Classification of Rice Varieties using Gaussian Naïve Bayes Learning Model

Team: Sudeep Hansda – 22CS60R74
Gajendra Singh – 22ID60R20
Anuj Kakde – 20CS30005

Introduction

Created a model to predict the quality of new rice samples based on their characteristics using the Gaussian Naive Bayes algorithm. This model was trained using 5-fold validation. Also calculated Accuracy, Precision, Recall, f1-score and support of the model. These scores are calculated for hyper-tuned model also.

Gaussian Equation:

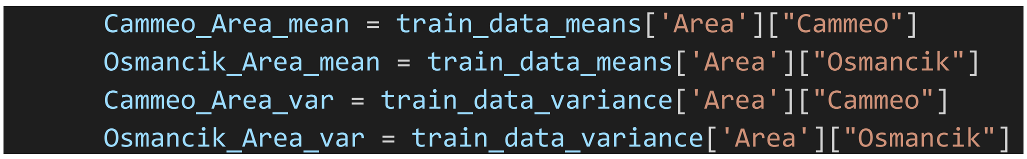
$$y = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

$$\begin{aligned} \mu &= \text{Mean} \\ \sigma &= \text{Standard Deviation} \\ \pi &\approx 3.14159\cdots \\ e &\approx 2.71828\cdots \end{aligned}$$

This equation is used to train the model.

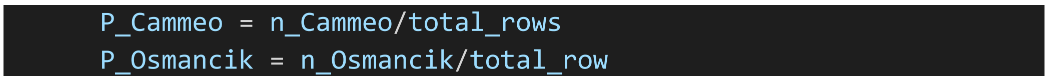
Explanation:

- First, a function is defined with the name guassian_function() which takes the three parameters and returns the gaussian value.
- Dataset is shuffled using the pandas' sample function and it is split into parts: Test (30%) and Training (70%)
- Means and Variances are calculated for each feature after grouping by class. This data contains two classes(Cammeo and Osmancik). Below is the Screenshot of calculating the "Area" mean for the Osmancik group.



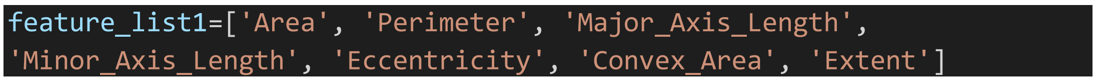
```
Cammeo_Area_mean = train_data_means['Area']["Cammeo"]
Osmancik_Area_mean = train_data_means['Area']["Osmancik"]
Cammeo_Area_var = train_data_variance['Area']["Cammeo"]
Osmancik_Area_var = train_data_variance['Area']["Osmancik"]
```

- All variance and means of all features are calculated and stored in separate variables, which are used to train the model.
- The calculated variance and means are stored in two different dictionaries named Osmancik_dict and Cammeo_dict.
- Prior probability of both classes is calculated using only the trained data. Below is the screenshot.



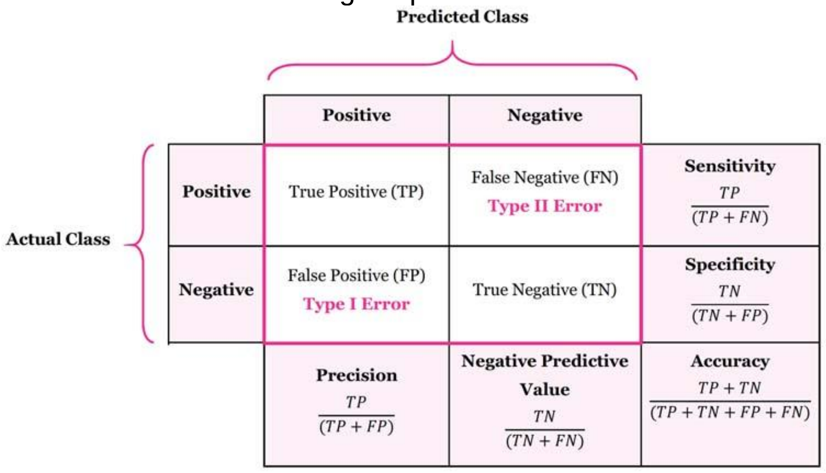
```
P_Cammeo = n_Cammeo/total_rows
P_Osmancik = n_Osmancik/total_row
```

- A list is maintained for the features on which the model is trained. Below is the screenshot.



```
feature_list1=['Area', 'Perimeter', 'Major_Axis_Length',
'Minor_Axis_Length', 'Eccentricity', 'Convex_Area', 'Extent']
```

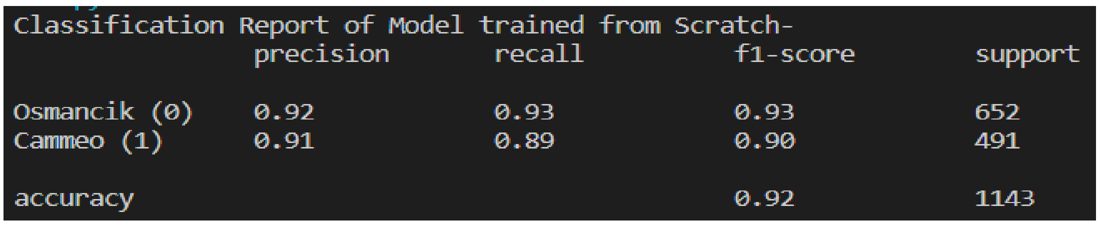
- This list allows the flexibility to decide the features on which the model will be trained. Only the required features can be added in the list and redundant features can be deleted from the list. It is also used for hypertuning (removing the dependent features (Perimeter and Convex_Area)).
- In this dataset, Osmancik is considered positive, and Cammeo is negative.
- For the confusion matrix, True Positive(t_p), True Negative(t_n), False Positive(f_p), False Negative(f_n). (Variable name in the program).This values are calculated using simple if-else Condition.



Predicted Class

Actual Class		Positive	Negative	
	Positive	True Positive (TP)	False Negative (FN) **Type II Error**	**Sensitivity** $\frac{TP}{(TP+FN)}$
	Negative	False Positive (FP) **Type I Error**	True Negative (TN)	**Specificity** $\frac{TN}{(TN+FP)}$
		Precision $\frac{TP}{(TP+FP)}$	**Negative Predictive Value** $\frac{TN}{(TN+FN)}$	**Accuracy** $\frac{TP+TN}{(TP+TN+FP+FN)}$

- The Accuracy, Precision, Recall, f1-score and Support are calculated using the above values. These scores are displayed in a tabular format.

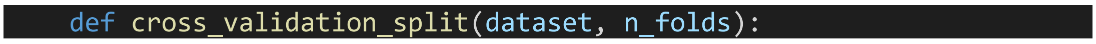


```
Classification Report of Model trained from Scratch-
                precision       recall          f1-score        support

Osmancik (0)    0.92            0.93            0.93            652
Cammeo (1)      0.91            0.89            0.90            491

accuracy                                        0.92            1143
```

- After K-fold validation, the model which gives the highest accuracy on validation data is the final model.



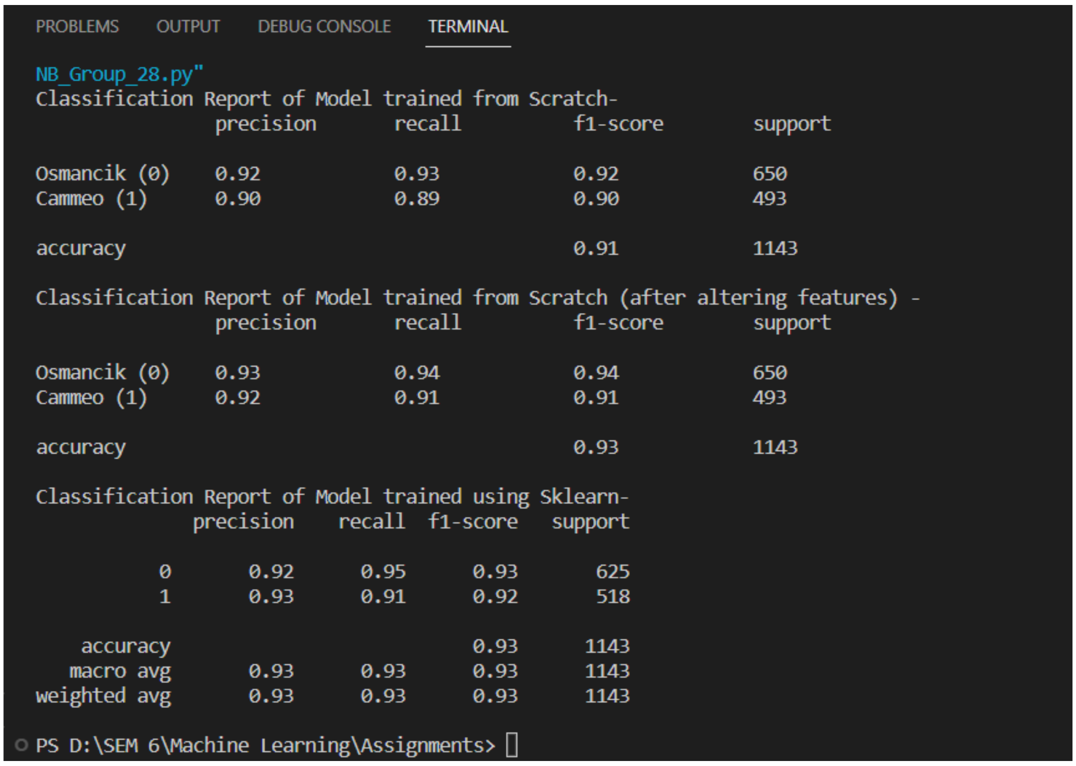
```
    def cross_validation_split(dataset, n_folds):
```

- This final model is then used to predict the test data and the classification report is printed on the terminal.

❖ **Conclusion:**

This model is built from scratch using only Pandas and Numpy libraries. Also the same dataset is tested using Sklearn Gaussian Naïve Bayes Model for comparision.

Below is the classification report for both the models in tabular form (with and without hyperparameter tuning). The classification report generated by sklearn is also presented for comparison. Precision, recall , f1-score and support of the model you for each class is shown.



```
NB_Group_28.py"
Classification Report of Model trained from Scratch-
                precision       recall          f1-score        support

Osmancik (0)    0.92            0.93            0.92            650
Cammeo (1)      0.90            0.89            0.90            493

accuracy                                        0.91            1143

Classification Report of Model trained from Scratch (after altering features) -
                precision       recall          f1-score        support

Osmancik (0)    0.93            0.94            0.94            650
Cammeo (1)      0.92            0.91            0.91            493

accuracy                                        0.93            1143

Classification Report of Model trained using Sklearn-
              precision    recall  f1-score   support

           0       0.92      0.95      0.93       625
           1       0.93      0.91      0.92       518

    accuracy                           0.93      1143
   macro avg       0.93      0.93      0.93      1143
weighted avg       0.93      0.93      0.93      1143

PS D:\SEM 6\Machine Learning\Assignments>
```

The accuracy of the model trained from scratch is almost same as that calculated by sklearn. The accuracy increases after tuning the parameters (Perimeter and Convex_Area features are dropped since they are fairly dependent on other features).