Report

ELG5255[EG] APPLIED MACHINE LEARNING [LEC] 20225

Assignment 1 (Classification)

GROUP 11

MOSTAFA MAHMOUD ABD ELWAHAB NOFAL
NADA ABDELLATEF SHAKER SEDIK
HADEER MAMDOUH ADBELFATTAH MOHAMMED

Objective:

The purpose of this assignment is to do multiclass classification on ("DUMD_ dataset"). using OvR, OvO, perceptron, and SVM techniques.

Implementation:

Import libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from mlxtend.plotting import plot_decision_regions

from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2
from sklearn.linear_model import Perceptron
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MultiLabelBinarizer

from sklearn import svm
from sklearn.metrics import classification_report, ConfusionMatrixDisplay, accuracy_score, confusion_matrix
```

Figure 1: Import Libraries

Load and import the dataset using pandas.

Read the files

```
[ ] df_train=pd.read_csv("DUMD_train.csv")
    df_test=pd.read_csv("DUMD_test.csv")
```

Figure 2: Load Data

Data Preparation&Evaluation:

Feature Selection

x_train

	STG	SCG	STR	LPR	PEG
0	0.00	0.00	0.00	0.00	0.00
1	0.08	0.08	0.10	0.24	0.90
2	0.10	0.10	0.15	0.65	0.30
3	0.08	0.08	0.08	0.98	0.24
4	0.09	0.15	0.40	0.10	0.66
...	...	...	...	...	...

Figure 3: Features Before Chi-Square

x_train

	LPR	PEG
0	0.00	0.00
1	0.24	0.90
2	0.65	0.30
3	0.98	0.24
4	0.10	0.66
...	...	...

Figure 4: Features After Chi-Square

We have used SelectKBest to select the features with best chi-square, we have passed two parameters one is the scoring metric that is chi2 and other is the value of K which signifies the number of features we want in final dataset. We have used fit_transform to fit and transform the current dataset into the desired dataset. Finally, we have printed the final dataset and the shape of initial and final dataset.

Chi-Square is to be used when the feature is categorical, the target variable is any way can be thought as categorical. It measures the degree of association between two categorical variables.

Label Encoding

We used the LabelEncoder function to convert the string values to numeric.

```
labelencoder_y_train=LabelEncoder()
labelencoder_y_test=LabelEncoder()
y_train_encoded = labelencoder_y_train.fit_transform(y_train)
y_test_encoded =labelencoder_y_test.fit_transform(y_test)
y_train_encoded

array([3, 0, 2, 1, 2, 2, 1, 0, 0, 1, 2, 0, 3, 1, 1, 2, 1, 2, 2, 1, 1, 0,
```

Figure 5: Label Encoder

Plot the data

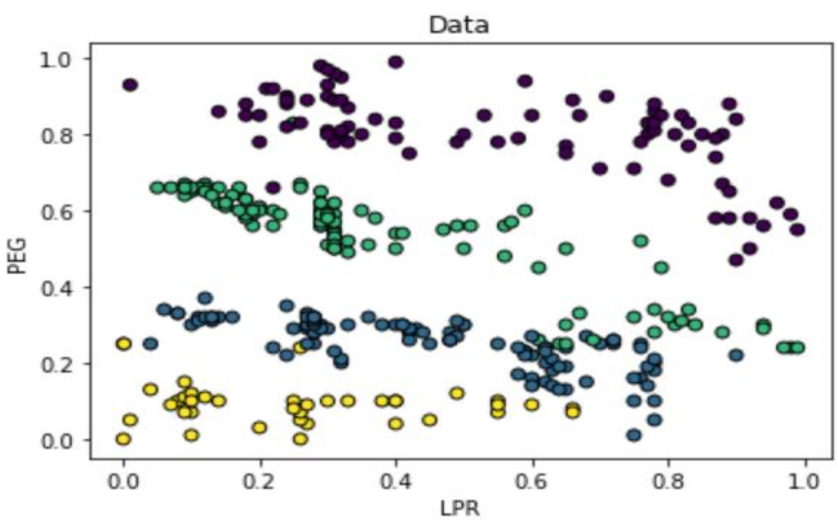


Figure 6: Data Plotting

Problem One: Applying the SVM model & Applying the perceptron model

- Applying the machine learning algorithm (Perceptron and SVM) to the dataset, after applying both algorithms, the accuracy of the SVM model (98.75%) is better than the Perceptron model (60%)

1- Train the SVM model

```
[ ]  svm_model = svm.SVC()
     svm_model.fit(x_train, y_train_encoded)
     ys_predicted = svm_model.predict(x_test)
```

Figure 7: SVM Model

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.96      1.00      0.98        26
           2       1.00      1.00      1.00        22
           3       1.00      0.91      0.95        11

    accuracy                           0.99        80
   macro avg       0.99      0.98      0.98        80
weighted avg       0.99      0.99      0.99        80

Confusion Matrix:

[[21  0  0  0]
 [ 0 26  0  0]
 [ 0  0 22  0]
 [ 0  1  0 10]]

Accuracy Score:

0.9875
```

Figure 10: Accuracy of SVM

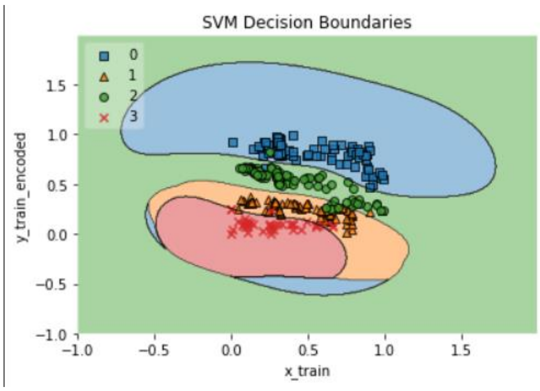


Figure 9: Decision Boundaries

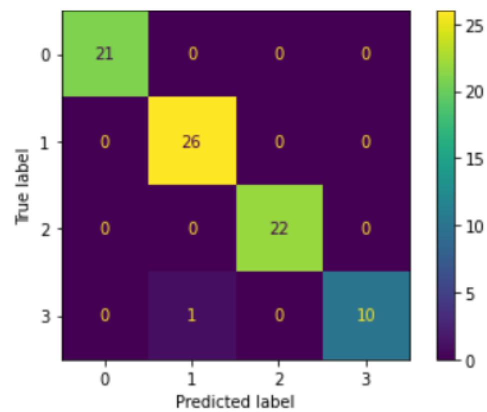


Figure 8: Confusion Matrix

Train the perceptron algorithm

2-

```
[ ]  perc_model = Perceptron(tol=1e-3, random_state=0)
     perc_model.fit(x_train, y_train_encoded)
     yp_predicted = perc_model.predict(x_test)
```

Figure 11: Perceptron Algorithm

```
Classification Report:

              precision    recall  f1-score   support

           0       0.95      1.00      0.98        21
           1       0.46      1.00      0.63        26
           2       0.00      0.00      0.00        22
           3       1.00      0.09      0.17        11

    accuracy                           0.60        80
   macro avg       0.60      0.52      0.44        80
weighted avg       0.54      0.60      0.48        80

Confusion Matrix:

[[21  0  0  0]
 [ 0 26  0  0]
 [ 1 21  0  0]
 [ 0 10  0  1]]

Accuracy Score:
```

Figure 12: Perceptron Model Accuracy

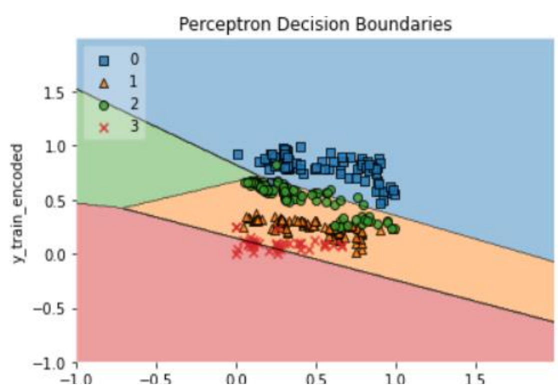


Figure 14: Perceptron Decision Boundaries

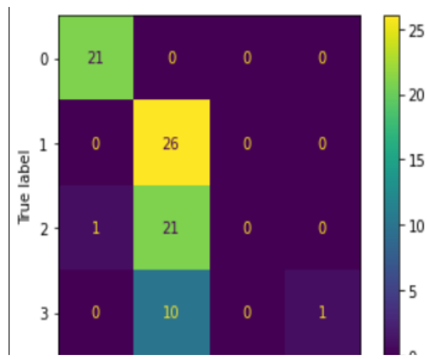


Figure 13: Confusion Matrix of Perceptron Model

Problem Two (One v Rest)

Label Binarizer

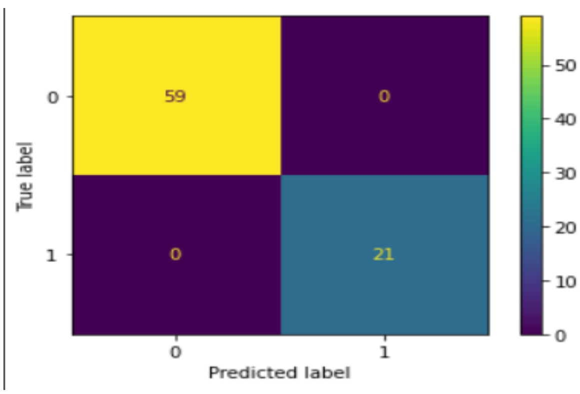
We used the MultiLabelBinarizer function to separate each label to its class, so we have 4 models for the 4 classes

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        59
           1       1.00      1.00      1.00        21

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80


Confusion Matrix:

[[59  0]
 [ 0 21]]

Accuracy Score:

1.0
```

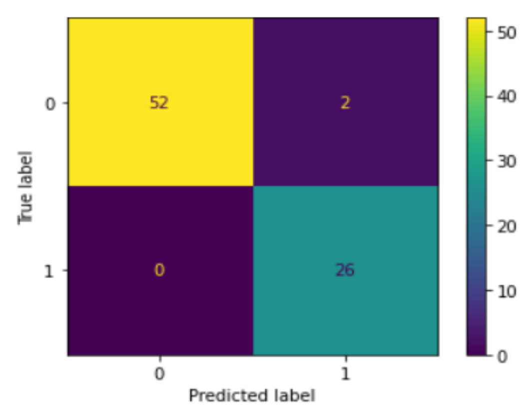
binarized model 1:Accuracy On SVM

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      0.96      0.98        54
           1       0.93      1.00      0.96        26

    accuracy                           0.97        80
   macro avg       0.96      0.98      0.97        80
weighted avg       0.98      0.97      0.98        80


Confusion Matrix:

[[52  2]
 [ 0 26]]

Accuracy Score:

0.975
```

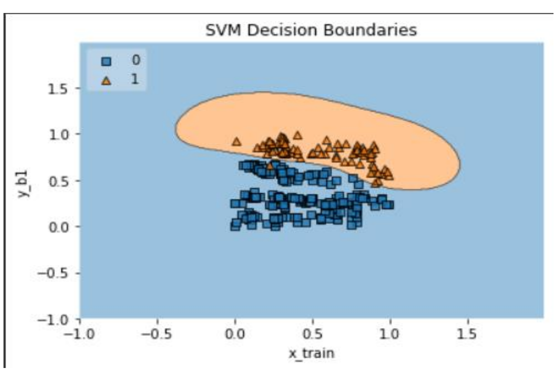
binarized model 2: Accuracy On SVM



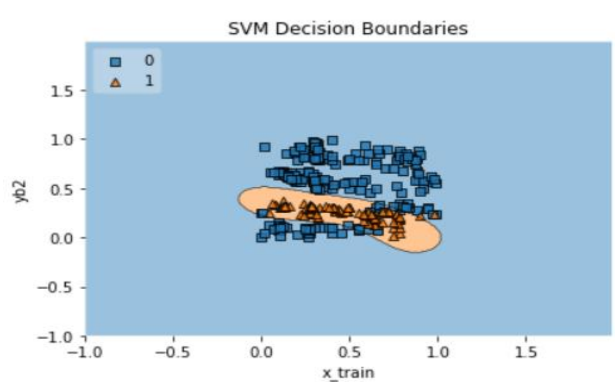
binarized model 1: Confusion Matrix



binarized model 2: Confusion Matrix



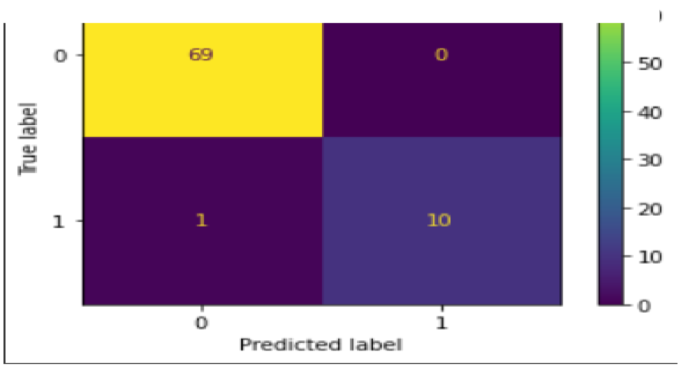
binarized model 1: Decision Boundaries



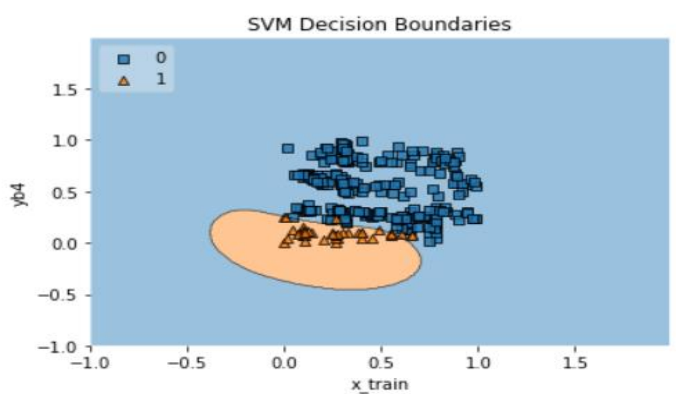
binarized model 2: Decision Boundaries

```
Classification Report:

              precision    recall  f1-score   support

           0       0.99      1.00      0.99        69
           1       1.00      0.91      0.95        11

    accuracy                           0.99        80
   macro avg       0.99      0.95      0.97        80
weighted avg       0.99      0.99      0.99        80


Confusion Matrix:

[[69  0]
 [ 1 10]]

Accuracy Score:

0.9875
```

binarized model 4: Accuracy



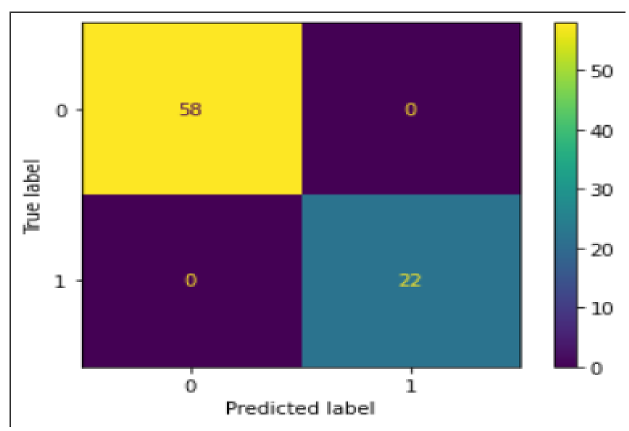
binarized model 4: Confusion Matrix



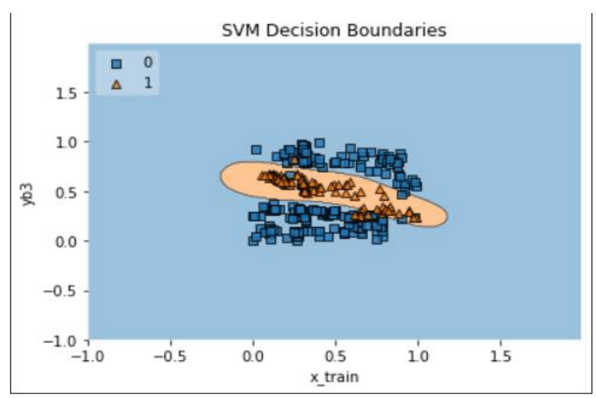
binarized model 4: Decision Boundaries

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        58
           1       1.00      1.00      1.00        22

    accuracy                           1.00        80
   macro avg       1.00      1.00      1.00        80
weighted avg       1.00      1.00      1.00        80


Confusion Matrix:

[[58  0]
 [ 0 22]]

Accuracy Score:

1.0
```

binarized model 3:Accuracy On SVM



binarized model 3: Confusion Matrix



binarized model 3: Decision Boundaries

Final label of OvR

We implemented a stack to store all the probabilities for the 4 binarized models so that we can use argmax function to aggregate the right class from the stack which has the highest probability.

Accuracies and confusion matrix for the final OvR model

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.96      1.00      0.98        26
           2       1.00      1.00      1.00        22
           3       1.00      0.91      0.95        11

    accuracy                           0.99        80
   macro avg       0.99      0.98      0.98        80
weighted avg       0.99      0.99      0.99        80


Confusion Matrix:

[[21  0  0  0]
 [ 0 26  0  0]
 [ 0  0 22  0]
 [ 0  1  0 10]]

Accuracy Score:

0.9875
```

Figure 17: Accuracy of Final Result

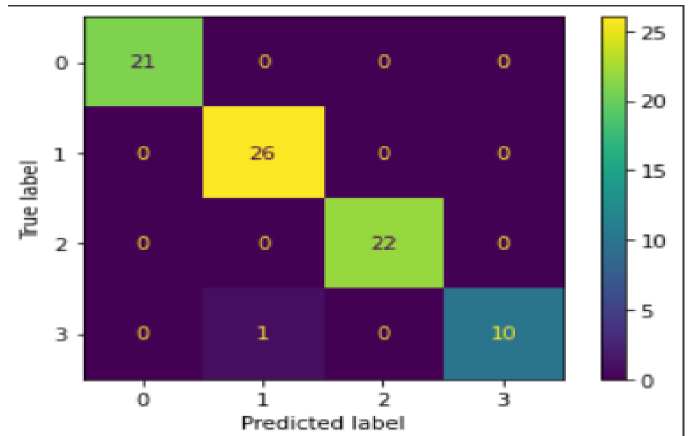


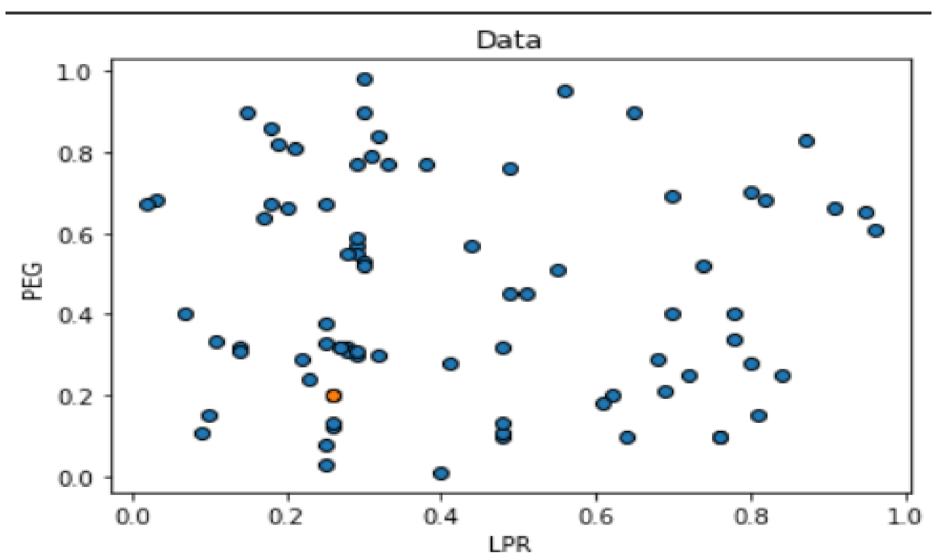
Figure 16: Confusion Matrix of Final Result

Plotting correct and wrong prediction points

We compared the predicted label with the actual label to spot the wrong prediction points and there was one wrong prediction. Then we searched for the features in our test samples that made this wrong prediction and plotted it

	LPR	PEG
64	0.26	0.2

Filtering the wrong prediction



plot test data and wrong data prediction

Problem 3: One v One

Preparing the data frames

We have 4 classes so with the formula when we substitute, it will be 6 models

$N(N-1)/2 \longrightarrow 4(4-1)/2 = 6$

Each model requires its training and testing data because we compare just two classes and drop the other two in each model

Models: Mode1_2, Model1_3, Model1_4, Model2_3, Model2_4, Model3_4

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       1.00      1.00      1.00        26

    accuracy                           1.00        47
   macro avg       1.00      1.00      1.00        47
weighted avg       1.00      1.00      1.00        47


Confusion Matrix:

[[21  0]
 [ 0 26]]

Accuracy Score:

1.0
```

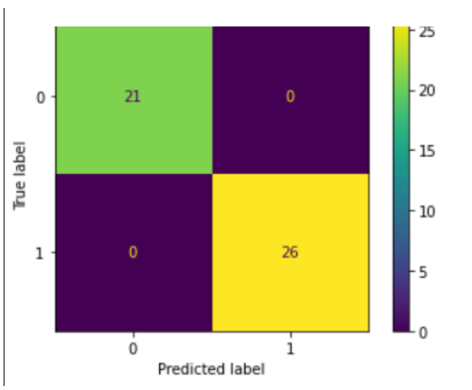
binarized model 1_2: Accuracy on SVM

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           2       1.00      1.00      1.00        22

    accuracy                           1.00        43
   macro avg       1.00      1.00      1.00        43
weighted avg       1.00      1.00      1.00        43


Confusion Matrix:

[[21  0]
 [ 0 22]]

Accuracy Score:

1.0
```

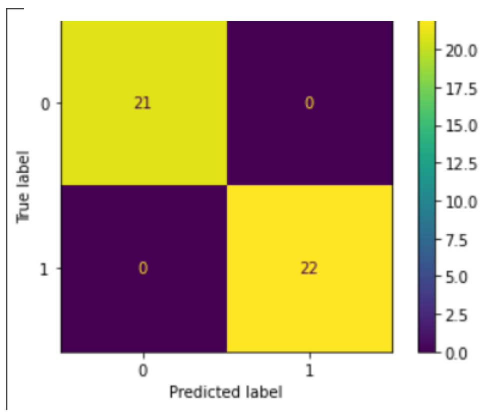
binarized model 1_3: Accuracy on SVM

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           3       1.00      1.00      1.00        11

    accuracy                           1.00        32
   macro avg       1.00      1.00      1.00        32
weighted avg       1.00      1.00      1.00        32


Confusion Matrix:

[[21  0]
 [ 0 11]]

Accuracy Score:

1.0
```

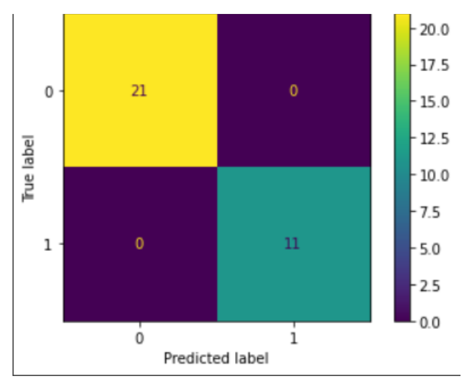
binarized model 1_4: Accuracy on SVM



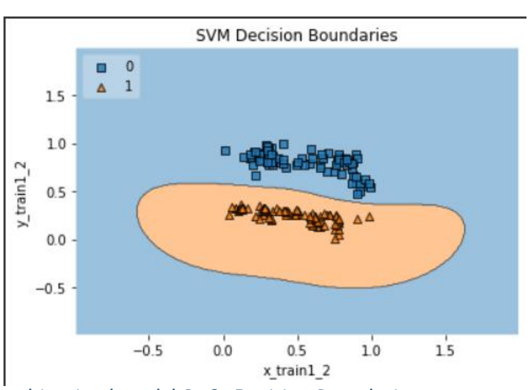
binarized model 1_2: Confusion Matrix



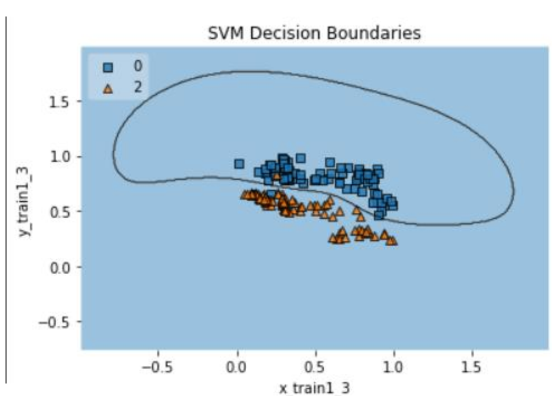
binarized model 1_3: Confusion Matrix



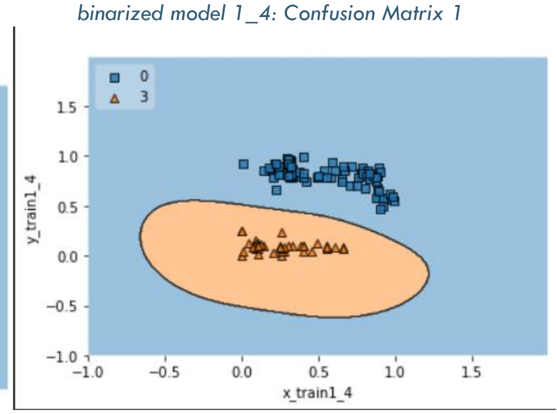
binarized model 1_4: Confusion Matrix 1



binarized model 1_2: Decision Boundaries



binarized model 1_3: Decision Boundaries



binarized model 1_4: Decision Boundaries

```
Classification Report:

              precision    recall  f1-score   support

           1       1.00      1.00      1.00        26
           2       1.00      1.00      1.00        22

    accuracy                           1.00        48
   macro avg       1.00      1.00      1.00        48
weighted avg       1.00      1.00      1.00        48


Confusion Matrix:

[[26  0]
 [ 0 22]]

Accuracy Score:

1.0
```

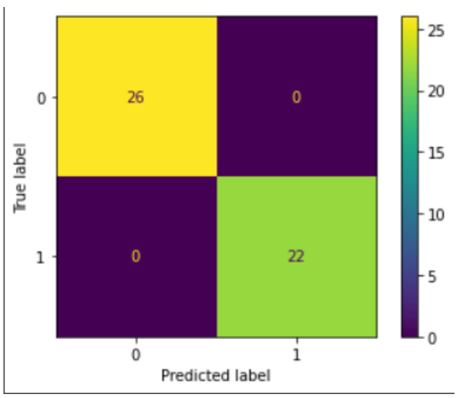
binarized model 2_3: Accuracy on SVM

```
Classification Report:

              precision    recall  f1-score   support

           1       0.96      1.00      0.98        26
           3       1.00      0.91      0.95        11

    accuracy                           0.97        37
   macro avg       0.98      0.95      0.97        37
weighted avg       0.97      0.97      0.97        37


Confusion Matrix:

[[26  0]
 [ 1 10]]

Accuracy Score:

0.972972972972973
```

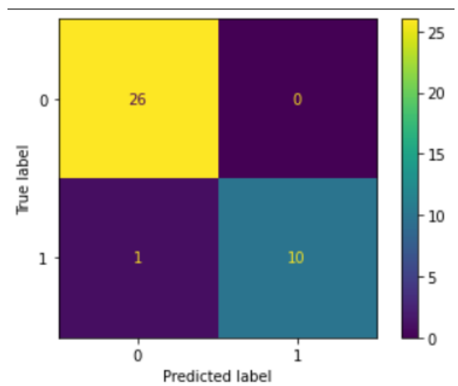
binarized model 2_4: Confusion Matrix

```
Classification Report:

              precision    recall  f1-score   support

           2       1.00      1.00      1.00        22
           3       1.00      1.00      1.00        11

    accuracy                           1.00        33
   macro avg       1.00      1.00      1.00        33
weighted avg       1.00      1.00      1.00        33


Confusion Matrix:

[[22  0]
 [ 0 11]]

Accuracy Score:

1.0
```

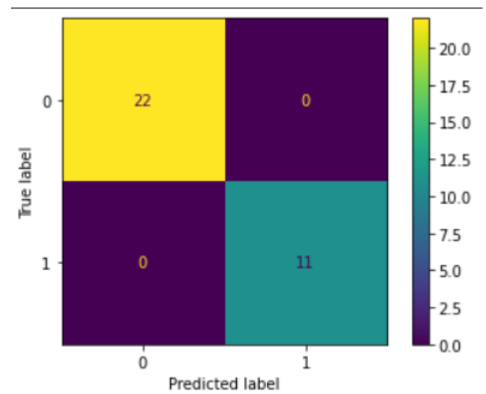
binarized model 3_4: Accuracy on SVM



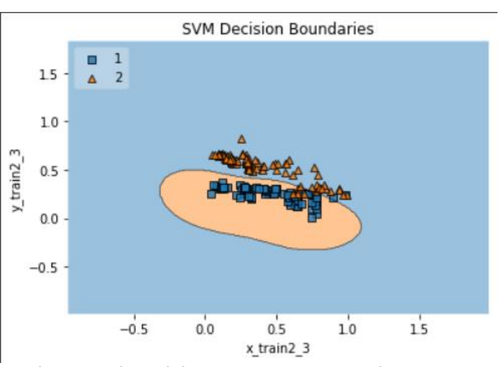
binarized model 2_3: Accuracy on SVM



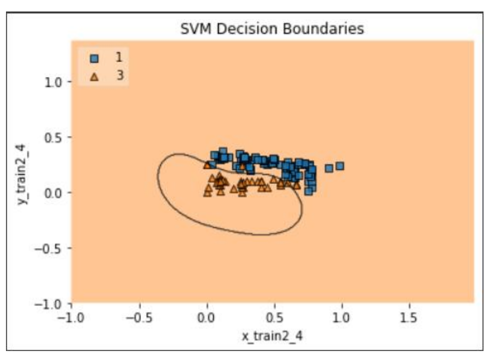
binarized model 2_4: Confusion Matrix



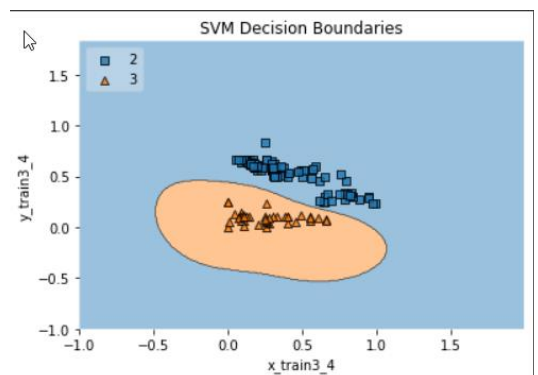
binarized model 3_4: Confusion Matrix



binarized model 2_3: Decision Boundaries



binarized model 2_4: Decision Boundaries



binarized model 3_4: Decision Boundaries

Final label of OvO:

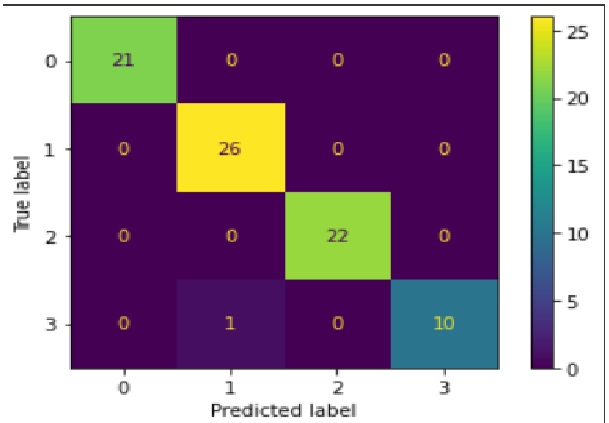
Each class of the 4 has 3 probabilities in the models out of 6 so we need to sum the probabilities of each of the 3 models to combine its probability.

Then we implement the same stack to store all the probabilities for the 4 binarized models so that we can use argmax function to aggregate the right class from the stack which has the highest probability

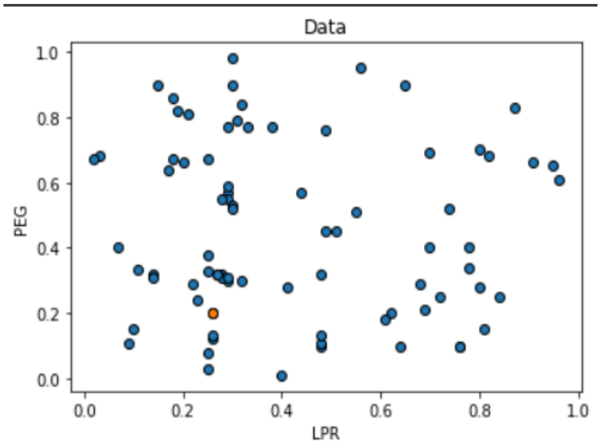
Compare the predicted label with the actual test data to calculate the accuracy of the model.

Accuracies and confusion matrix for the final OvO model

```
Classification Report:

              precision    recall  f1-score   support

           0       1.00      1.00      1.00        21
           1       0.96      1.00      0.98        26
           2       1.00      1.00      1.00        22
           3       1.00      0.91      0.95        11

    accuracy                           0.99        80
   macro avg       0.99      0.98      0.98        80
weighted avg       0.99      0.99      0.99        80


Confusion Matrix:

[[21  0  0  0]
 [ 0 26  0  0]
 [ 0  0 22  0]
 [ 0  1  0 10]]

Accuracy Score:

0.9875
```



Plotting correct and wrong prediction points



	LPR	PEG
64	0.26	0.2

Filtering the wrong prediction

3. Conclusion:

In conclusion, this report could be summarized by loading the data from "DUMD" dataset. For the feature engineering after splitting the data, we chose chi-square to select our two most important features. We plotted the data to make sure that it is separated and our decision on choosing the chi-square was correct. Then we used the label encoder to change the label to numeric. We applied support vector machine and perceptron models on the data. SVM model performance was very good at differentiating the classes. Its accuracy was (98.75) in comparison of the perceptron's accuracy which was (60%). We applied one versus rest (OvR) and one versus one (OvO) classification techniques. Argmax function was used in both to aggregate confidence scores from the binarized models and to obtain the final label's performance. The evaluation for each model was measured by confusion matrix and the data was visualized to show how the model was able to classify the data. Finally, we ended with the same accuracy which was (98.75%).

We learned the concept of feature engineering and how to choose the most important features for the training. The way of implementing and comparing different models on our data. Also, we understood the meaning of the one versus one (OvO) and one versus rest (OvR) multi-model classification concept.