Import Libraries

In [46]:
```
import pandas as pd
import numpy as np
```

Reading Dataset

In [49]:
```
df = pd.read_csv("C:/Users/User/Downloads/Telco-Customer-Churn.csv")
#https://drive.google.com/file/d/15DiFz3J0XezApOjYZIon7ozgFnYH-AyE/view?usp=sharing
```

The code df.head() is used to display the first few rows of a DataFrame in pandas. It allows you to quickly inspect the data and get an overview of its structure and contents.

In [52]:
```
df.head()
```

Out[52]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSupport	StreamingTV	StreamingMovies	Contract	PaperlessBilling
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	No	No	No	Month-to-month	Yes
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	No	No	No	One year	No
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	No	No	No	Month-to-month	Yes
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	Yes	No	No	One year	No
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	No	No	No	Month-to-month	Yes

5 rows × 21 columns

Exploring data

In [53]:
```
df.info
```

Out[53]:
```
<bound method DataFrame.info of       customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
...          ...     ...            ...     ...        ...     ...
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
0              No  No phone service             DSL             No  ...
1             Yes                No             DSL            Yes  ...
2             Yes                No             DSL            Yes  ...
3              No  No phone service             DSL            Yes  ...
4             Yes                No     Fiber optic             No  ...
...           ...               ...             ...            ...  ...
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95        1889.5
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
...               ...                        ...            ...           ...
7038              Yes               Mailed check          84.80        1990.5
7039              Yes    Credit card (automatic)         103.20        7362.9
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40         306.6
7042              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]>
```

The code df.dtypes is used to display the data types of each column in a pandas DataFrame.

Alt+Q

In [6]:
```
df.dtypes
```

Out[6]:
```
customerID           object
gender               object
SeniorCitizen         int64
Partner              object
Dependents           object
tenure                int64
PhoneService         object
MultipleLines        object
InternetService      object
OnlineSecurity       object
OnlineBackup         object
DeviceProtection     object
TechSupport          object
StreamingTV          object
StreamingMovies      object
Contract             object
PaperlessBilling     object
PaymentMethod        object
MonthlyCharges      float64
TotalCharges         object
Churn                object
dtype: object
```

The code df.describe() is used to generate descriptive statistics of a pandas DataFrame. It provides a summary of the central tendency, dispersion, and shape of the numerical columns in the DataFrame.

In [7]:
```
df.describe()
```

Out[7]:

	SeniorCitizen	tenure	MonthlyCharges
count	7043.000000	7043.000000	7043.000000
mean	0.162147	32.371149	64.761692
std	0.368612	24.559481	30.090047
min	0.000000	0.000000	18.250000
25%	0.000000	9.000000	35.500000
50%	0.000000	29.000000	70.350000
75%	0.000000	55.000000	89.850000
max	1.000000	72.000000	118.750000

The code below is used to check for missing values in each column of a pandas DataFrame and calculate the total number of missing values per column.

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

The code below is used to identify and count the number of duplicated rows in a pandas DataFrame.

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

The code df.dropna(how="any", inplace=True) is used to remove rows with missing values from a pandas DataFrame.
The dropna() function is applied to the DataFrame df with the parameter how="any", which means that if any of the values in a row are missing, that entire row will be dropped. The inplace=True parameter ensures that the changes are made directly to the DataFrame df without creating a new DataFrame.

The code df.shape is then used to retrieve the new dimensions of the DataFrame after the removal of missing values. df.shape returns a tuple representing the number of rows and columns in the DataFrame.

In [10]:
```
df.dropna(how="any",inplace=True)
df.shape
```

Out[10]:
```
(7043, 21)
```

Calling df.isnull().sum() will display the count of missing values for each column in the modified DataFrame.

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

The code df.columns.values is used to retrieve the column names of a pandas DataFrame df as an array-like object.

In [12]:
```
df.columns.values
```

Out[12]:
```
array(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection',
       'TechSupport', 'StreamingTV', 'StreamingMovies', 'Contract',
       'PaperlessBilling', 'PaymentMethod', 'MonthlyCharges',
       'TotalCharges', 'Churn'], dtype=object)
```

The attribute df.columns is used to retrieve the column names of a pandas DataFrame df as an Index object.

In [13]:
```
df.columns
```

Out[13]:
```
Index(['customerID', 'gender', 'SeniorCitizen', 'Partner', 'Dependents',
       'tenure', 'PhoneService', 'MultipleLines', 'InternetService',
       'OnlineSecurity', 'OnlineBackup', 'DeviceProtection', 'TechSupport',
       'StreamingTV', 'StreamingMovies', 'Contract', 'PaperlessBilling',
       'PaymentMethod', 'MonthlyCharges', 'TotalCharges', 'Churn'],
      dtype='object')
```

The code df.tail() is used to display the last few rows of a pandas DataFrame df.

In [14]:
```
df.tail
```

Out[14]:
```
<bound method NDFrame.tail of       customerID  gender  SeniorCitizen Partner Dependents  tenure  \
0     7590-VHVEG  Female              0     Yes         No       1
1     5575-GNVDE    Male              0      No         No      34
2     3668-QPYBK    Male              0      No         No       2
3     7795-CFOCW    Male              0      No         No      45
4     9237-HQITU  Female              0      No         No       2
...          ...     ...            ...     ...        ...     ...
7038  6840-RESVB    Male              0     Yes        Yes      24
7039  2234-XADUH  Female              0     Yes        Yes      72
7040  4801-JZAZL  Female              0     Yes        Yes      11
7041  8361-LTMKD    Male              1     Yes         No       4
7042  3186-AJIEK    Male              0      No         No      66

     PhoneService     MultipleLines InternetService OnlineSecurity  ...  \
0              No  No phone service             DSL             No  ...
1             Yes                No             DSL            Yes  ...
2             Yes                No             DSL            Yes  ...
3              No  No phone service             DSL            Yes  ...
4             Yes                No     Fiber optic             No  ...
...           ...               ...             ...            ...  ...
7038          Yes               Yes             DSL            Yes  ...
7039          Yes               Yes     Fiber optic             No  ...
7040           No  No phone service             DSL            Yes  ...
7041          Yes               Yes     Fiber optic             No  ...
7042          Yes                No     Fiber optic            Yes  ...

     DeviceProtection TechSupport StreamingTV StreamingMovies        Contract  \
0                  No          No          No              No  Month-to-month
1                 Yes          No          No              No        One year
2                  No          No          No              No  Month-to-month
3                 Yes         Yes          No              No        One year
4                  No          No          No              No  Month-to-month
...               ...         ...         ...             ...             ...
7038              Yes         Yes         Yes             Yes        One year
7039              Yes          No         Yes             Yes        One year
7040               No          No          No              No  Month-to-month
7041               No          No          No              No  Month-to-month
7042              Yes         Yes         Yes             Yes        Two year

     PaperlessBilling              PaymentMethod MonthlyCharges  TotalCharges  \
0                 Yes           Electronic check          29.85         29.85
1                  No               Mailed check          56.95        1889.5
2                 Yes               Mailed check          53.85        108.15
3                  No  Bank transfer (automatic)          42.30       1840.75
4                 Yes           Electronic check          70.70        151.65
...               ...                        ...            ...           ...
7038              Yes               Mailed check          84.80        1990.5
7039              Yes    Credit card (automatic)         103.20        7362.9
7040              Yes           Electronic check          29.60        346.45
7041              Yes               Mailed check          74.40         306.6
7042              Yes  Bank transfer (automatic)         105.65        6844.5

     Churn
0       No
1       No
2      Yes
3       No
4      Yes
...    ...
7038    No
7039    No
7040    No
7041   Yes
7042    No

[7043 rows x 21 columns]>
```

The code df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce') is used to convert the values in the "TotalCharges" column of a pandas DataFrame df from their current data type to numeric type.

In [15]:
```
df.TotalCharges = pd.to_numeric(df.TotalCharges, errors='coerce')
```

Check for missing values in each column of the DataFrame df.

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
customerID           0
gender               0
SeniorCitizen        0
Partner              0
Dependents           0
tenure               0
PhoneService         0
MultipleLines        0
InternetService      0
OnlineSecurity       0
OnlineBackup         0
DeviceProtection     0
TechSupport          0
StreamingTV          0
StreamingMovies      0
Contract             0
PaperlessBilling     0
PaymentMethod        0
MonthlyCharges       0
TotalCharges        11
Churn                0
dtype: int64
```

Remove rows with missing values from a pandas DataFrame df.

In [17]:
```
df.dropna(how="any",inplace=True)
```

Display the count of missing values for each column in the modified DataFrame.

In [18]:
```
df.isnull().sum()
```

Out[18]:
```
customerID          0
gender              0
SeniorCitizen       0
Partner             0
Dependents          0
tenure              0
PhoneService        0
MultipleLines       0
InternetService     0
OnlineSecurity      0
OnlineBackup        0
DeviceProtection    0
TechSupport         0
StreamingTV         0
StreamingMovies     0
Contract            0
PaperlessBilling    0
PaymentMethod       0
MonthlyCharges      0
TotalCharges        0
Churn               0
dtype: int64
```

In [19]:
```
from sklearn.preprocessing import LabelEncoder
```

By executing this code, you are encoding the labels in the "Churn" column as integers, which is a common step in preparing categorical data for machine learning algorithms. The encoded values will be numerical representations of the original labels, allowing them to be used as inputs for various machine learning models.

In [20]:
```
encoder = LabelEncoder()
df['Churn'] = encoder.fit_transform(df['Churn'])
Churn = {index : label for index, label in enumerate(encoder.classes_)}
Churn
```

Out[20]:
```
{0: 'No', 1: 'Yes'}
```

Encoding the labels in the "PaymentMethod" column as integers, similar to the previous code snippet. This encoding allows the categorical variable to be used as input in machine learning algorithms that require numerical inputs.

In [21]:
```
df['PaymentMethod'] = encoder.fit_transform(df['PaymentMethod'])
PaymentMethod = {index : label for index, label in enumerate(encoder.classes_)}
PaymentMethod
```

Out[21]:
```
{0: 'Bank transfer (automatic)',
 1: 'Credit card (automatic)',
 2: 'Electronic check',
 3: 'Mailed check'}
```

Encoding the labels in the "customerID" column as integers, similar to the previous code snippet. This encoding allows the categorical variable to be used as input in machine learning algorithms that require numerical inputs.

In [22]:
```
df['customerID'] = encoder.fit_transform(df['customerID'])
customerID = {index : label for index, label in enumerate(encoder.classes_)}
customerID
```

Out[22]:
```
{0: '0002-ORFBO',
 1: '0003-MKNFE',
 2: '0004-TLHLJ',
 3: '0011-IGKFF',
 4: '0013-EXCHZ',
 5: '0013-MHZWF',
 6: '0013-SMEOE',
 7: '0014-BMAQU',
 8: '0015-UOCOJ',
 9: '0016-QLJIS',
 10: '0017-DINOC',
 11: '0017-IUDMW',
 12: '0018-NYROU',
 13: '0019-EFAEP',
 14: '0019-GFNTW',
 15: '0020-INWCK',
 16: '0020-JDNXP',
 17: '0021-IKXGC',
 18: '0022-TCJCI',
```

In [23]:
```
df['gender'] = encoder.fit_transform(df['gender'])
gender = {index : label for index, label in enumerate(encoder.classes_)}
gender
```

Out[23]:
```
{0: 'Female', 1: 'Male'}
```

In [24]:
```
df['Partner'] = encoder.fit_transform(df['Partner'])
Partner = {index : label for index, label in enumerate(encoder.classes_)}
Partner
```

Out[24]:
```
{0: 'No', 1: 'Yes'}
```

In [25]:
```
df['Dependents'] = encoder.fit_transform(df['Dependents'])
Dependents = {index : label for index, label in enumerate(encoder.classes_)}
Dependents
```

Out[25]:
```
{0: 'No', 1: 'Yes'}
```

In [26]:
```
df['PhoneService'] = encoder.fit_transform(df['PhoneService'])
PhoneService = {index : label for index, label in enumerate(encoder.classes_)}
PhoneService
```

Out[26]:
```
{0: 'No', 1: 'Yes'}
```

In [27]:
```
df['MultipleLines'] = encoder.fit_transform(df['MultipleLines'])
MultipleLines = {index : label for index, label in enumerate(encoder.classes_)}
MultipleLines
```

Out[27]:
```
{0: 'No', 1: 'No phone service', 2: 'Yes'}
```

In [28]:
```
df['InternetService'] = encoder.fit_transform(df['InternetService'])
InternetService = {index : label for index, label in enumerate(encoder.classes_)}
InternetService
```

Out[28]:
```
{0: 'DSL', 1: 'Fiber optic', 2: 'No'}
```

In [29]:
```
df['OnlineSecurity'] = encoder.fit_transform(df['OnlineSecurity'])
OnlineSecurity = {index : label for index, label in enumerate(encoder.classes_)}
OnlineSecurity
```

Out[29]:
```
{0: 'No', 1: 'No internet service', 2: 'Yes'}
```

In [30]:
```
df['OnlineBackup'] = encoder.fit_transform(df['OnlineSecurity'])
OnlineSecurity = {index : label for index, label in enumerate(encoder.classes_)}
OnlineSecurity
```

Out[30]:
```
{0: 0, 1: 1, 2: 2}
```

In [31]:
```
df['DeviceProtection'] = encoder.fit_transform(df['DeviceProtection'])
DeviceProtection = {index : label for index, label in enumerate(encoder.classes_)}
DeviceProtection
```

Out[31]:
```
{0: 'No', 1: 'No internet service', 2: 'Yes'}
```

In [32]:
```
df['TechSupport'] = encoder.fit_transform(df['TechSupport'])
TechSupport = {index : label for index, label in enumerate(encoder.classes_)}
TechSupport
```

Out[32]:
```
{0: 'No', 1: 'No internet service', 2: 'Yes'}
```

In [33]:
```
df['StreamingTV'] = encoder.fit_transform(df['StreamingTV'])
StreamingTV = {index : label for index, label in enumerate(encoder.classes_)}
StreamingTV
```

Out[33]:
```
{0: 'No', 1: 'No internet service', 2: 'Yes'}
```

In [34]:
```
df['StreamingMovies'] = encoder.fit_transform(df['StreamingMovies'])
StreamingMovies = {index : label for index, label in enumerate(encoder.classes_)}
StreamingMovies
```

Out[34]:
```
{0: 'No', 1: 'No internet service', 2: 'Yes'}
```

In [35]:
```
df['Contract'] = encoder.fit_transform(df['Contract'])
Contract = {index : label for index, label in enumerate(encoder.classes_)}
Contract
```

Out[35]:
```
{0: 'Month-to-month', 1: 'One year', 2: 'Two year'}
```

In [36]:
```
df['PaperlessBilling'] = encoder.fit_transform(df['PaperlessBilling'])
PaperlessBilling = {index : label for index, label in enumerate(encoder.classes_)}
PaperlessBilling
```

Out[36]:
```
{0: 'No', 1: 'Yes'}
```

Separate the target variable y from the feature matrix X. This separation allows you to perform modeling or predictive analysis on the features in X to predict the target variable y.

In [37]:
```
y = df['Churn'].values
X = df.drop(columns = ['Churn'])
```

Scale the values in the feature matrix X using the minimum and maximum values of each feature. This scaling ensures that all the features are on a similar scale, which can be beneficial for certain machine learning algorithms that are sensitive to the scale of the features.

In [38]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(copy=True, feature_range=(0, 1))
X = scaler.fit_transform(X)

#showing data
print('x \n' , X[:10])
print('y \n' , y[:10])
```

```
x
 [[0.76304935 0.         0.         1.         0.         0.
  0.         0.5        0.         0.         0.         0.
  0.         0.         0.         0.         1.         0.66666667
  0.11542289 0.0012751 ]
 [0.56222443 1.         0.         0.         0.         0.46478873
  1.         0.         0.         1.         1.         1.
  0.         0.         0.         0.5        0.         1.
  0.38507463 0.21586661]
 [0.36381738 1.         0.         0.         0.         0.01408451
  1.         0.         0.         1.         1.         0.
  0.         0.         0.         0.         1.         1.
  0.35422886 0.01031041]
 [0.78566349 1.         0.         0.         0.         0.61971831
  0.         0.5        0.         1.         1.         1.
  1.         0.         0.         0.5        0.         0.
  0.23930348 0.21024117]
 [0.92447731 0.         0.         0.         0.         0.01408451
  1.         0.         0.5        0.         0.         0.
  0.         0.         0.         0.         1.         0.66666667
  0.52189055 0.01533003]
 [0.93016641 0.         0.         0.         0.         0.09859155
  1.         1.         0.5        0.         0.         1.
  0.         1.         1.         0.         1.         0.66666667
  0.80995025 0.09251096]
 [0.14236951 1.         0.         0.         1.         0.29577465
  1.         1.         0.5        0.         0.         0.
  0.         1.         0.         0.         1.         0.33333333
  0.70497512 0.22277868]
 [0.67700185 0.         0.         0.         0.         0.12676056
  0.         0.5        0.         1.         1.         0.
  0.         0.         0.         0.         0.         1.
  0.11442786 0.0326679 ]
 [0.79547717 0.         0.         1.         0.         0.38028169
  1.         1.         0.5        0.         0.         1.
  1.         1.         1.         0.         1.         0.66666667
  0.86119403 0.34932495]
 [0.64343621 1.         0.         0.         1.         0.85915493
  1.         0.         0.         1.         1.         0.
  0.         0.         0.         0.5        0.         0.
  0.37711443 0.40031733]]
y
 [0 0 1 0 1 1 0 0 1 0]
```

SPLIT DATA INTO TRAIN AND TEST DATASET

In [39]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=30,random_state=0)
```

MODEL

In [40]:
```
# Import the LogisticRegression class from the sklearn.linear_model module.
from sklearn.linear_model import LogisticRegression
LogisticRegressionModel = LogisticRegression(penalty='l2',solver='sag',C=1.0,random_state=33)

# Fit the model to the training data.
LogisticRegressionModel.fit(X_train, y_train)

#Calculating Details
# Calculate the train and test scores.
print('LogisticRegressionModel Train Score is : ' , LogisticRegressionModel.score(X_train, y_train))
print('LogisticRegressionModel Test Score is : ' , LogisticRegressionModel.score(X_test, y_test))

# Print the classes of the model.
print('LogisticRegressionModel Classes are : ' , LogisticRegressionModel.classes_)

# Print the number of iterations used by the model.
print('LogisticRegressionModel No. of iteratios is : ' , LogisticRegressionModel.n_iter_)
print('----------------------------------------------------')

#Calculating Prediction
y_pred = LogisticRegressionModel.predict(X_test)
y_pred_prob = LogisticRegressionModel.predict_proba(X_test)

# Print the first 10 predicted values.
print('Predicted Value for LogisticRegressionModel is : ' , y_pred[:10])

# Print the first 10 prediction probabilities.
print('Prediction Probabilities Value for LogisticRegressionModel is : ' , y_pred_prob[:10])
```

```
LogisticRegressionModel Train Score is :  0.8056269637246501
LogisticRegressionModel Test Score is :  0.8333333333333334
LogisticRegressionModel Classes are :  [0 1]
LogisticRegressionModel No. of iteratios is :  [28]
----------------------------------------------------
Predicted Value for LogisticRegressionModel is :  [0 0 0 1 1 0 0 1 0 0]
Prediction Probabilities Value for LogisticRegressionModel is :  [[0.79492667 0.20507333]
 [0.84049315 0.15950685]
 [0.81140841 0.18859159]
 [0.41279408 0.58720592]
 [0.24185618 0.75814382]
 [0.6509279  0.3490721 ]
 [0.721617   0.278383  ]
 [0.35994968 0.64005032]
 [0.7658831  0.2341169 ]
 [0.60855041 0.39144959]]
```

Model Accuracy

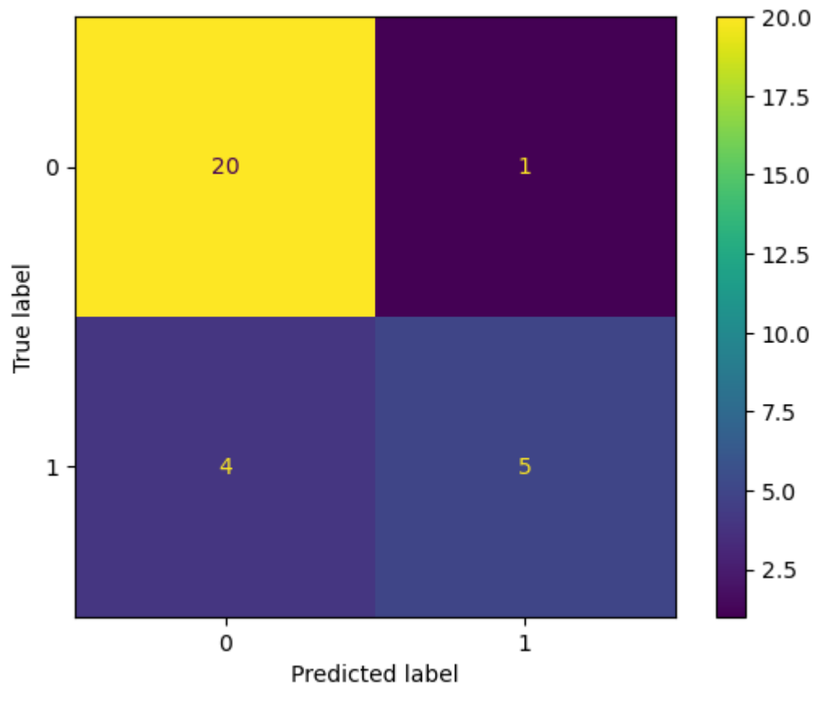
In [41]:
```
from sklearn import metrics
prediction_test = LogisticRegressionModel.predict(X_test)
# Print the prediction accuracy
print (metrics.accuracy_score(y_test, prediction_test))
```

```
0.8333333333333334
```

In [42]:
```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

# DecisionTreeClassifier  Model confusion_matrixconfusion_matrix
y_pred = LogisticRegressionModel.predict(X_test)
CM = confusion_matrix(y_test, y_pred)

CM_display = ConfusionMatrixDisplay(CM).plot()
```



In [43]:
```
from sklearn.metrics import classification_report
```

Print the classification report for the logistic regression model's predictions on the test set. The report will provide a summary of the model's performance in terms of precision, recall, F1-score, and other metrics for each class in the target variable.

In [44]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.83      0.95      0.89        21
           1       0.83      0.56      0.67         9

    accuracy                           0.83        30
   macro avg       0.83      0.75      0.78        30
weighted avg       0.83      0.83      0.82        30
```