Program:

#Import Library

```
import pandas as pd
import numpy as np
import warnings
warnings.simplefilter("ignore")
```

 # Load the dataset

```
df=pd.read_csv("/content/iris (1).csv")
df.head()
```

Output:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

```
df.info()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 5 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   sepal_length  150 non-null    float64
 1   sepal_width   150 non-null    float64
 2   petal_length  150 non-null    float64
 3   petal_width   150 non-null    float64
 4   species       150 non-null    object
dtypes: float64(4), object(1)
memory usage: 6.0+ KB
```

df.describe()

Output:

	sepal_length	sepal_width	petal_length	petal_width
count	150.000000	150.000000	150.000000	150.000000
mean	5.843333	3.054000	3.758667	1.198667
std	0.828066	0.433594	1.764420	0.763161
min	4.300000	2.000000	1.000000	0.100000
25%	5.100000	2.800000	1.600000	0.300000
50%	5.800000	3.000000	4.350000	1.300000
75%	6.400000	3.300000	5.100000	1.800000
max	7.900000	4.400000	6.900000	2.500000

 # Explore and preprocess the dataset

df.isnull().sum()

Output:

```
sepal_length    0
sepal_width     0
petal_length    0
petal_width     0
species         0
dtype: int64
```

#Split the dataset into features(x)

x=df.iloc[:,:-1]

x

Output:

	sepal_length	sepal_width	petal_length	petal_width
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

#Split the variables into target variables(y)

```
y=df.iloc[:,-1]
```

y

```
0         setosa
1         setosa
2         setosa
3         setosa
4         setosa
         ...
145    virginica
146    virginica
147    virginica
148    virginica
149    virginica
Name: species, Length: 150, dtype: object
```

 # Further split the dataset into training and testing sets

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,random_state=0)
```

```
x_train.shape
```

(112, 4)

```
x_test.shape
```

(38, 4)

```
y_train.shape
```

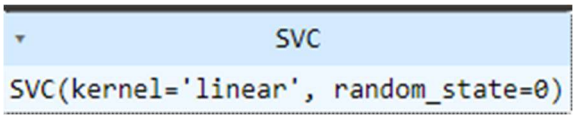
(112,)

```
y_test.shape
```

(38,)

```
from sklearn.svm import SVC
model = SVC (kernel = 'linear', random_state = 0)
model.fit(x_train,y_train)
```

Output:




```
y_pred=model.predict(x_test)
y_pred
```

Output:

```
array(['virginica', 'versicolor', 'setosa', 'virginica', 'setosa',
       'virginica', 'setosa', 'versicolor', 'versicolor', 'versicolor',
       'virginica', 'versicolor', 'versicolor', 'versicolor',
       'versicolor', 'setosa', 'versicolor', 'versicolor', 'setosa',
       'setosa', 'virginica', 'versicolor', 'setosa', 'setosa',
       'virginica', 'setosa', 'setosa', 'versicolor', 'versicolor',
       'setosa', 'virginica', 'versicolor', 'setosa', 'virginica',
       'virginica', 'versicolor', 'setosa', 'virginica'], dtype=object)
```

```
from sklearn.metrics import accuracy_score,confusion_matrix
confusion_matrix(y_test,y_pred)
```

Output:

```
array([[13,  0,  0],
       [ 0, 15,  1],
       [ 0,  0,  9]])
```

```
accuracy=accuracy_score(y_test,y_pred)*100
print("Accuracy of the model is {:.2f}".format(accuracy))
```

Accuracy of the model is 97.37

```
from sklearn.metrics import classification_report
class_report = classification_report(y_test, y_pred)
print(f"\nClassification Report:\n{class_report}")
```

Output:

```
Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        13
  versicolor       1.00      0.94      0.97        16
   virginica       0.90      1.00      0.95         9

    accuracy                           0.97        38
   macro avg       0.97      0.98      0.97        38
weighted avg       0.98      0.97      0.97        38
```

```
new_flower = [[5.1, 3.5, 1.4, 0.2]]
predicted_class = model.predict(new_flower)
predicted_class
```

Output:

```
array(['setosa'], dtype=object)
```

Program:

 # Import Libraries

```
import pandas as pd
import numpy as np
import warnings
warnings. Simplefilter ("ignore")
```

 # Load the Dataset

```
df = pd. read_csv("C:\\Users\\Documents\\loan.csv")
df. head ()
```

Output:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	LP001002	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	LP001003	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	LP001005	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	LP001006	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	LP001008	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

```
df. info ()
```

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   Loan_ID            614 non-null    object
 1   Gender             601 non-null    object
 2   Married            611 non-null    object
 3   Dependents         599 non-null    object
 4   Education          614 non-null    object
 5   Self_Employed      582 non-null    object
 6   ApplicantIncome    614 non-null    int64
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object
 12  Loan_Status        614 non-null    object
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

df. describe ()

Output:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

 # Explore and preprocess the data

df. isnull (). Sum ()

Output:

```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

 # Drop the null values

df = df. drop ('Loan_ID', axis =1)

df. head ()

Output:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	Male	No	0	Graduate	No	5849	0.0	NaN	360.0	1.0
1	Male	Yes	1	Graduate	No	4583	1508.0	128.0	360.0	1.0
2	Male	Yes	0	Graduate	Yes	3000	0.0	66.0	360.0	1.0
3	Male	Yes	0	Not Graduate	No	2583	2358.0	120.0	360.0	1.0
4	Male	No	0	Graduate	No	6000	0.0	141.0	360.0	1.0

Encoding the categorical values

```
from sklearn. preprocessing import LabelEncoder

label_encoder = LabelEncoder ()

categorical_columns = ['Gender', 'Married', 'Education', 'Self_Employed', 'Property_Area', 'Loan_Status']

df[categorical_columns]=df[categorical_columns].apply(label_encoder.fit_transform)

df[categorical_columns]
```

Output:

	Gender	Married	Education	Self_Employed	Property_Area	Loan_Status
0	1	0	0	0	2	1
1	1	1	0	0	0	0
2	1	1	0	1	2	1
3	1	1	1	0	2	1
4	1	0	0	0	2	1
...	...	...	...	...	...	...
609	0	0	0	0	0	1
610	1	1	0	0	0	1
611	1	1	0	0	2	1
612	1	1	0	0	2	1
613	0	0	0	1	1	0

614 rows × 6 columns

Replace the null value

```
from sklearn. impute import SimpleImputer

numeric_columns = ['Gender', 'Married','Dependents', 'Self_Employed', 'LoanAmount', 'Loan_Amount_Term',

'Credit_History']

df[numeric_columns] = SimpleImputer(strategy='most_frequent'). fit_transform(df[numeric_columns])
```

df[numeric_columns]

Output:

	Gender	Married	Dependents	Self_Employed	LoanAmount	Loan_Amount_Term	Credit_History
0	1	0	0	0	120.0	360.0	1.0
1	1	1	1	0	128.0	360.0	1.0
2	1	1	0	1	66.0	360.0	1.0
3	1	1	0	0	120.0	360.0	1.0
4	1	0	0	0	141.0	360.0	1.0
...	...	...	...	...	...	...	...
609	0	0	0	0	71.0	360.0	1.0
610	1	1	3+	0	40.0	180.0	1.0
611	1	1	1	0	253.0	360.0	1.0
612	1	1	2	0	187.0	360.0	1.0
613	0	0	0	1	133.0	360.0	0.0

614 rows × 7 columns

Handle special case in ‘Dependents’ column

df['Dependents'] = df['Dependents']. replace ('3+', 3). astype(float)

df['Dependents']

Output:

```
0      0.0
1      1.0
2      0.0
3      0.0
4      0.0
      ...
609    0.0
610    3.0
611    1.0
612    2.0
613    0.0
Name: Dependents, Length: 614, dtype: float64
```

Split the data into features (x)

x = df. iloc [: , :-1]

x

Output:

	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
0	1	0	0.0	0	0	5849	0.0	120.0	360.0	1.0
1	1	1	1.0	0	0	4583	1508.0	128.0	360.0	1.0
2	1	1	0.0	0	1	3000	0.0	66.0	360.0	1.0
3	1	1	0.0	1	0	2583	2358.0	120.0	360.0	1.0
4	1	0	0.0	0	0	6000	0.0	141.0	360.0	1.0
...	...	...	...	...	...	...	...	...	...	...
609	0	0	0.0	0	0	2900	0.0	71.0	360.0	1.0
610	1	1	3.0	0	0	4106	0.0	40.0	180.0	1.0
611	1	1	1.0	0	0	8072	240.0	253.0	360.0	1.0
612	1	1	2.0	0	0	7583	0.0	187.0	360.0	1.0
613	0	0	0.0	0	1	4583	0.0	133.0	360.0	0.0

614 rows × 11 columns

 # Split the data into target variable (y)

y = df. iloc [: , -1]

y

Output:

```
0      1
1      0
2      1
3      1
4      1
      ..
609    1
610    1
611    1
612    1
613    0
Name: Loan_Status, Length: 614, dtype: int64
```

 # Further split the dataset into training and testing sets

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=0)

 # Feature scaling using StandardScaler

from sklearn. preprocessing import StandardScaler

scaler = StandardScaler ()

x_train = scaler.fit_transform(x_train)

x_test = scaler. transform(x_test)

x_train

Output:

```
array([[ 0.36547961,  0.6622422 ,  0.20631248, ...,  0.27778225,
         0.41649656,  1.20186498],
       [ 0.36547961, -1.42427431, -0.77207659, ...,  0.27778225,
         0.41649656, -1.31684978],
       [ 0.36547961, -1.42427431,  1.18470154, ...,  0.27778225,
         0.41649656, -1.31684978],
       ...,
       [ 0.36547961,  0.6622422 ,  2.1630906 , ...,  0.27778225,
         0.41649656, -0.0574924 ],
       [ 0.36547961,  0.6622422 , -0.77207659, ...,  0.27778225,
         0.41649656,  1.20186498],
       [-2.00241646,  0.6622422 , -0.77207659, ...,  0.27778225,
         0.41649656, -0.0574924 ]])
```

x_test

Output:

```
array([[ 0.36547961, -1.42427431, -0.77207659, ...,  0.27778225,
         0.41649656, -0.0574924 ],
       [-2.00241646, -1.42427431, -0.77207659, ...,  0.27778225,
         0.41649656, -0.0574924 ],
       [ 0.36547961,  0.6622422 , -0.77207659, ...,  0.27778225,
         0.41649656,  1.20186498],
       ...,
       [ 0.36547961,  0.6622422 , -0.77207659, ...,  0.27778225,
         0.41649656,  1.20186498],
       [ 0.36547961, -1.42427431, -0.77207659, ...,  0.27778225,
         0.41649656, -0.0574924 ],
       [ 0.36547961,  0.6622422 ,  0.20631248, ...,  0.27778225,
        -2.40098019, -0.0574924 ]])
```

 # Intialize the KNN model

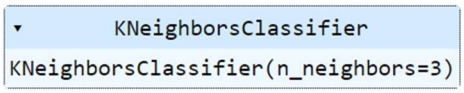
from sklearn. neighbors import KNeighborsClassifier

model = KNeighborsClassifier(n_neighbors=3)

 # Train the KNN model

model.fit (x_train, y_train)

Output:



 # Make predictions

y_pred=model. predict(x_test)

y_pred

Output:

```
array([1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1,
       1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 0, 0, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0])
```

 # Evaluate the model

from sklearn. metrics import accuracy_score, classification_report

accuracy = accuracy_score (y_test, y_pred) * 100

print ("Accuracy of the model is {:.2f}". format(accuracy))

Output:

```
Accuracy of the model is 78.57
```

from sklearn. metrics import classification_report

class_report = classification_report (y_test, y_pred)

print (f"\nClassification Report:\n{class_report}")

Output:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.64      0.53      0.58        43
           1       0.83      0.88      0.86       111

    accuracy                           0.79       154
   macro avg       0.73      0.71      0.72       154
weighted avg       0.78      0.79      0.78       154
```

 # Predict for new data

new_data = [1, 1, 2, 1, 1, 4106.0, 240.0, 253.0, 360.0, 1, 2]

```
predictions = model. predict([new_data])
print(predictions)
```

Output:

```
[1]
```

```
# Assuming a label encoder for decoding categorical predictions
decoded_predictions = label_encoder. inverse_transform(predictions)
print(decoded_predictions)
```

Output:

```
['Y']
```

PROGRAM:

 # Import Libraries

import pandas as pd

import numpy as np

import warnings

warnings. Simplefilter ("ignore")

 # Load the Dataset

df = pd. read_csv("C:\\Users\\Documents\\Twitter.csv")

df. head ()

Output:

	status_id	text	created_at	favorite_count	retweet_count	location	followers_count	friends_count	statuses_count	category
0	1046207313588230000	Entitled, obnoxious, defensive, lying weasel. ...	2018-09-30T01:17:15Z	5	1	McAllen, TX	2253	2303	23856	0
1	1046207328113080000	Thank you and for what you did for the women...	2018-09-30T01:17:19Z	5	2	Tampa, FL	2559	4989	19889	0
2	1046207329589490000	Knitting (s) & getting ready for January 1...	2018-09-30T01:17:19Z	0	0	St Cloud, MN	16	300	9	0
3	1046207341283160000	Yep just like triffeling women weaponized thei...	2018-09-30T01:17:22Z	1	0	flyover country	3573	3732	38361	1

df.info ()

Output:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 807174 entries, 0 to 807173
Data columns (total 10 columns):
 #   Column           Non-Null Count   Dtype
---  ------           --------------   -----
 0   status_id        807174 non-null  int64
 1   text             803638 non-null  object
 2   created_at       807174 non-null  object
 3   favorite_count   807174 non-null  int64
 4   retweet_count    807174 non-null  int64
 5   location         616394 non-null  object
 6   followers_count  807174 non-null  int64
 7   friends_count    807174 non-null  int64
 8   statuses_count   807174 non-null  int64
 9   category         807174 non-null  int64
dtypes: int64(7), object(3)
memory usage: 61.6+ MB
```

df. describe ()

Output:

	status_id	favorite_count	retweet_count	followers_count	friends_count	statuses_count	category
count	8.071740e+05	807174.000000	807174.000000	8.071740e+05	807174.000000	8.071740e+05	807174.000000
mean	1.061411e+18	6.466671	2.557508	4.663755e+04	6306.597207	4.793559e+04	0.118108
std	1.428176e+16	159.865656	50.724140	6.111813e+05	40549.763583	1.646474e+05	0.322737
min	1.046207e+18	0.000000	0.000000	0.000000e+00	0.000000	1.000000e+00	0.000000
25%	1.050184e+18	0.000000	0.000000	1.070000e+02	160.000000	1.663250e+03	0.000000
50%	1.054643e+18	0.000000	0.000000	5.390000e+02	528.000000	8.007000e+03	0.000000
75%	1.071177e+18	1.000000	0.000000	2.844000e+03	1756.000000	3.317500e+04	0.000000
max	1.097647e+18	70385.000000	17484.000000	5.457643e+07	899383.000000	9.565126e+06	1.000000

df.isnull().sum()

Output:

```
status_id              0
text                3536
created_at             0
favorite_count         0
retweet_count          0
location          190780
followers_count        0
friends_count          0
statuses_count         0
category               0
dtype: int64
```

 # Replace the null value

df['text']. fillna ('', inplace=True)

df['text']

Output:

```
0         Entitled, obnoxious, defensive, lying weasel. ...
1         Thank you  and  for what you did for the women...
2         Knitting (s) &amp;amp; getting ready for January 1...
3         Yep just like triffeling women weaponized thei...
4         No, the President wants to end  movement posin...
                                ...
807169    Letâ□□s not forget that this â□□iconic kissâ□□...
807170    DEFINITELY....the only one any of us should su...
807171    Did the  movement count the dollars of Erin An...
807172    This is one of my all time fav songs &amp;amp; vid...
807173     I watched your news on the death of the sailo...
Name: text, Length: 807174, dtype: object
```

df['location']. fillna ('', inplace=True)

df['location']

Output:

```
0                 McAllen, TX
1                   Tampa, FL
2                 St Cloud, MN
3              flyover country
4                        World
                   ...
807169           South Florida
807170
807171                  Philly
807172    Berlin, Deutschland
807173     Massachusetts, USA
Name: location, Length: 807174, dtype: object
```

 # Drop the null values

df = df. drop (['status_id', 'created_at'], axis =1)

df. head ()

Output:

	text	favorite_count	retweet_count	location	followers_count	friends_count	statuses_count	category
0	Entitled, obnoxious, defensive, lying weasel. ...	5	1	McAllen, TX	2253	2303	23856	0
1	Thank you and for what you did for the women...	5	2	Tampa, FL	2559	4989	19889	0
2	Knitting (s) & getting ready for January 1...	0	0	St Cloud, MN	16	300	9	0
3	Yep just like triffeling women weaponized thei...	1	0	flyover country	3573	3732	38361	1
4	No, the President wants to end movement posin...	0	0	World	294	312	7635	0

 # Split the data into features (x)

x = df. iloc [: , :-1]

x

Output:

	text	favorite_count	retweet_count	location	followers_count	friends_count	statuses_count
0	Entitled, obnoxious, defensive, lying weasel. ...	5	1	McAllen, TX	2253	2303	23856
1	Thank you and for what you did for the women...	5	2	Tampa, FL	2559	4989	19889
2	Knitting (s) & getting ready for January 1...	0	0	St Cloud, MN	16	300	9
3	Yep just like triffeling women weaponized thei...	1	0	flyover country	3573	3732	38361
4	No, the President wants to end movement posin...	0	0	World	294	312	7635
...	...	...	...	...	...	...	...
807169	Letâ□□s not forget that this â□□iconic kissâ□□...	2	0	South Florida	206	412	1247
807170	DEFINITELY....the only one any of us should su...	3	0		63	6	137
807171	Did the movement count the dollars of Erin An...	0	0	Philly	2721	3509	66966
807172	This is one of my all time fav songs & vid...	1	1	Berlin, Deutschland	2683	1011	15455
807173	I watched your news on the death of the sailo...	1	0	Massachusetts, USA	237	741	789

 # Split the data into target variable (y)

y = df. iloc [: , -1]

Output:

```
0         0
1         0
2         0
3         1
4         0
         ..
807169    0
807170    0
807171    0
807172    1
807173    0
Name: category, Length: 807174, dtype: int64
```

 # Further split the dataset into training and testing sets

from sklearn. model_selection import train_test_split

x_train, x_test, y_train, y_test=train_test_split (x, y, random_state=0)

 # Use CountVectorizer for text feature on training data

from sklearn. feature_extraction.text import CountVectorizer

vectorizer = CountVectorizer ()

x_train_text = vectorizer.fit_transform(x_train['text'])

x_train_text

Output:

```
<605380x130326 sparse matrix of type '<class 'numpy.int64'>'
        with 11031463 stored elements in Compressed Sparse Row format>
```

 # Apply the same CountVectorizer to transform the text feature in the test data

x_test_text = vectorizer. transform(x_test['text'])

x_test_text

Output:

```
<201794x130326 sparse matrix of type '<class 'numpy.int64'>'
        with 3651979 stored elements in Compressed Sparse Row format>
```

 # Intialize the Naïve Bayes model

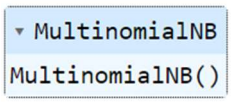
from sklearn. naive_bayes import MultinomialNB

model = MultinomialNB ()

 # Train the Naïve Bayes model

model.fit (x_train, y_train)

Output:



 # Make predictions

y_pred = model. predict(x_test_text)

y_pred

Output:

array ([0, 0, 0, ..., 0, 0, 0], dtype=int64)

 # Evaluate the model

from sklearn. metrics import accuracy_score, classification_report

accuracy = accuracy_score (y_test, y_pred) * 100

print ("Accuracy of the model is {:.2f}". format(accuracy))

Output:

Accuracy score of the model is93.11

```
from sklearn. metrics import classification_report
class_report = classification_report (y_test, y_pred)
print (f"\nClassification Report:\n{class_report}")
```

Output:

```
Classification Report:
              precision    recall  f1-score   support

           0       0.96      0.96      0.96    178149
           1       0.71      0.71      0.71     23645

    accuracy                           0.93    201794
   macro avg       0.83      0.83      0.83    201794
weighted avg       0.93      0.93      0.93    201794
```