

Practical No. 6

Name: Prathamesh Pawar | Roll No: B-23

Data Analytics III

1. Implement Simple Naïve Bayes classification algorithm using Python/R on iris.csv dataset.
2. Compute Confusion matrix to find TP, FP, TN, FN, Accuracy, Error rate, Precision, Recall on the given dataset.

```
In [8]: import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
import seaborn as sns
df = pd.read_csv('iris.csv')
df.head()
```

```
Out[8]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.4	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

```
In [9]: df.info()

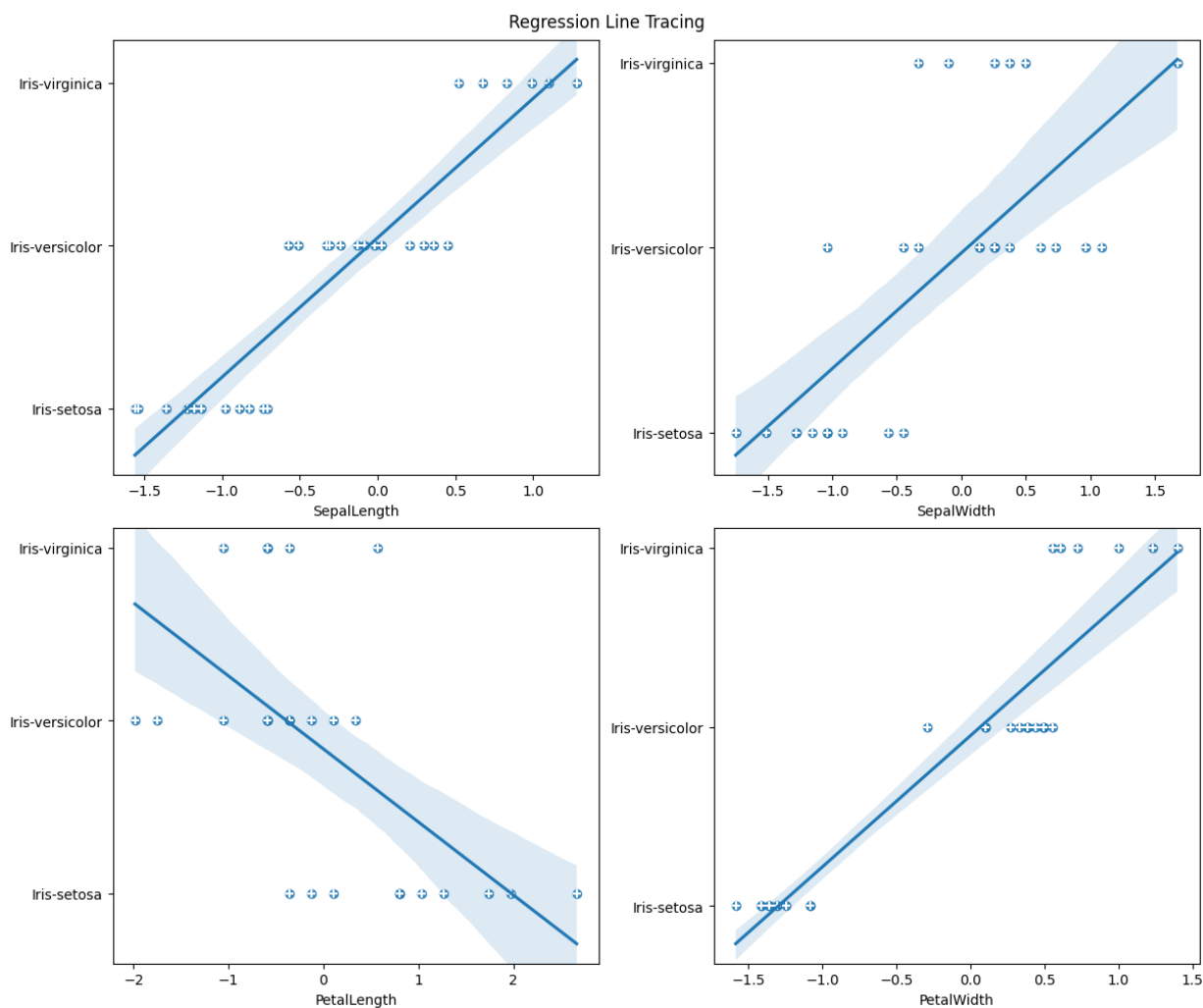
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 6 columns):
 #   Column          Non-Null Count  Dtype  
---  --
 0   Id              150 non-null   int64   
 1   SepalLengthCm   150 non-null   float64  
 2   SepalWidthCm    150 non-null   float64  
 3   PetalLengthCm   150 non-null   float64  
 4   PetalWidthCm    150 non-null   float64  
 5   Species         150 non-null   object  
dtypes: float64(4), int64(1), object(1)
memory usage: 7.2+ KB
```

```
In [10]: X = df.iloc[:, :4].values
Y = df['Species'].values
```

```
In [11]: from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state=0)
sc_X = StandardScaler()
X_train = sc_X.fit_transform(X_train)
X_test = sc_X.transform(X_test)
print(f'Train Dataset Size - X: {X_train.shape}, Y: {Y_train.shape}')
print(f'Test Dataset Size - X: {X_test.shape}, Y: {Y_test.shape}')

Train Dataset Size - X: (120, 4), Y: (120,)
Test Dataset Size - X: (30, 4), Y: (30,)
```

```
In [14]: from sklearn.naive_bayes import GaussianNB
classifier = GaussianNB()
classifier.fit(X_train, Y_train)
predictions = classifier.predict(X_test)
mapper = {'Iris-setosa': 0, 'Iris-versicolor': 1, 'Iris-virginica': 2}
predictions_ = [mapper[i] for i in predictions]
fig, axs = plt.subplots(2, 2, figsize = (12, 10), constrained_layout = True)
fig.suptitle('Regression Line Tracing')
for i in range(4):
    x, y = i // 2, i % 2
    sns.regplot(x = X_test[:, i], y = predictions_, ax=axs[x, y])
    axs[x, y].scatter(X_test[:, i][::-1], Y_test[:, i][::-1], marker = '+', color="white")
    axs[x, y].set_xlabel(df.columns[i + 1][:-2])
```



Confusion matrix

```
In [16]: from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
cm = confusion_matrix(Y_test, predictions)
print(f'''Confusion matrix :\n
| Positive Prediction\t| Negative Prediction
-----+-----
Positive Class | True Positive (TP) {cm[0, 0]}\t| False Negative (FN) {cm[0, 1]}
-----+-----
Negative Class | False Positive (FP) {cm[1, 0]}\t| True Negative (TN) {cm[1, 1]}\n''')
cm = classification_report(Y_test, predictions)
print('Classification report : \n', cm)
```

Confusion matrix :

	Positive Prediction	Negative Prediction
Positive Class	True Positive (TP) 11	False Negative (FN) 0
Negative Class	False Positive (FP) 0	True Negative (TN) 13

Classification report :

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30