**Seeds Classification – SVM and Perceptron**

**Problem Statement:**

The goal of Supervised multiclass classification algorithms is to assig a class label for each input in the two-class case, where the labelsare just +1 or -1 for the involved two classes. Some algorithms can be naturally extended to the multiclass case. and this what will be applied here.

**Data Set:**

Data is the basic pillar here so we didn’t save any effort to understand its nature and collect any piece of information that might help us building the model.

Seeds data set, numerical data with 169 instances (0 - 168) with two features and one target.

Table

Description automatically generated

The target has three values (1 “*Kama*”, 2 “*Rosa*”, 3 “*Canadian*”), for 1 and 3 there’re 58 values, and 53 for 2, All of this for Training data. for Testing data we have 42 values with different targets.

A picture containing text

Description automatically generated

And this is the discribtion of the data just to figure out how does it look like.

Table

Description automatically generated

**Data Plotting**

In classification problems like SVM or Perceptron, the data need to be plotted to see if it’s linearly seperable or not, and in this case it seems to be linearly seprable with acceptable accuracy.

Chart, scatter chart

Description automatically generated

This is the test set that will be used to check the accuracy of the model that’s going to be used later.

Chart, scatter chart

Description automatically generated

**Binary Classification**

After dropping the Kama data, we will have two classes to be classified, the plot shows that the data is linearly separable, so let’s see what our models will act like.

Chart, scatter chart

Description automatically generated

**SVM One vs One**

Support Vector Machines separates the data perfectly with 100% accuracy with test data.

Chart, scatter chart

Description automatically generated Chart

Description automatically generated

**Perceptron One vs One**

Perceptron classifier didn’t manage to draw the correct line between the two classes in the training data, that’s why it shows worse accuracy with the testing points.

Chart, scatter chart

Description automatically generated Chart, treemap chart

Description automatically generated

**Multi-class Classification**

Multi-class classification is those tasks where examples are assigned exactly one of more than two classes, some algorithms are designed for this, but some are designed for binary classification like SVM and Perceptron. as such they cannot be used for multi-class classification directly.

Instead, heuristic methods can be used to split them into multiple binary classification datasets and train a binary classification model for each one. We used one of them is **One-vs-One (OvO)**, that we used before, after dropping one class “Kama” and the other is **One-vs-Rest (OvR)** that we are going to use with our models SVM and Perceptron after turning data into multiple binary classes.

**Data Binarization**

As mentioned before, data need to be split into multiple binary classes first.

Text

Description automatically generated with low confidence

This code turns the data from (1, 2, 3) labels as whole into 2 labels for each one. for example, data belongs to 1 class will be labeled as 1, otherwise will be 0, and the same with the two other labels so are going to have three more columns for each label as shown below.

Text

Description automatically generated

Now, after we have the data binarized, it is ready to be classified using SVM or Perceptron OvR.

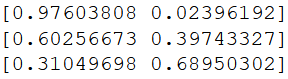
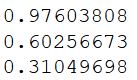
**SVM One vs Rest**

One vs Rest will combine One vs One for all the three classes. let’s talk about it in details.

**First step:**

the model is trained on the training data separately using OvO methodology, first it trains the two features with y1, then it does the same with the two other classes y2 and y3, In the testing phase the outcome will be the probability by which the point is classified as class 1 or 0.

For example, the first line shows that the model predicts the point to be in class 1 with more than 0.97 probability and belongs otherwise with about 0.024 for y1, so we take the probability of being in class 1.

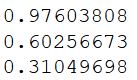
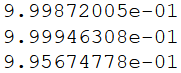
 

**Second step:**

now we have the results of the three probability lists for each class y1, y2 and y3 respectively, so we will compare between them and choose which model has the higher probability.

for the first point y1 predicts it to be in “1” class with more than 0.97 and y3 with .13 but y2 with more than .99 prob. so the model will choose the highest probability, and this means that it will predict it belongs to “2” class.

**y1 (1) y2 (2) y3 (3)**

  A picture containing text, orange

Description automatically generated

The output for these three points will be like [2, 2, 2]

But sometimes the output is inaccurate because the probabilities of two or more classes may be all high or low and this makes it hard to choose the best. in this case, classes one and two are both higher than 0.9 for the first point and here misclassification comes in.

The maximum probability here can be calculated using argmax or any code that does the same.

Text

Description automatically generated

This function for SVM model takes the training and testing data as inputs and in range of 3 for the three labels y1, y2 and y3 it trains and fits the model then it predicts on the testing data and print the confusion matrix and the classification report.

The accuracy and confusion matrices for y1, y2 and y3 respectively:

88.10 97.62 100.00

Chart

Description automatically generatedChart

Description automatically generatedChart

Description automatically generated

Classification report for y1:

Table

Description automatically generated with medium confidence

This fuction will be used to calculate the final accuracy of the three models aggregated comapring the predictions with the real targets.

Text

Description automatically generated

This is the final step aggregating predictions using argmax and getting final accuracy using the last function and printing the confusion matrix and accuracy of the final model.

Graphical user interface, text, application

Description automatically generated

The final output:

Graphical user interface, application, Word

Description automatically generated