In [ ]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [ ]:
```
df = pd.read_csv("/content/Exam_Score_Prediction.csv")
df.head()
```

Out[ ]:

	student_id	age	gender	course	study_hours	class_attendance	internet_ac
0	1	17	male	diploma	2.78	92.9	
1	2	23	other	bca	3.37	64.8	
2	3	22	male	b.sc	7.88	76.8	
3	4	20	other	diploma	0.67	48.4	
4	5	20	female	diploma	0.89	71.6	

In [ ]:
```
df.shape
```

Out[ ]:
```
(20000, 13)
```

In [ ]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 20000 entries, 0 to 19999
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   student_id        20000 non-null  int64  
 1   age               20000 non-null  int64  
 2   gender            20000 non-null  object 
 3   course            20000 non-null  object 
 4   study_hours       20000 non-null  float64
 5   class_attendance  20000 non-null  float64
 6   internet_access   20000 non-null  object 
 7   sleep_hours       20000 non-null  float64
 8   sleep_quality     20000 non-null  object 
 9   study_method      20000 non-null  object 
 10  facility_rating   20000 non-null  object 
 11  exam_difficulty   20000 non-null  object 
 12  exam_score        20000 non-null  float64
dtypes: float64(4), int64(2), object(7)
memory usage: 2.0+ MB
```

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

Out[ ]:

	student_id	age	study_hours	class_attendance	sleep_hours
count	20000.000000	20000.000000	20000.000000	20000.000000	20000.00000
mean	10000.504600	20.473300	4.007604	70.017365	7.00856
std	5773.654959	2.284458	2.308313	17.282262	1.73209
min	1.000000	17.000000	0.080000	40.600000	4.10000
25%	5000.750000	18.000000	2.000000	55.100000	5.50000
50%	10000.500000	20.000000	4.040000	69.900000	7.00000
75%	15000.250000	22.000000	6.000000	85.000000	8.50000
max	20001.000000	24.000000	7.910000	99.400000	9.90000

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

Out[ ]:

	0
student_id	0
age	0
gender	0
course	0
study_hours	0
class_attendance	0
internet_access	0
sleep_hours	0
sleep_quality	0
study_method	0
facility_rating	0
exam_difficulty	0
exam_score	0

dtype: int64

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

Out[ ]:
```
Index(['student_id', 'age', 'gender', 'course', 'study_hours',
       'class_attendance', 'internet_access', 'sleep_hours', 'sleep_quality',
       'study_method', 'facility_rating', 'exam_difficulty', 'exam_score'],
      dtype='object')
```

This dataset has no null values so our data is cleaned

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

Out[ ]:

	0
student_id	int64
age	int64
gender	object
course	object
study_hours	float64
class_attendance	float64
internet_access	object
sleep_hours	float64
sleep_quality	object
study_method	object
facility_rating	object
exam_difficulty	object
exam_score	float64

dtype: object

In [ ]:
```
df = df.drop("student_id", axis=1)
```

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

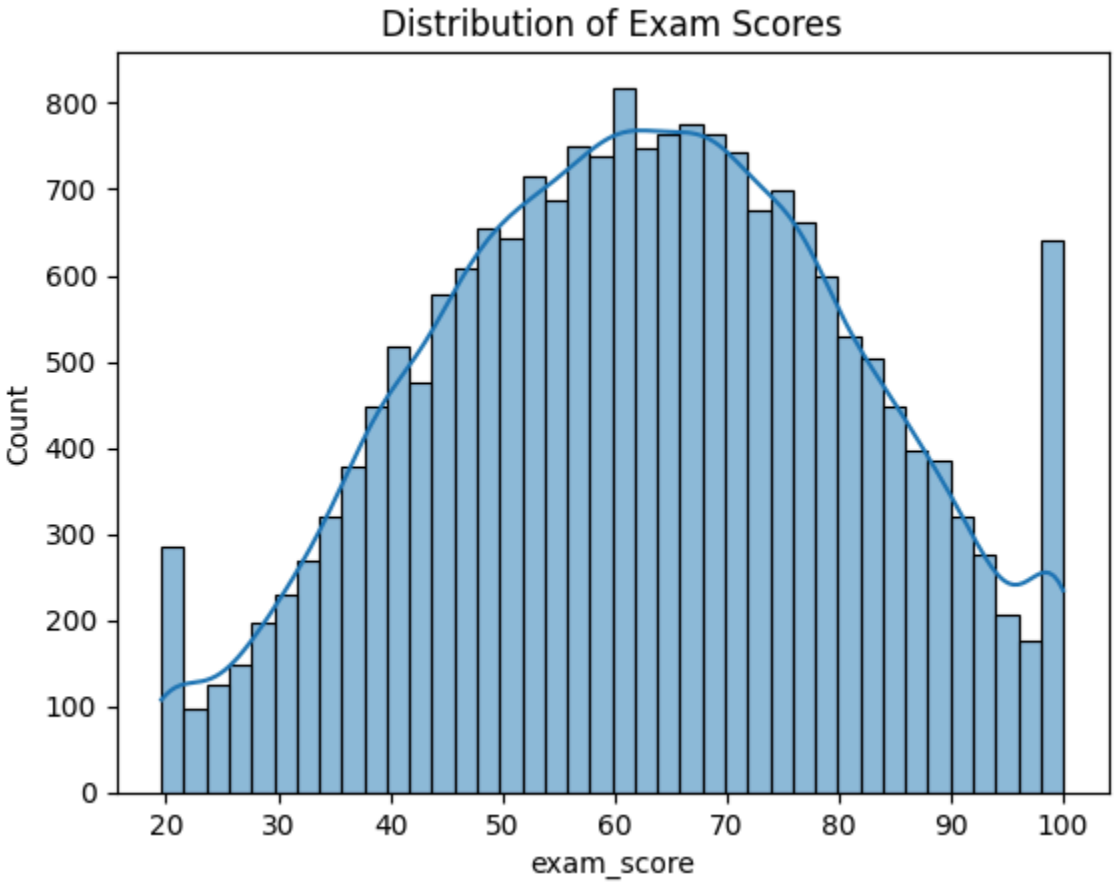
Out[ ]:
```
Index(['age', 'gender', 'course', 'study_hours', 'class_attendance',
       'internet_access', 'sleep_hours', 'sleep_quality', 'study_method',
       'facility_rating', 'exam_difficulty', 'exam_score'],
      dtype='object')
```

Label encoding

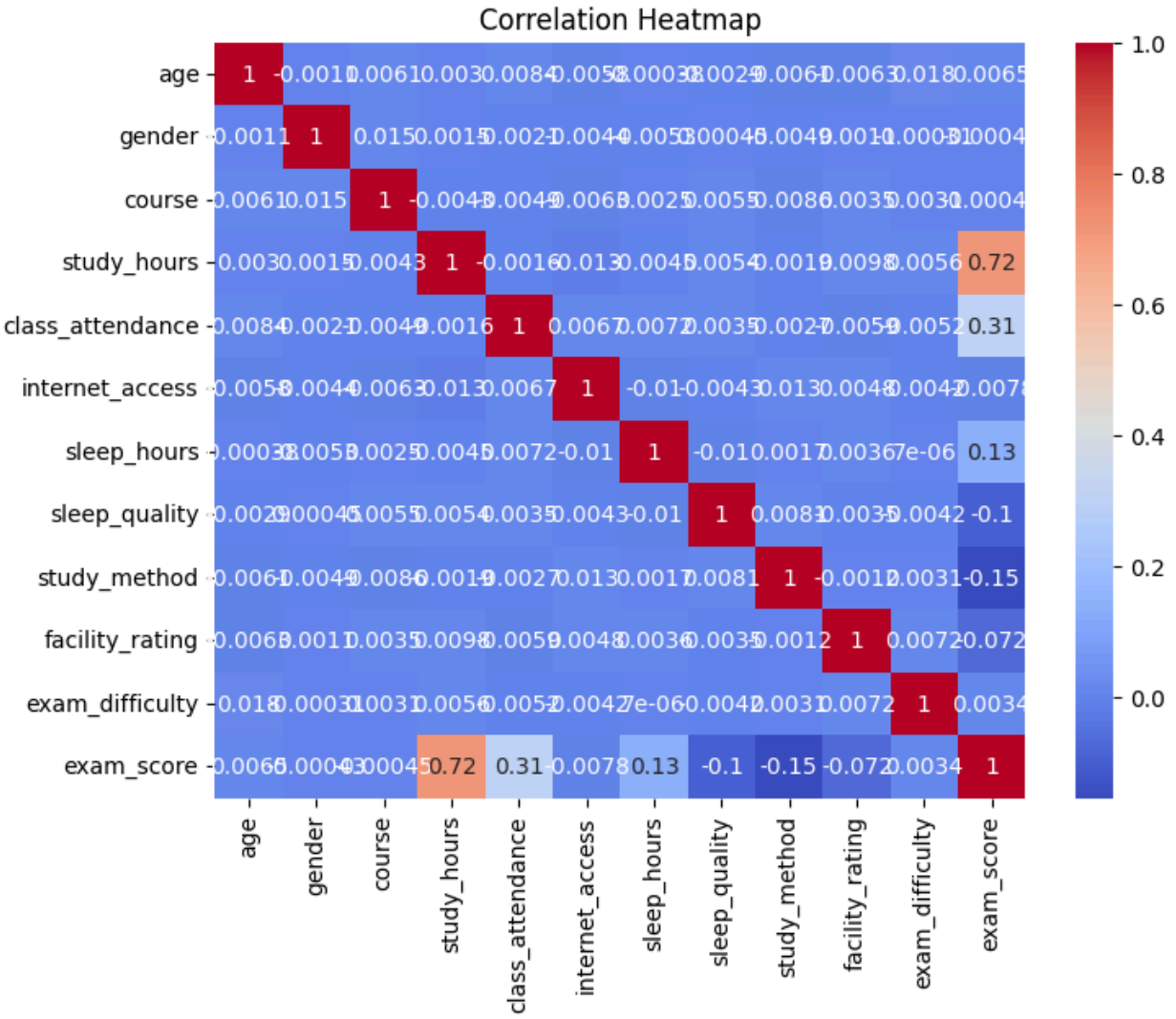
In [ ]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
categorical_cols = [
    "gender", "course", "internet_access", "sleep_quality",
    "study_method", "facility_rating", "exam_difficulty"
]

for col in categorical_cols:
  df[col]=le.fit_transform(df[col])
```

Data Visulalization

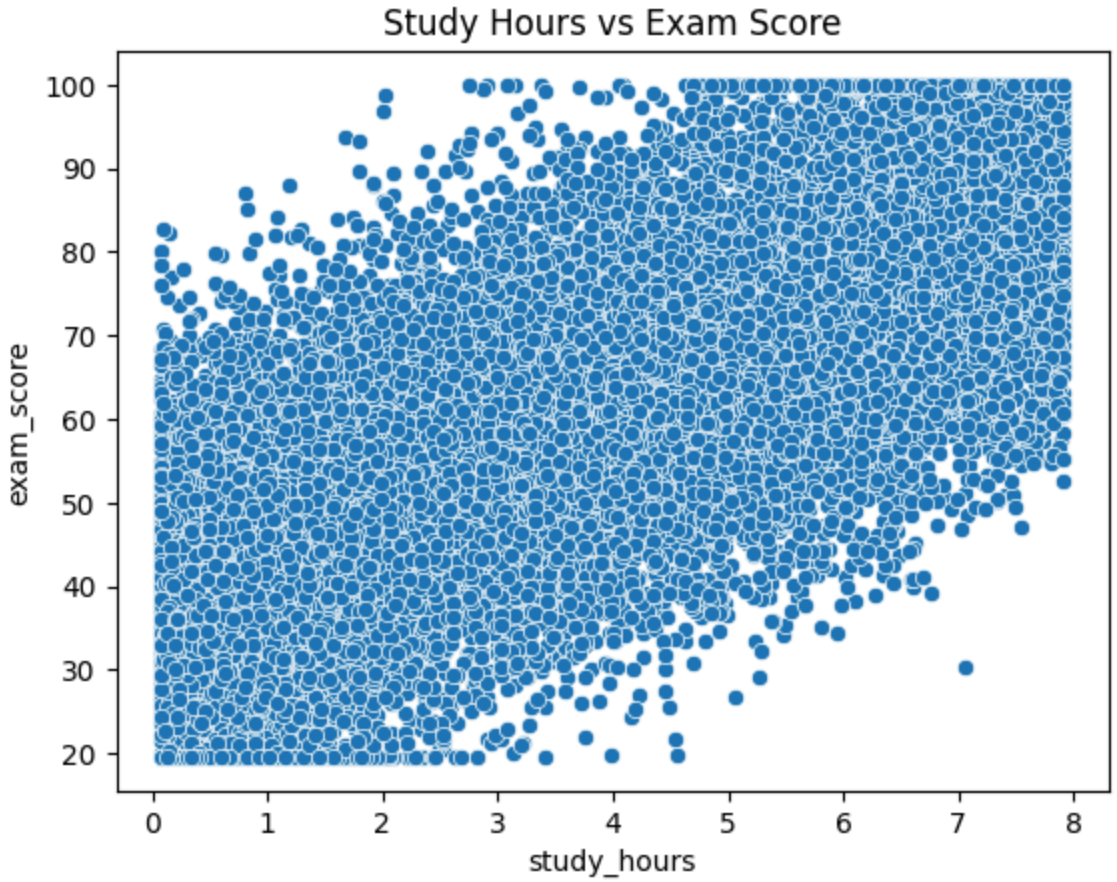
In [ ]:
```
sns.histplot(df["exam_score"], kde=True)
plt.title("Distribution of Exam Scores")
plt.show()
```



In [ ]:
```
plt.figure(figsize=(8,6))
sns.heatmap(df.corr(), annot=True, cmap="coolwarm")
plt.title("Correlation Heatmap")
plt.show()
```



In [ ]:
```
sns.scatterplot(x="study_hours", y="exam_score", data=df)
plt.title("Study Hours vs Exam Score")
plt.show()
```



In [ ]:
```
df["Result"] = df["exam_score"].apply(
    lambda x: 1 if x >= 50 else 0
)
```

In [ ]:
```
df = df.drop("exam_score", axis=1)
```

SVM

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

In [ ]:
```
#Extracting datasets
x = df.iloc[:, :-1].values #independent
y = df.iloc[:, -1].values #dependent
```

In [ ]:
```
#Splitting the dataset into the Training set and Test set
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size=0.2, random
```

In [ ]:
```
# Feature Scaling
from sklearn.preprocessing import StandardScaler
st_x=StandardScaler()
x_train=st_x.fit_transform(x_train)
```

```
x_test=st_x.transform(x_test)
```

In [ ]:
```
from sklearn.svm import SVC #SUPPORT VECTOR CLASSIFIER
classifier= SVC(kernel='linear',random_state=0)
classifier.fit(x_train,y_train)
```

Out[ ]:
```
SVC(kernel='linear', random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[ ]:
```
array([[ 707,  398],
       [ 256, 2639]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8365

Classification Report:
              precision    recall  f1-score   support

           0       0.73      0.64      0.68      1105
           1       0.87      0.91      0.89      2895

    accuracy                           0.84      4000
   macro avg       0.80      0.78      0.79      4000
weighted avg       0.83      0.84      0.83      4000
```

Now SVM ON PARTICULAR COLUMNS

In [ ]:
```
selected_features = [
    "study_hours",
    "class_attendance",
    "study_method",
    "sleep_hours",
    "sleep_quality",
    "internet_access",
    "exam_difficulty",
    "age",
    "gender"
]
```

In [ ]:
```
X = df[selected_features].values
y = df["Result"].values
```

In [ ]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [ ]:
```
from sklearn.svm import SVC

classifier = SVC(kernel='linear', random_state=0)
classifier.fit(x_train, y_train)
```

Out[ ]:
```
SVC(kernel='linear', random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[ ]:
```
array([[ 709,  396],
       [ 254, 2641]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.8375

Classification Report:
              precision    recall  f1-score   support

           0       0.74      0.64      0.69      1105
           1       0.87      0.91      0.89      2895

    accuracy                           0.84      4000
   macro avg       0.80      0.78      0.79      4000
weighted avg       0.83      0.84      0.83      4000
```

In [ ]:
```
selected_features = [
    "study_hours",
    "class_attendance",
    "study_method",
    "sleep_hours",
    "sleep_quality",
    "internet_access",
    "exam_difficulty",
    "age",
    "gender"
]
```

In [ ]:
```
X = df[selected_features].values
y = df["Result"].values
```

In [ ]:
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=0
)
```

In [ ]:
```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
```

In [ ]:
```
from sklearn.svm import SVC

classifier = SVC(kernel='rbf', C=1, gamma='scale', random_state=0)
classifier.fit(x_train, y_train)
```

Out[ ]:
```
SVC
SVC(C=1, random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test, y_pred)
cm
```

Out[ ]:
```
array([[ 699,  406],
       [ 226, 2669]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score, classification_report
print("Accuracy:", accuracy_score(y_test, y_pred))
print("\nClassification Report:\n", classification_report(y_test, y_pred))
```

```
Accuracy: 0.842

Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.63      0.69      1105
           1       0.87      0.92      0.89      2895

    accuracy                           0.84      4000
   macro avg       0.81      0.78      0.79      4000
weighted avg       0.84      0.84      0.84      4000
```

LOGISTIC REGRESSION

TAKING ALL COLUMNS

In [ ]:
```
#Extracting datasets
x = df.iloc[:, :-1].values #independent
y = df.iloc[:, -1].values #dependent
```

In [ ]:
```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, rand
```

In [ ]:
```
#feature Scaling
#column=> mean=>
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In [ ]:
```
#Fitting Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

Out[ ]:
```
LogisticRegression
LogisticRegression(random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
cm
```

Out[ ]:
```
array([[ 884,  490],
       [ 326, 3300]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.84
```

TAKING PARTICULAR COLUMNS

In [ ]:
```
selected_features = [
    "study_hours",
    "class_attendance",
    "study_method",
    "sleep_hours",
    "sleep_quality",
    "internet_access",
    "exam_difficulty",
    "age",
    "gender"
]
```

In [ ]:
```
X = df[selected_features].values
y = df["Result"].values
```

In [ ]:
```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, rand
```

In [ ]:
```
#feature Scaling
#column=> mean=>
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In [ ]:
```
#Fitting Logistic Regression to the training set
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

Out[ ]:
```
▾ LogisticRegression
LogisticRegression(random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
```

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

Out[ ]:
```
array([[ 884,  490],
       [ 326, 3300]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.84
```

In [ ]:
```
corr_matrix = df.corr()
```

In [ ]:
```
corr_with_target = corr_matrix["Result"].sort_values(ascending=False)
print(corr_with_target)
```

```
Result              1.000000
study_hours         0.561711
class_attendance    0.221395
sleep_hours         0.096430
exam_difficulty     0.001381
age                 0.000972
course             -0.000088
gender             -0.001972
internet_access    -0.002469
facility_rating    -0.046881
sleep_quality      -0.081065
study_method       -0.111482
Name: Result, dtype: float64
```

In [ ]:
```
selected_features_corr = [
    "study_hours",
    "class_attendance"
]
```

In [ ]:
```
X = df[selected_features].values
y = df["Result"].values
```

In [ ]:
```
# Splitting the dataset into training and test set.
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test= train_test_split(x, y, test_size= 0.25, rand
```

In [ ]:
```
#feature Scaling
#column=> mean=>
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)
```

In [ ]:
```
#Fitting Logistic Regression to the training set
```

```
from sklearn.linear_model import LogisticRegression
classifier= LogisticRegression(random_state=0)
classifier.fit(x_train, y_train)
```

Out[ ]:

```
LogisticRegression
LogisticRegression(random_state=0)
```

In [ ]:
```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

In [ ]:
```
#Creating the Confusion matrix
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_pred)
cm
```

Out[ ]:
```
array([[ 884,  490],
       [ 326, 3300]])
```

In [ ]:
```
from sklearn.metrics import accuracy_score

accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy:.2f}")
```

```
Accuracy: 0.84
```