

**jupyter Practical 5** Last Checkpoint: 15 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipykernel)

```
In [6]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay, classification_report, accuracy_score, precision_score, recall_score
```

```
In [7]: #Loading the dataset and printing the first 5 rows of the dataset
data = pd.read_csv('Social_Network_Ads.csv')
data.head(5)
```

```
Out[7]:
```

	Age	EstimatedSalary	Purchased
0	19	19000	0
1	35	20000	0
2	26	43000	0
3	27	57000	0
4	19	76000	0

```
In [8]: data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 3 columns):
 #   Column            Non-Null Count  Dtype
---  -
 0   Age               400 non-null   int64
 1   EstimatedSalary   400 non-null   int64
 2   Purchased         400 non-null   int64
dtypes: int64(3)
memory usage: 9.5 KB
```

```
In [9]: # Checking Basic statistics of the dataset
data.describe()
```

```
Out[9]:
```

	Age	EstimatedSalary	Purchased
count	400.000000	400.000000	400.000000
mean	37.655000	69742.500000	0.357500
std	10.482877	34096.960282	0.479864
min	18.000000	15000.000000	0.000000
25%	29.750000	43000.000000	0.000000
50%	37.000000	70000.000000	0.000000

**jupyter Practical 5** Last Checkpoint: 15 hours ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted | Python 3 (ipykernel)

```
In [10]: #Checking for NAN values
data.isnull().sum()
```

```
Out[10]: Age      0
EstimatedSalary  0
Purchased      0
dtype: int64
```

```
In [11]: #Displaying dimensions of the dataframe
data.shape
```

```
Out[11]: (400, 3)
```

```
In [12]: #Loading 'Age', 'Estimated_salary' as the features in 'x'
#and 'Purchased' as the target variable in 'y'
x = data.iloc[:,2:4]
```

```
In [24]: y = data.iloc[:,2]
```

```
In [15]: #Splitting data into 75% training and 25% testing data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.25, random_state=42)
```

```
In [16]: #Using the StandardScaler to standardize the values
scale = StandardScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)
```

```
In [17]: #Preparing and Training our Logistic Regression model with
#the training and testing data
lr = LogisticRegression(random_state = 0, solver = 'lbfgs')
lr.fit(x_train,y_train)
pred = lr.predict(x_test)

print(x_test[:10])
print('-'*15)
print(pred[:10])

[[-0.73918317
  1.35284466
 -0.73918317
  1.35284466
 -0.73918317
 -0.73918317
  1.35284466
 -0.73918317
 -0.73918317
 -0.73918317]]
-----
[0 1 0 1 0 0 1 0 0 0]
```

```
In [18]: print('Expected Output:',pred[:10])
print('-'*15)
print('Predicted Output:\n',y_test[:10])
```

Expected Output: [0 1 0 1 0 0 1 0 0 0]

Predicted Output:

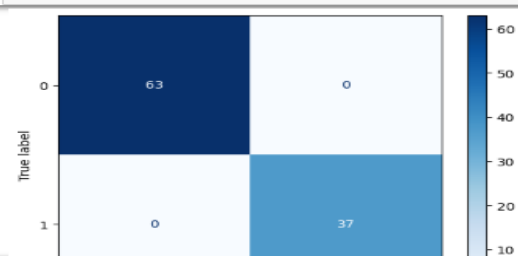
```
209 0
289 1
33 0
210 1
93 0
84 0
329 1
94 0
266 0
126 0
Name: Purchased, dtype: int64
```

```
In [19]: #Evaluating the model based on confusion matrix
matrix = confusion_matrix(y_test,pred,labels = lr.classes_)
print(matrix)

tp, fn, fp, tn = confusion_matrix(y_test,pred,labels=[1,0]).reshape(-1)
```

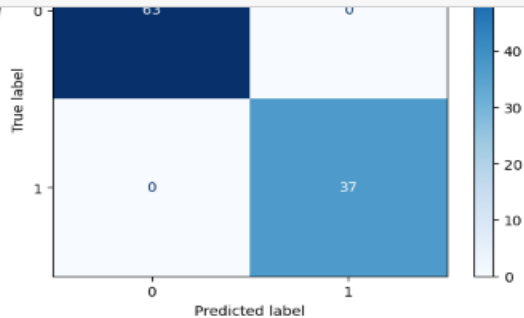
```
[[63  0]
 [ 0 37]]
```

```
In [20]: conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=lr.classes_)
conf_matrix.plot(cmap=plt.cm.Blues)
plt.show()
```



In [21]: #Determining the Accuracy,Error Rate,Recall,Precision of the model

```
In [20]: conf_matrix = ConfusionMatrixDisplay(confusion_matrix=matrix,display_labels=lr.classes_)
conf_matrix.plot(cmap=plt.cm.Blues)
plt.show()
```



```
In [21]: #Determining the Accuracy,Error Rate,Recall,Precision of the model
print(classification_report(y_test,pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	63
1	1.00	1.00	1.00	37
accuracy			1.00	100
macro avg	1.00	1.00	1.00	100
weighted avg	1.00	1.00	1.00	100

```
In [22]: print('\nAccuracy: {:.2f}'.format(accuracy_score(y_test,pred)))
print('Error Rate: ',(fp+fn)/(tp+tn+fn+fp))
print('Sensitivity (Recall or True positive rate) :',tp/(tp+fn))
print('Specificity (True negative rate) :',tn/(fp+tn))
print('Precision (Positive predictive value) :',tp/(tp+fp))
print('False Positive Rate :',fp/(tn+fp))
```

```
Accuracy: 1.00
Error Rate: 0.0
Sensitivity (Recall or True positive rate) : 1.0
Specificity (True negative rate) : 1.0
Precision (Positive predictive value) : 1.0
False Positive Rate : 0.0
```