```python
import numpy as np
import pandas as pd
```

Cmd 3

```python
from pyspark.sql import SparkSession

# Create a SparkSession
spark = SparkSession.builder \
    .appName("BankFraud") \
    .getOrCreate()
```

Command took 0.05 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 1:41:38 PM on

```python
df1 = spark.read.csv('dbfs:/mnt/output1/train_transaction.csv', header=True, inferSchema=True)
```

▸ (2) Spark Jobs

▸ df1: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 392 more fields]

Command took 16.62 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 1:41:40 PM on Sayali Chaudhari's Cluster

Cmd 5

```python
df2 = spark.read.csv('dbfs:/mnt/output1/train_identity.csv', header=True, inferSchema=True)
```

▸ (2) Spark Jobs

```python
num_rows = df1.count()

# Get the number of columns
num_columns = len(df1.columns)

print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

▸ (2) Spark Jobs

```
Number of Rows: 590540
Number of Columns: 394
```

```python
num_rows = df2.count()

# Get the number of columns
num_columns = len(df2.columns)

print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

(2) Spark Jobs

```
Number of Rows: 144233
Number of Columns: 41
```

```python
df3 = df1.join(df2, on="TransactionID", how="left")
```

df3: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 432 more fields]

Command took 0.19 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 1:42:07 PM on Sayal

Cmd 9

```python
num_rows = df3.count()

# Get the number of columns
num_columns = len(df3.columns)

print("Number of Rows:", num_rows)
print("Number of Columns:", num_columns)
```

(3) Spark Jobs

```
Number of Rows: 590540
Number of Columns: 434
```

```python
from pyspark.sql.functions import col, sum
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 1:42:15 PM on Sayali Chaudhari's Cluster

Cmd 11

```python
total_rows = df3.count()
null_percentages = [(col_name, df3.filter(col(col_name).isNull()).count() / total_rows) for col_name in df3.columns]
```

(94) Spark Jobs

Command took 43.01 minutes -- by chaudharisayali12@gmail.com at 8/26/2023, 1:42:19 PM on Sayali Chaudhari's Cluster

```
columns_to_drop = [col_name for col_name, null_percentage in null_percentages if null_percentage > 0.4]
```

Command took 0.11 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:26:57 PM on Sayali Chaudhari's Cluster

Cmd 13

```
df = df3.drop(*columns_to_drop)
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.20 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:03 PM on Sayali Chaudhari's Cluster

Cmd 14

```
len(df.columns)
```

Out[10]: 202

Command took 0.06 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:06 PM on Sayali Chaudhari's Cluster

Cmd 15

```
columns_to_replace = ["card2", "addr1", "addr2","D4"]
```

```
from pyspark.sql.functions import col, mean
```

Command took 0.04 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:10 PM on Sayali Chaudhari's Cluster

Cmd 17

```
means = df.select(*[mean(col(c)).alias(c) for c in columns_to_replace]).collect()[0]
```

(3) Spark Jobs

Command took 6.06 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:14 PM on Sayali Chaudhari's Cluster

Cmd 18

```
mean_values = {col_name: mean_val for col_name, mean_val in zip(columns_to_replace, means)}
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:25 PM on Sayali Chaudhari's Cluster

Cmd 19

```
for col_name in columns_to_replace:
    df = df.fillna(mean_values[col_name], subset=[col_name])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.28 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:26 PM on Sayali Chaudhari's Cluster

```
for i in columns_to_replace:
    null_count = df.select(sum(col(i).isNull().cast("int")).alias("null_count")).collect()[0]["null_count"]

# Display the null count
    print(f"Null count for {i}: {null_count}")
```

(12) Spark Jobs

```
Null count for card2: 0
Null count for addr1: 0
Null count for addr2: 0
Null count for D4: 0
```

```
from pyspark.sql.functions import median
```

Command took 0.03 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:32 PM on Sayali Chaudhari's Cluster

Cmd 22

```
columns_to_replace = ["card3", "card5"]
```

Command took 0.12 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:33 PM on Sayali Chaudhari's Cluster

Cmd 23

```
medians = df.select(*[median(col(c)).alias(c) for c in columns_to_replace]).collect()[0]
```

(3) Spark Jobs

Command took 6.04 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:36 PM on Sayali Chaudhari's Cluster

Cmd 24

```
median_values = {col_name: median_val for col_name, median_val in zip(columns_to_replace, medians)}
```

Command took 0.10 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:37 PM on Sayali Chaudhari's Cluster

```
for col_name in columns_to_replace:
    df = df.fillna(median_values[col_name], subset=[col_name])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.19 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:48 PM on Sayali Chaudhari's Cluster

Cmd 26

```
from pyspark.sql.functions import mode
```

Command took 0.05 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:27:57 PM on Sayali Chaudhari's Cluster

Cmd 27

```
column_to_replace = 'card4'
```

Command took 0.08 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:01 PM on Sayali Chaudhari's Cluster

Cmd 28

```
mode_value = df.select(mode(col(column_to_replace))).collect()[0][0]
```

(3) Spark Jobs

Command took 6.21 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:01 PM on Sayali Chaudhari's Cluster

```
df = df.fillna(mode_value, subset=[column_to_replace])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.15 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:11 PM on Sayali Chaudhari's Cluster

Cmd 30

```
column_to_replace = 'card6'
```

Command took 0.06 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:12 PM on Sayali Chaudhari's Cluster

Cmd 31

```
mode_value = df.select(mode(col(column_to_replace))).collect()[0][0]
```

(3) Spark Jobs

Command took 5.97 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:17 PM on Sayali Chaudhari's Cluster

Cmd 32

```
df = df.fillna(mode_value, subset=[column_to_replace])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.19 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:18 PM on Sayali Chaudhari's Cluster

```
column_to_replace = 'P_emaildomain'
```

Command took 0.03 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:27 PM on Sayali Chaudhari's Cluster

Cmd 34

```
mode_value = df.select(mode(col(column_to_replace))).collect()[0][0]
```

(3) Spark Jobs

Command took 6.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:29 PM on Sayali Chaudhari's Cluster

Cmd 35

```
df = df.fillna(mode_value, subset=[column_to_replace])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.12 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:40 PM on Sayali Chaudhari's Cluster

Cmd 36

```
column_to_replace = 'M6'
```

Command took 0.13 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:41 PM on Sayali Chaudhari's Cluster

Cmd 37

```
mode_value = df.select(mode(col(column_to_replace))).collect()[0][0]
```

```
df = df.fillna(mode_value, subset=[column_to_replace])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.13 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:54 PM on Sayali Chaudhari's Cluster

Cmd 39

```
columns_to_replace = ['D1','D10','D15']
```

Command took 0.04 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:58 PM on Sayali Chaudhari's Cluster

Cmd 40

```
medians = df.select(*[median(col(c)).alias(c) for c in columns_to_replace]).collect()[0]
```

(3) Spark Jobs

Command took 6.27 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:28:58 PM on Sayali Chaudhari's Cluster

Cmd 41

```
median_values = {col_name: median_val for col_name, median_val in zip(columns_to_replace, medians)}
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:29:10 PM on Sayali Chaudhari's Cluster

```
for col_name in columns_to_replace:
    df = df.fillna(median_values[col_name], subset=[col_name])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 0.16 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:29:11 PM on Sayali Chaudhari's Cluster

Cmd 43

```
columns_to_replace = [col_name for col_name in df.columns if col_name.startswith("V")]
```

Command took 0.12 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:29:18 PM on Sayali Chaudhari's Cluster

Cmd 44

```
medians = df.select(*[median(col(c)).alias(c) for c in columns_to_replace]).collect()[0]
```

(3) Spark Jobs

Command took 19.18 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:29:18 PM on Sayali Chaudhari's Cluster

Cmd 45

```
median_values = {col_name: median_val for col_name, median_val in zip(columns_to_replace, medians)}
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:30:45 PM on Sayali Chaudhari's Cluster

```
for col_name in columns_to_replace:
    df = df.fillna(median_values[col_name], subset=[col_name])
```

df: pyspark.sql.dataframe.DataFrame = [TransactionID: integer, isFraud: integer ... 200 more fields]

Command took 9.01 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:30:47 PM on Sayali Chaudhari's Cluster

Cmd 47

```
df = df.drop('TransactionID')
```

df: pyspark.sql.dataframe.DataFrame = [isFraud: integer, TransactionDT: integer ... 199 more fields]

Command took 0.16 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:31:00 PM on Sayali Chaudhari's Cluster

Cmd 48

```
new_df = df.toPandas()
```

(2) Spark Jobs

```
new_df.select_dtypes(include=['object'])
```

	ProductCD	card4	card6	P_emaildomain	M6
0	H	mastercard	credit	gmail.com	F
1	H	visa	debit	anonymous.com	F
2	C	mastercard	credit	gmail.com	F
3	C	mastercard	debit	hotmail.com	F
4	H	visa	debit	aol.com	F
...	...	...	...	...	...
590535	W	visa	debit	gmail.com	F
590536	W	mastercard	debit	gmail.com	T
590537	W	mastercard	debit	gmail.com	T
590538	W	mastercard	debit	aol.com	T
590539	W	mastercard	credit	gmail.com	T

590540 rows × 5 columns

```python
from sklearn.preprocessing import LabelEncoder
```

1

Command took 0.82 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:31:46 PM on Sayali Chaudhari's Cluster

Cmd 54

```python
label_encoder = LabelEncoder()

# Specify columns to label encode
columns_to_encode = ['P_emaildomain']

# Apply label encoding to the specified columns
for column in columns_to_encode:
    new_df[column] = label_encoder.fit_transform(new_df[column])
```

```python
# Specify columns to one-hot encode
columns_to_encode = ['ProductCD', 'card4', 'card6','M6']

# Apply one-hot encoding to the specified columns
new_df = pd.get_dummies(new_df, columns=columns_to_encode, prefix=columns_to_encode)
```

Command took 1.78 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:31:57 PM on Sayali Chaudhari's Cluster

d 56

```python
new_cpy = new_df
```

Command took 0.11 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:04 PM on Sayali Chaudhari's Cluster

d 57

```python
new_cpy.shape
```

Out[51]: (590540, 212)

Command took 0.08 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:06 PM on Sayali Chaudhari's Cluster

d 58

```python
fraud_group = new_df[new_df['isFraud'] == 1]
non_fraud_group = new_df[new_df['isFraud'] == 0]
```

```python
from scipy.stats import f_oneway

selected_features = []

for feature in new_df.columns:
    if feature != 'isFraud':
        f_statistic, p_value = f_oneway(fraud_group[feature], non_fraud_group[feature])

        if p_value < 0.05:  # You can adjust the significance level
            selected_features.append(feature)

print("Selected Features:", selected_features)
```

```
len(selected_features)
```

Out[54]: 189

Command took 0.10 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:27 PM on Sayali Chaudhari

Cmd 61

```
new_df = new_df[selected_features]
```

Command took 0.93 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:30 PM on Sayali Chaudhari

Cmd 62

```
new_df['isFraud'] = new_cpy['isFraud']
```

```
new_df.shape
```

Out[57]: (590540, 190)

Command took 0.11 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:40 PM on Sayali

Cmd 64

```
pip install imblearn
```

```
from imblearn.over_sampling import SMOTE
```

Command took 0.14 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:47 PM on Sayali Chaudhari's Cluster

Cmd 67

```
smote = SMOTE(sampling_strategy='auto', random_state=42)
```

Command took 0.12 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:32:52 PM on Sayali Chaudhari's Cluster

Cmd 68

```
x = new_df.drop('isFraud', axis=1)
y = new_df['isFraud']
```

Command took 1.02 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:33:09 PM on Sayali Chaudhari's Cluster

Cmd 69

```
x_sm, y_sm = smote.fit_resample(x, y)
```

```
y.value_counts()
```

```
Out[62]: 0    569877
1     20663
Name: isFraud, dtype: int64
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com

Cmd 71

```
y_sm.value_counts()
```

```
Out[63]: 0    569877
1    569877
Name: isFraud, dtype: int64
```

```
from sklearn.preprocessing import StandardScaler
```

Command took 0.05 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:34:26 PM on Sayali Chaudhari's Clus

Cmd 73

```
scaler = StandardScaler()
```

Command took 0.03 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:34:35 PM on Sayali Chaudhari's Clus

Cmd 74

```
x_sm = scaler.fit_transform(x_sm)
```

Command took 8.00 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:34:44 PM on Sayali Chaudhari's Clus

```
from sklearn.model_selection import train_test_split
```

Command took 0.11 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:35:08 PM on Sayali Chaudhari's Cluster

Cmd 76

```
x_train, x_test, y_train, y_test = train_test_split(x_sm, y_sm, test_size=0.25)
```

Command took 3.76 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:35:32 PM on Sayali Chaudhari's Cluster

Logistic Regression

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:35:54 PM on Sayali Chaudhari's Cluster

Cmd 78

```
model = LogisticRegression()

model.fit(x_train, y_train)
```

```
y_pred = model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Print classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8944932073180576
Confusion Matrix:
 [[135995   6878]
 [ 23185 118881]]
Classification Report:
               precision    recall  f1-score   support

           0       0.85      0.95      0.90    142873
           1       0.95      0.84      0.89    142066

    accuracy                           0.89    284939
   macro avg       0.90      0.89      0.89    284939
weighted avg       0.90      0.89      0.89    284939
```

Decision Tree

```
from sklearn.tree import DecisionTreeClassifier
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:37:15 PM on Sayali Chaudhari's Cluster

nd 81

```
model = DecisionTreeClassifier(random_state=42)

# Train the model on the training data
model.fit(x_train, y_train)
```

Out[73]: DecisionTreeClassifier(random_state=42)

```
y_pred = model.predict(x_test)

# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Print classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9799536041047382
Confusion Matrix:
 [[139780   3093]
 [  2619 139447]]
Classification Report:
               precision    recall  f1-score   support

           0       0.98      0.98      0.98    142873
           1       0.98      0.98      0.98    142066

    accuracy                           0.98    284939
   macro avg       0.98      0.98      0.98    284939
weighted avg       0.98      0.98      0.98    284939
```

Command took 0.75 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:40:06 PM on Sayali Chaudhari's Cluster

XGB

```
import xgboost as xgb
```

Command took 0.13 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:40:19 PM on Sayali Chaudhar

Cmd 84

```
xgb_classifier = xgb.XGBClassifier(
    objective='binary:logistic',  # For binary classification
    learning_rate=0.1,
    max_depth=3,
    n_estimators=100
)
```

```
xgb_classifier.fit(x_train, y_train)
```

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

Command took 0.28 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2

```
# Calculate accuracy
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)

# Print confusion matrix
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))

# Print classification report
print("Classification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.9673298495467451
Confusion Matrix:
 [[140036   2837]
 [  6472 135594]]
Classification Report:
               precision    recall  f1-score   support

           0       0.96      0.98      0.97    142873
           1       0.98      0.95      0.97    142066

    accuracy                           0.97    284939
   macro avg       0.97      0.97      0.97    284939
weighted avg       0.97      0.97      0.97    284939
```

Random forest

```
from sklearn.ensemble import RandomForestClassifier
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:51:54 PM on Sayali Chaudhari's Cluster

Cmd 89

```
rf = RandomForestClassifier(n_estimators=10,criterion='entropy',random_state=0)
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:52:10 PM on Sayali Chaudhari's Cluster

Cmd 90

```
mod1 = rf.fit(x_train,y_train)
```

Command took 41.81 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:52:23 PM on Sayali Chaudhari's Cluster

Cmd 91

```
y_pred = mod1.predict(x_test)
```

```
print("confusion_matrix")
print(confusion_matrix(y_pred,y_test))
print()
print("accuracy_score")
print(accuracy_score(y_pred,y_test))
print()
print("classification_report")
print(classification_report(y_pred,y_test))
print()
```

```
confusion_matrix
[[142542   2833]
 [   331 139233]]

accuracy_score
0.9888958689403697

classification_report
              precision    recall  f1-score   support

           0       1.00      0.98      0.99    145375
           1       0.98      1.00      0.99    139564

    accuracy                           0.99    284939
   macro avg       0.99      0.99      0.99    284939
weighted avg       0.99      0.99      0.99    284939
```

KNN

```
from sklearn.neighbors import KNeighborsClassifier
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:54:33 PM on Sayali Chaudhari's Cluster

Cmd 94

```
knnc = KNeighborsClassifier(n_neighbors=5)
```

Command took 0.09 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:54:51 PM on Sayali Chaudhari's Cluster

Cmd 95

```
mod1 = knnc.fit(x_train,y_train)
```

Command took 0.21 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 2:55:36 PM on Sayali Chaudhari's Cluster

Cmd 96

```
y_pred = mod1.predict(x_test)
```

```
print("confusion_matrix")
print(confusion_matrix(y_pred,y_test))
print()
print("accuracy_score")
print(accuracy_score(y_pred,y_test))
print()
print("classification_report")
print(classification_report(y_pred,y_test))
print()
```

```
confusion_matrix
[[138462   1812]
 [  4411 140254]]

accuracy_score
0.9781602378052846

classification_report
              precision    recall  f1-score   support

           0       0.97      0.99      0.98    140274
           1       0.99      0.97      0.98    144665

    accuracy                           0.98    284939
   macro avg       0.98      0.98      0.98    284939
weighted avg       0.98      0.98      0.98    284939
```

Naïve Bayes

```
from sklearn.naive_bayes import GaussianNB
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 4:39:58 PM on Sayali Chaudhari's Cluster

Cmd 99

```
model = GaussianNB()
```

Command took 0.07 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 4:40:15 PM on Sayali Chaudhari's Cluster

Cmd 100

```
mod1 = model.fit(x_train,y_train)
```

Command took 2.67 seconds -- by chaudharisayali12@gmail.com at 8/26/2023, 4:40:33 PM on Sayali Chaudhari's Cluster

Cmd 101

```
y_pred = mod1.predict(x_test)
```

```
print("confusion_matrix")
print(confusion_matrix(y_pred,y_test))
print()
print("accuracy_score")
print(accuracy_score(y_pred,y_test))
print()
print("classification_report")
print(classification_report(y_pred,y_test))
print()
```

```
confusion_matrix
[[ 60518  12538]
 [ 82355 129528]]

accuracy_score
0.6669708253345453

classification_report
              precision    recall  f1-score   support

           0       0.42      0.83      0.56     73056
           1       0.91      0.61      0.73    211883

    accuracy                           0.67    284939
   macro avg       0.67      0.72      0.65    284939
weighted avg       0.79      0.67      0.69    284939
```