In [1]:
```
import numpy as np
import pandas as pd

from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
from sklearn.metrics import classification_report, confusion_matrix
from sklearn.metrics import mean_absolute_error,mean_squared_error
```

In [3]:
```
col_names = ['pregnant', 'glucose', 'bp', 'skin', 'insulin', 'bmi', 'pedigree', 'ag

# Load dataset
pima = pd.read_csv("pima-indians-diabetes.csv", header = 0, names = col_names)
pima.head(10)
```

Out[3]:

	pregnant	glucose	bp	skin	insulin	bmi	pedigree	age	label
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1
5	5	116	74	0	0	25.6	0.201	30	0
6	3	78	50	32	88	31.0	0.248	26	1
7	10	115	0	0	0	35.3	0.134	29	0
8	2	197	70	45	543	30.5	0.158	53	1
9	8	125	96	0	0	0.0	0.232	54	1

In [5]:
```
exam = pd.read_csv("exam_scores.csv", header = 0)
exam.head(10)
```

Out[5]:

	Exam Score1	Exam Score2	Pass
0	22.99	43.42	0
1	4.32	64.49	0
2	52.59	52.69	1
3	11.90	24.99	0
4	52.37	45.93	0
5	18.60	12.89	0
6	24.38	80.38	0
7	74.71	61.49	1
8	79.42	67.92	1
9	62.75	97.53	1

In [7]:

```
car = pd.read_csv("car_data.csv", header = 0)
car.loc[car["Gender"] == "Male", "Gender"] = 1
car.loc[car["Gender"] == "Female", "Gender"] = 1
car.head(10)
```

Out[7]:

	User ID	Gender	Age	AnnualSalary	Purchased
0	385	1	35	20000	0
1	681	1	40	43500	0
2	353	1	49	74000	0
3	895	1	40	107500	1
4	661	1	25	79000	0
5	846	1	47	33500	1
6	219	1	46	132500	1
7	588	1	42	64000	0
8	85	1	30	84500	0
9	465	1	41	52000	0

In [9]:

```
ad = pd.read_csv("advertising.csv", header = 0)
ad = ad.rename(columns={'Clicked on Ad': 'Clicked'})
ad.head(10)
```

Out[9]:

	Daily Time Spent on Site	Age	Area Income	Daily Internet Usage	Ad Topic Line	City	Male	Country	Timestam
0	68.95	35	61833.90	256.09	Cloned 5thgeneration orchestration	Wrightburgh	0	Tunisia	2016-03-2 00:53:1
1	80.23	31	68441.85	193.77	Monitored national standardization	West Jodi	1	Nauru	2016-04-0 01:39:0
2	69.47	26	59785.94	236.50	Organic bottom-line service-desk	Davidton	0	San Marino	2016-03-1 20:35:4
3	74.15	29	54806.18	245.89	Triple-buffered reciprocal time-frame	West Terrifurt	1	Italy	2016-01-1 02:31:1
4	68.37	35	73889.99	225.58	Robust logistical utilization	South Manuel	0	Iceland	2016-06-0 03:36:1
5	59.99	23	59761.56	226.74	Sharable client-driven software	Jamieberg	1	Norway	2016-05-1 14:30:1
6	88.91	33	53852.85	208.36	Enhanced dedicated support	Brandonstad	0	Myanmar	2016-01-2 20:59:3
7	66.00	48	24593.33	131.76	Reactive local challenge	Port Jefferybury	1	Australia	2016-03-0 01:40:1
8	74.53	30	68862.00	221.51	Configurable coherent function	West Colin	1	Grenada	2016-04-1 09:33:4
9	69.88	20	55642.32	183.82	Mandatory homogeneous architecture	Ramirezton	1	Ghana	2016-07-1 01:42:5

In [11]:

```
# split dataset in features and target variable
pima_feature_cols = ['pregnant', 'insulin', 'bmi', 'age', 'glucose', 'bp', 'pedigre

pima_X = pima[pima_feature_cols]  # Features
pima_y = pima.label          # Target variable

exam_feature_cols = ['Exam Score1', 'Exam Score2']

exam_X = exam[exam_feature_cols]
exam_y = exam.Pass
```

```
car_feature_cols = ['Gender', 'Age', 'AnnualSalary']

car_X = car[car_feature_cols]
car_y = car.Purchased

ad_feature_cols = ['Daily Time Spent on Site', 'Age', 'Area Income', 'Daily Interne

ad_X = ad[ad_feature_cols]
ad_y = ad.Clicked
```

In [13]:

```
# split X and y into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(pima_X, pima_y, test_size=0.25,

print("Results for diabetes data")

# instantiate the model (using the default parameters)
model = LogisticRegression(random_state = 16, max_iter = 200)
model.fit(X_train, y_train)

print(model.score(X_test, y_test))

# Test using test set
y_test_pred = model.predict(X_test)

mae = mean_absolute_error(y_true=y_test,y_pred=y_test_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_test_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_test_pred,squared=False)

print("MAE: ", mae)
print("MSE: ", mse)
print("RMSE: ", rmse)

target_names = ['Without Diabetes', 'With diabetes']

report = classification_report(y_test, y_test_pred, target_names = target_names)
print("\nReport for diabetes data\n",report)
```

```
Results for diabetes data
0.8177083333333334
MAE:  0.18229166666666666
MSE:  0.18229166666666666
RMSE:  0.42695628191498325

Report for diabetes data
                   precision    recall  f1-score   support

Without Diabetes       0.82      0.92      0.87       125
   With diabetes       0.81      0.63      0.71        67

        accuracy                           0.82       192
       macro avg       0.81      0.77      0.79       192
    weighted avg       0.82      0.82      0.81       192
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:492: Futur
eWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calc
ulate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

In [15]:
```
X_train, X_test, y_train, y_test = train_test_split(exam_X, exam_y, test_size=0.25,

print("Results for exam data")

# instantiate the model (using the default parameters)
model = LogisticRegression(random_state = 16, max_iter = 200)
model.fit(X_train, y_train)

print(model.score(X_test, y_test))

# Test using test set
y_test_pred = model.predict(X_test)

mae = mean_absolute_error(y_true=y_test,y_pred=y_test_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_test_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_test_pred,squared=False)

print("MAE: ", mae)
print("MSE: ", mse)
print("RMSE: ", rmse)

target_names = ['Pass', 'Fail']

report = classification_report(y_test, y_test_pred, target_names = target_names)
print("\nReport for exam data\n",report)
```

```
Results for exam data
0.92
MAE:  0.08
MSE:  0.08
RMSE:  0.282842712474619

Report for exam data
               precision    recall  f1-score   support

        Pass       0.93      0.97      0.95        77
        Fail       0.89      0.74      0.81        23

    accuracy                           0.92       100
   macro avg       0.91      0.86      0.88       100
weighted avg       0.92      0.92      0.92       100
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:492: Futur
eWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calc
ulate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

In [17]:
```
X_train, X_test, y_train, y_test = train_test_split(car_X, car_y, test_size=0.25, r

print("Results for car purchasing data")
```

```
# instantiate the model (using the default parameters)
model = LogisticRegression(random_state = 16, max_iter = 200)
model.fit(X_train, y_train)

print(model.score(X_test, y_test))

# Test using test set
y_test_pred = model.predict(X_test)

mae = mean_absolute_error(y_true=y_test,y_pred=y_test_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_test_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_test_pred,squared=False)

print("MAE: ", mae)
print("MSE: ", mse)
print("RMSE: ", rmse)

target_names = ['Purchased', 'Not Purchased']

report = classification_report(y_test, y_test_pred, target_names = target_names)
print("\nReport for car purchasing data\n",report)
```

```
Results for car purchasing data
0.832
MAE:  0.168
MSE:  0.168
RMSE:  0.40987803063838396

Report for car purchasing data
                precision    recall  f1-score   support

    Purchased       0.84      0.90      0.87       156
Not Purchased       0.81      0.72      0.76        94

     accuracy                           0.83       250
    macro avg       0.83      0.81      0.82       250
 weighted avg       0.83      0.83      0.83       250
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:492: Futur
eWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calc
ulate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```

In [21]:
```
X_train, X_test, y_train, y_test = train_test_split(ad_X, ad_y, test_size=0.25, ran

print("Results for ad clicking data")

# instantiate the model (using the default parameters)
model = LogisticRegression(random_state = 16, max_iter = 1000000000)
model.fit(X_train, y_train)

print(model.score(X_test, y_test))

# Test using test set
y_test_pred = model.predict(X_test)
```

```
mae = mean_absolute_error(y_true=y_test,y_pred=y_test_pred)
mse = mean_squared_error(y_true=y_test,y_pred=y_test_pred) #default=True
rmse = mean_squared_error(y_true=y_test,y_pred=y_test_pred,squared=False)

print("MAE: ", mae)
print("MSE: ", mse)
print("RMSE: ", rmse)

target_names = ['Clicked', 'Not Clicked']

report = classification_report(y_test, y_test_pred, target_names = target_names)
print("Report for ad clicking data\n",report)
```

```
Results for ad clicking data
0.948
MAE:  0.052
MSE:  0.052
RMSE:  0.22803508501982758
Report for ad clicking data
               precision    recall  f1-score   support

     Clicked       0.92      0.98      0.95       125
 Not Clicked       0.97      0.92      0.95       125

    accuracy                           0.95       250
   macro avg       0.95      0.95      0.95       250
weighted avg       0.95      0.95      0.95       250
```

```
C:\ProgramData\anaconda3\Lib\site-packages\sklearn\metrics\_regression.py:492: Futur
eWarning: 'squared' is deprecated in version 1.4 and will be removed in 1.6. To calc
ulate the root mean squared error, use the function'root_mean_squared_error'.
  warnings.warn(
```