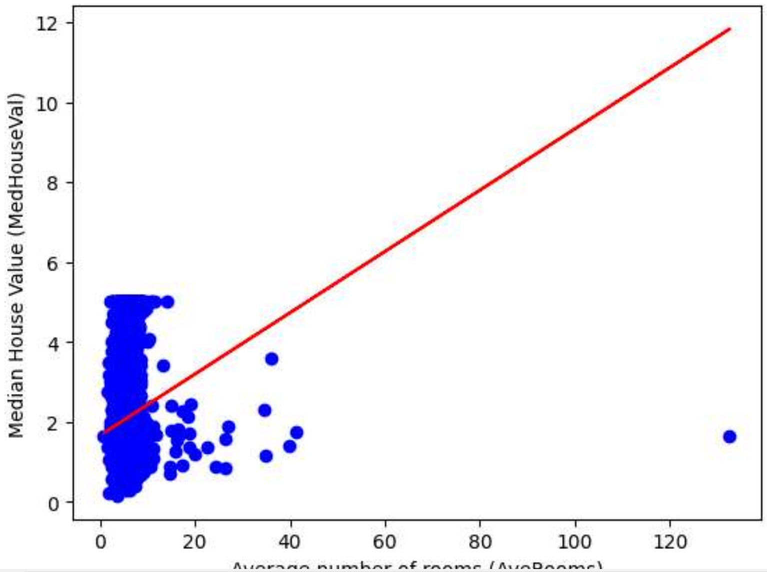
```
df['MedHouseVal'] = california.target


#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()
```



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
```

LOGISTIC REGRESSION

```
#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
```

```
#Here I am using synthetic datasetX, y = make_classification(n_samples=1000, n_features=3, n_informative=2, n_redundant=1, n_clusters_
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


df.head()
```

	Feature1	Feature2	Feature3	Target
0	0.324689	1.682530	-0.381186	1
1	0.993077	0.755945	-1.172352	0
2	0.804408	1.354479	-0.948528	0
3	-0.193718	3.103090	0.233485	0
4	1.582040	1.086596	1.871016	1

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
#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)
```

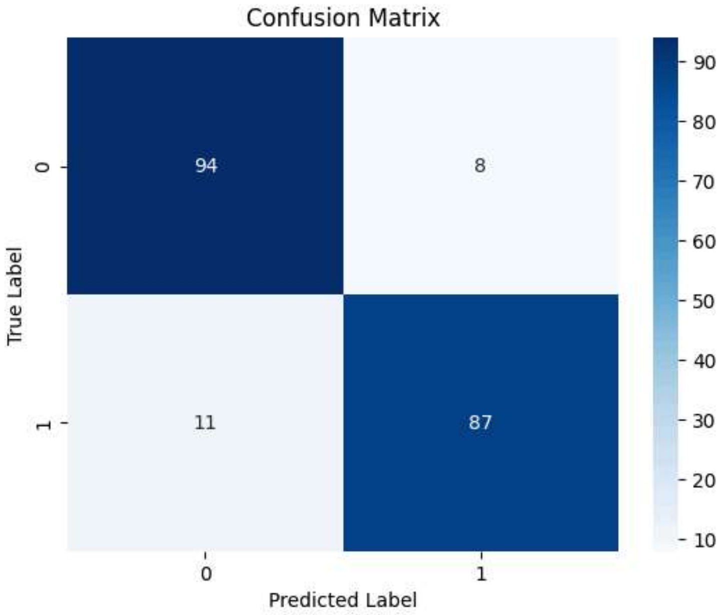
```
▾ LogisticRegression
LogisticRegression()
```

```
#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()
```

```
    plt.show()
```


Confusion Matrix
True Label
0
1
94
8
11
87
0
1
Predicted Label
90
80
70
60
50
40
30
20
10