DSBDL Assignment 10 - Data Analytics 3

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.

```
from google.colab import drive
drive.mount('/content/drive')
```

```
     Mounted at /content/drive
```

```
import numpy as np
import seaborn as sns
import pandas as pd
```

```
ds = pd.read_csv('/content/drive/My Drive/DSBDL/Assignment10/iris.csv')
ds
```

	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
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	virginica
146	6.3	2.5	5.0	1.9	virginica
147	6.5	3.0	5.2	2.0	virginica
148	6.2	3.4	5.4	2.3	virginica
149	5.9	3.0	5.1	1.8	virginica

150 rows × 5 columns

Next steps: Generate code with ds   View recommended plots

```
features = ds.drop( [ "species" ] , axis=1 ).columns
label = "species"
classes = np.unique( ds[label] )
print( features )
print( label )
print( classes )
```

```
Index(['sepal_length', 'sepal_width', 'petal_length', 'petal_width'], dtype='object')
species
['setosa' 'versicolor' 'virginica']
```

```
def normalize( feature ):
    ds[ feature ] = ( ds[ feature ] - ds[ feature ].min() ) / ( ds[feature].max() - ds[fe

for feature in features:
    normalize( feature )
ds
```

	sepal_length	sepal_width	petal_length	petal_width	species
0	0.222222	0.625000	0.067797	0.041667	setosa
1	0.166667	0.416667	0.067797	0.041667	setosa
2	0.111111	0.500000	0.050847	0.041667	setosa
3	0.083333	0.458333	0.084746	0.041667	setosa
4	0.194444	0.666667	0.067797	0.041667	setosa
...	...	...	...	...	...
145	0.666667	0.416667	0.711864	0.916667	virginica
146	0.555556	0.208333	0.677966	0.750000	virginica
147	0.611111	0.416667	0.711864	0.791667	virginica
148	0.527778	0.583333	0.745763	0.916667	virginica
149	0.444444	0.416667	0.694915	0.708333	virginica

150 rows × 5 columns

Next steps: Generate code with ds View recommended plots

```
import math

def gaussian_pdf( mean: float , std: float ):
    def pdf( x ):
        return (1.0 / (std * math.sqrt( 2 * math.pi ))) * math.exp( -0.5 * ((x - mean)/st
    return pdf

def compute_prior( class_: str ) -> float:
    return len( train_ds[ train_ds["species"] == class_ ] ) / len( train_ds )

def compute_log_likelihood( sample , class_: str ) -> float:
    prior_prob = compute_prior( class_ )
    posterior_prob = 0.0
    for i , feature in enumerate(features):
        pdf = gaussian_pdf(
            train_ds[ train_ds[label] == class_ ][feature].mean() ,
            train_ds[ train_ds[label] == class_ ][feature].std()
        )
        posterior_prob += math.log( pdf( sample[i] ) )
    return posterior_prob + math.log( prior_prob )

def classifier( sample ):
    max_likelihood = -math.inf
    max_likelihood_class = None
    for class_ in classes:
        class_likelihood = compute_log_likelihood( sample , class_ )
        if class_likelihood > max_likelihood:
            max_likelihood = class_likelihood
            max_likelihood_class = class_
    return max_likelihood_class


def train_test_split( test_split = 0.3 ):
    global ds
    ds = ds.sample(frac=1)
    num_test_samples = int(len( ds ) * test_split)
    test_ds = ds.head( num_test_samples )
    train_ds = ds.tail( len(ds) - num_test_samples )
    return train_ds , test_ds

train_ds , test_ds = train_test_split()
target_labels = list( test_ds[label] )
pred_labels = []
for test_sample in test_ds.to_numpy():
    pred_y = classifier( test_sample[ : -1 ] )
    pred_labels.append( pred_y )


print( pred_labels )
```

```
['versicolor', 'versicolor', 'setosa', 'versicolor', 'virginica', 'virginica', 'versi
```

```
print( target_labels )
```

```
['versicolor', 'versicolor', 'setosa', 'versicolor', 'virginica', 'virginica', 'versi
```

```
# Accuracy
acc = sum( [ 1.0 for i in range( len( pred_labels ) ) if pred_labels[i] == target_labels[i
print( f"Accuracy is {acc}" )
```

```
Accuracy is 0.9777777777777777
```

Start coding or generate with AI.