uOttawa

Professional Master's in Artificial Intelligence

Applied Machine Learning (ELG5255)

Subject: Assignment 1 (Multiclass Classification)

By

Abdelmageed Ahmed Abdelmageed Hassan

Mohamed Sayed Abdelwahab Hussein

Under Supervision

Dr. Murat Simsek

1 - Load the dataset

Steps:

1 - Import the data

2 - Adjust the columns name

3 – Shuffle the data to avoid bias through model Training

Original Data

	14.84	2.221	1
0	14.09	2.699	1
1	13.94	2.259	1
2	14.99	1.355	1
3	14.49	3.586	1
4	14.10	2.750	1

Cleaned Data

	first_feature	fifth_feature	label
0	14.84	2.221	1
1	14.02	3.531	1
2	12.93	5.462	3
3	13.02	3.597	3
4	16.52	4.933	2

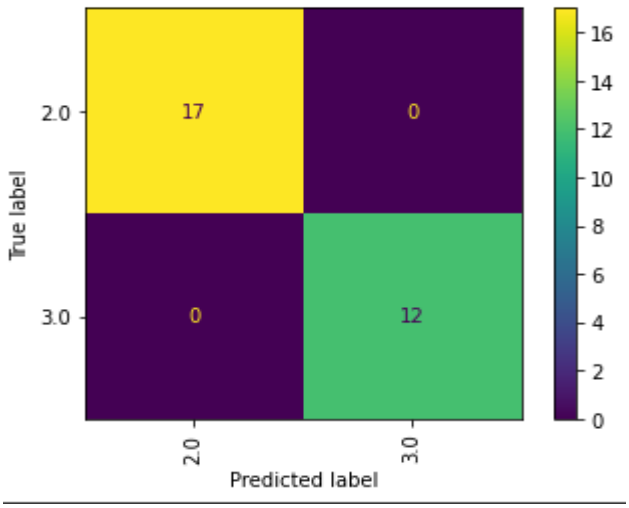
2 - Compare performance of the SVM model and Perceptron

2.1 - **SVM**

Applying SVC gives 100% for testing accuracy and correctly classifies the 41 test data points

testing accuracy : 1.0

	precision	recall	f1-score	support
2.0	1.00	1.00	1.00	17
3.0	1.00	1.00	1.00	12
accuracy			1.00	29
macro avg	1.00	1.00	1.00	29
weighted avg	1.00	1.00	1.00	29

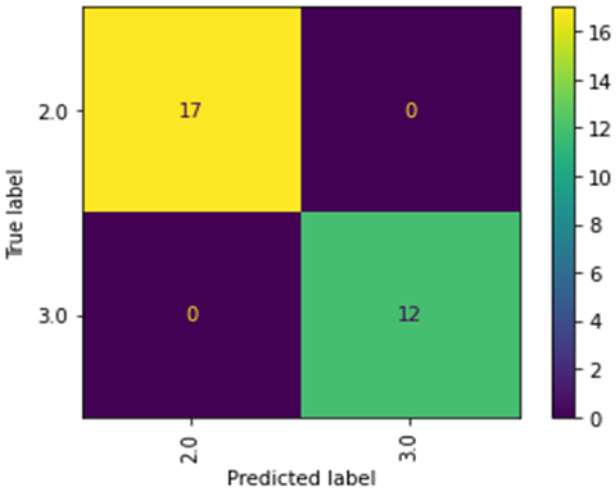


2.2 - Perceptron

Applying Perceptron model also gives 100% for testing accuracy and correctly classifies the 41 test data points

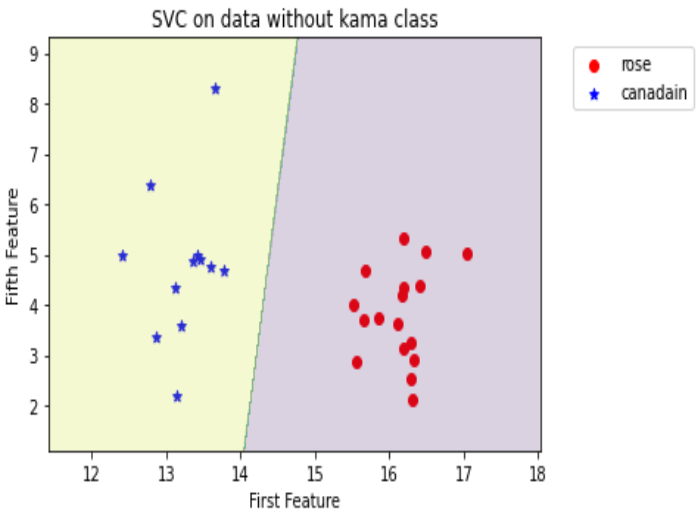
testing accuracy : 1.0

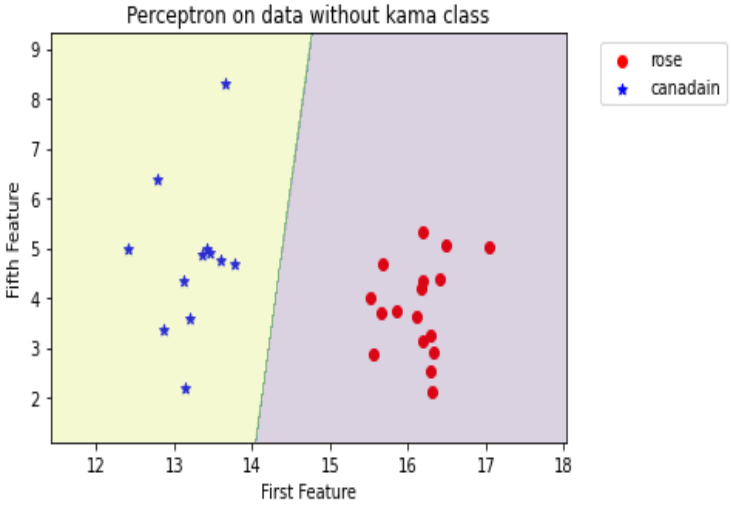
	precision	recall	f1-score	support
2.0	1.00	1.00	1.00	17
3.0	1.00	1.00	1.00	12
accuracy			1.00	29
macro avg	1.00	1.00	1.00	29
weighted avg	1.00	1.00	1.00	29



2.3 - Insights

By removing Kama class, the data become clearly linear separable so estimators can easily Classify the data correctly with 100%





3. Building OvR-Perceptron and OvR-SVM and testing on Seeds testing dataset (which contains 3 classes).

3.1 - Obtain the binarized labels.1 for positive class, -1 for negative class (OvR) :

We made a small function that takes labels and positive class {1,2,3} and return binarized labels using *sklearn.preprocessing.LabelBinarizer*

```
def label_binary(labels , pos_cls):
    ylabels = []
    lb = preprocessing.LabelBinarizier(neg_label  = -1 ,pos_label =1)
    lb.fit([pos_cls])
    y_label_bin = lb.transform(labels)
    for i in range(len(y_label_bin)):
      ylabels.append(y_label_bin[i][0])
    return ylabels
```

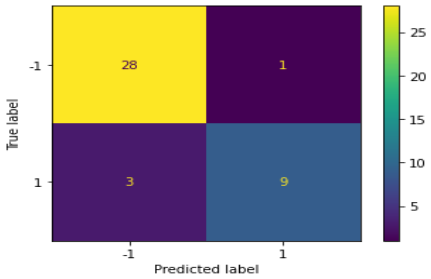
3.2 Obtain the SVM and Perceptron(confusion matrix, accuracy and decision boundary)

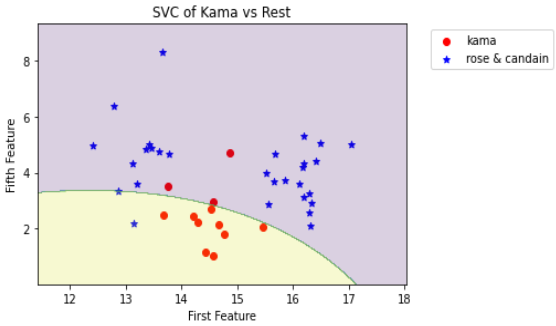
3.2.1 For Kama vs Rest (positive class = 1)

- **SVC**

  SVC achieves 90% testing accuracy by correctly classify 37 of 41 test points and misclassify 4 points

```
testing accuracy :   0.9024390243902439

              precision    recall  f1-score   support

          -1       0.90      0.97      0.93        29
           1       0.90      0.75      0.82        12

    accuracy                           0.90        41
   macro avg       0.90      0.86      0.88        41
weighted avg       0.90      0.90      0.90        41
```

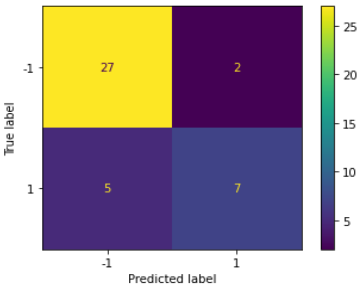


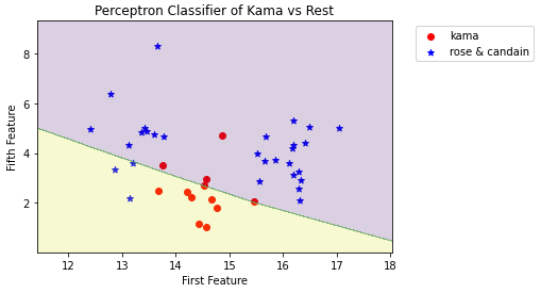


- **Perceptron**

  SVC achieves 90% testing accuracy by correctly classify 37 of 41 test points
  and misclassify 4 points, as we can see from the figure below the data is not linearly separable (the kama class has values between the two classes) and the perceptron with single layer so it can't classify all the data correctly.

```
testing accuracy :   0.8292682926829268

              precision    recall  f1-score   support

          -1       0.84      0.93      0.89        29
           1       0.78      0.58      0.67        12

    accuracy                           0.83        41
   macro avg       0.81      0.76      0.78        41
weighted avg       0.82      0.83      0.82        41
```



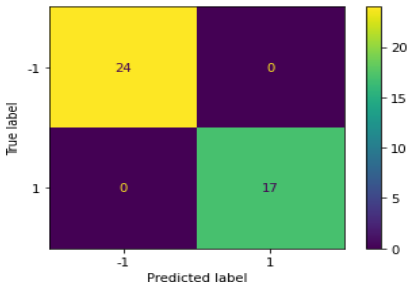


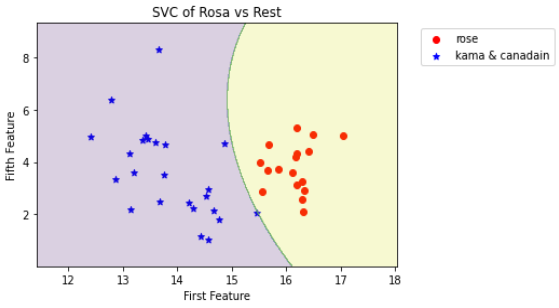
3.2.2 For Rosa vs Rest (positive class = 2)

- **SVC**

  SVC achieves 100% testing accuracy by correctly classify 41 test points correctly, that's because the data is linearly separable so the model can classify them successfully.

```
testing accuracy :   1.0

              precision    recall  f1-score   support

          -1       1.00      1.00      1.00        24
           1       1.00      1.00      1.00        17

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41
```

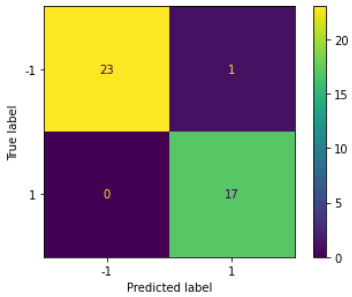


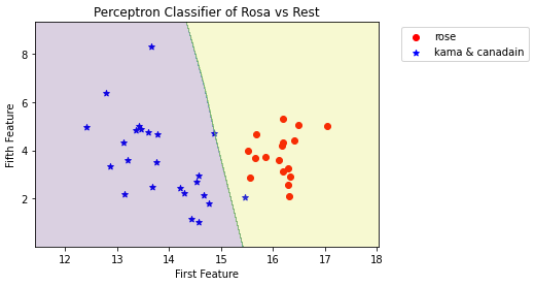


- **Perceptron**

  Also, the perceptron achieves higher accuracy 97%, it only misclassifies one point from rest.

```
testing accuracy :   0.975609756097561

              precision    recall  f1-score   support

          -1       1.00      0.96      0.98        24
           1       0.94      1.00      0.97        17

    accuracy                           0.98        41
   macro avg       0.97      0.98      0.98        41
weighted avg       0.98      0.98      0.98        41
```



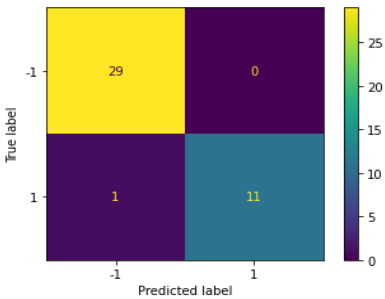


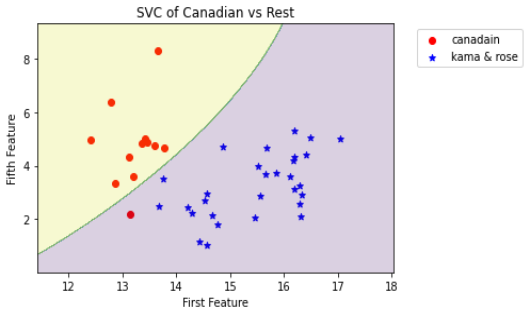
3.2.3 For Rosa vs Rest (positive class = 3)

- **SVC**

  SVC achieves higher accuracy 97% and only misclassify one point from candain class

```
testing accuracy :   0.975609756097561

              precision    recall  f1-score   support

          -1       0.97      1.00      0.98        29
           1       1.00      0.92      0.96        12

    accuracy                           0.98        41
   macro avg       0.98      0.96      0.97        41
weighted avg       0.98      0.98      0.98        41
```

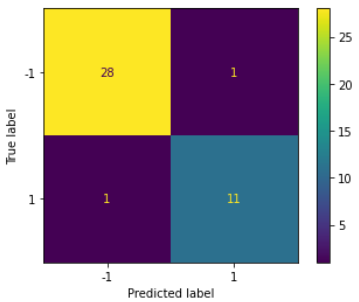


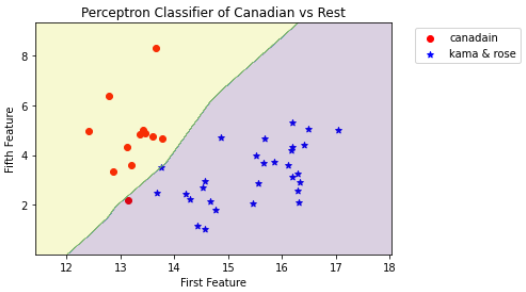


- **Perceptron**

  Perceptron achieves higher accuracy 95% and misclassify one point for each class

```
testing accuracy :   0.9512195121951219

              precision    recall  f1-score   support

          -1       0.97      0.97      0.97        29
           1       0.92      0.92      0.92        12

    accuracy                           0.95        41
   macro avg       0.94      0.94      0.94        41
weighted avg       0.95      0.95      0.95        41
```





Note that: we adapt the hyper parameters of Perceptron to achieve this accuracy (alpha=.0001, n_iter_no_change=1000, max_iter=5000)

4. Use argmax to aggregate confidence scores and obtain the final label and obtain the performance

Argmax technique: we compare the output of argmax prediction with *y_test* label to the actual label to get the label that the model misclassifies them, by adding these points as the fourth label to the copy of *y_label* We can plot the new copied label which has four classes (Kama, Rosa, Canadian and Wrong) by using colour code in plotting we can distinguish the misclassify points.

4.1 - SVM models argmax aggregate

First we Apply the ***Argmax*** function to the confidence score for the Three Models, for SVM we got the confidence score using method **predict_proba** → SVC().fit(X_train,y_train).predict_probe(X_test)

predicts = np.argmax([y_prob_svm1[:,1],y_prob_svm2[:,1],y_prob_svm3[:,1]], axis=0)

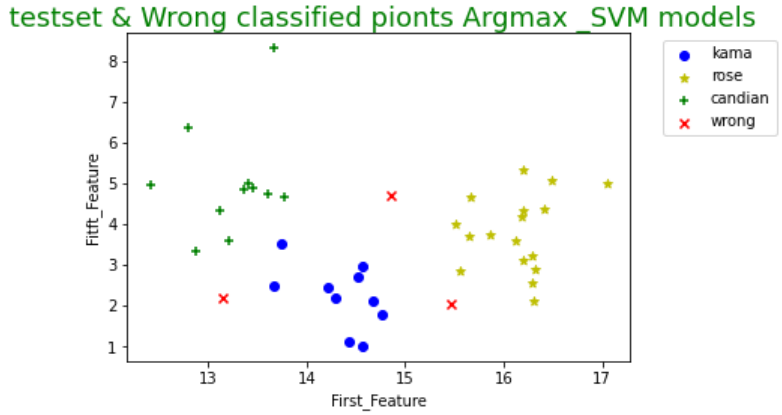
Apply the Agrmax to the confidance score for SVM models

```
predicts = np.argmax([y_prob_svm1[:,1],y_prob_svm2[:,1],y_prob_svm3[:,1]], axis=0)
predictions = predicts + 1
svm_y_label= np.copy(y_test_all)
i=-1
number_of_worng_prediction_svm= []
for input, prediction, label in zip(X_test_all, predictions , y_test_all):
  i+=1
  if prediction != label:
    svm_y_label[i]=4
    data_point= np.stack(input)
    number_of_worng_prediction_svm.append(data_point)

    print(data_point, 'has been classified as ', prediction, 'and should be ', label)

print('\n Number of Wrong Predictions of Argmax for SVM  is : ' ,len(number_of_worng_prediction_svm),'\n')
```

```
[13.15   2.201] has been classified as  1 and should be  3.0
[14.86   4.711] has been classified as  2 and should be  1.0
[15.46  2.04] has been classified as  2 and should be  1.0

 Number of Wrong Predictions of Argmax for SVM  is :  3
```

By applying the Argmax on the confidence score of the 3 model of SVM we get a prediction label with only 3 points misclassify marked red point as shown in the next figure

```
Classification Report:

              precision    recall  f1-score   support

         1.0       0.91      1.00      0.95        10
         2.0       0.89      1.00      0.94        17
         3.0       1.00      1.00      1.00        11
         4.0       0.00      0.00      0.00         3

    accuracy                           0.93        41
   macro avg       0.70      0.75      0.72        41
weighted avg       0.86      0.93      0.89        41


Confusion Matrix:

[[10  0  0  0]
 [ 0 17  0  0]
 [ 0  0 11  0]
 [ 1  2  0  0]]
```



4.2 - Perceptron Models Argmax Aggregate

Unlike the SVM, Perceptron model doesn't have the predict_proba to get the confidence score, so we pass the perceptron model as an argument in ***CalibratedClassifierCV*** model that fit the model and return the *confidence_score* of the predicted output.

```
perc = Perceptron(alpha=.0001,n_iter_no_change=1000, max_iter=5000 )
model  = CalibratedClassifierCV(perc3 , method='isotonic')
```

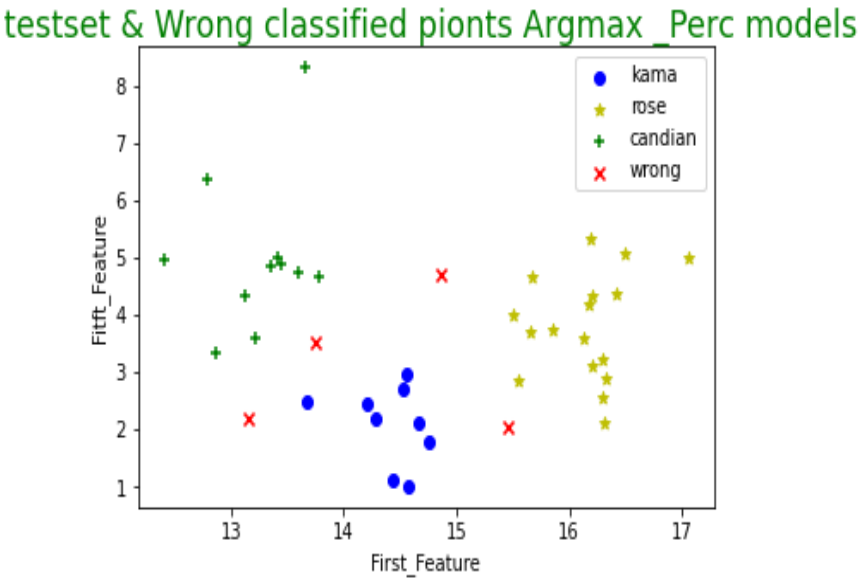
Apply the Agrmax to the confidance score for preceptron models

```
#Applying argmax on the confidance score  of the 3 preceptron models prediction
predicts_2 = np.argmax([y_prob_perc1[:,1],y_prob_perc2[:,1],y_prob_perc3[:,1]], axis=0)
predictions_2 = predicts_2 + 1
#Showing were is the wrong prediction
number_of_wrong_prediction_argmax_prec  = []

svm_y_2= np.copy(y_test_all)
i=-1
for input_2, prediction_2, label_2 in zip(X_test_all, predictions_2, y_test_all):
  i+=1
  if prediction_2 != label_2:
    svm_y_2[i]=4
    data_point_2= np.stack(input_2)
    number_of_wrong_prediction_argmax_prec.append(data_point_2)
    print(data_point_2, 'has been classified as ', prediction_2, 'and should be ', label_2)

print('\n Number of Wrong Predictions of Argmax for SVM  is : ' ,len(number_of_wrong_prediction_argmax_prec),'\n')
```

```
[13.75   3.533] has been classified as  3 and should be  1.0
[13.15   2.201] has been classified as  1 and should be  3.0
[14.86   4.711] has been classified as  2 and should be  1.0
[15.46  2.04] has been classified as  2 and should be  1.0

 Number of Wrong Predictions of Argmax for SVM  is :  4
```

By applying the Argmax on the confidence score of the 3 model of Perceptron we get a prediction label with four points misclassify marked red point as shown in the next figure

```
Classification Report:

              precision    recall  f1-score   support

         1.0       0.90      1.00      0.95         9
         2.0       0.89      1.00      0.94        17
         3.0       0.92      1.00      0.96        11
         4.0       0.00      0.00      0.00         4

    accuracy                           0.90        41
   macro avg       0.68      0.75      0.71        41
weighted avg       0.81      0.90      0.86        41


Confusion Matrix:

[[ 9  0  0  0]
 [ 0 17  0  0]
 [ 0  0 11  0]
 [ 1  2  1  0]]
```



5 - Improve an alternative aggregation strategy instead of existing argmax

from the previous models in step 3 we got the confidence score each model, so our aggregation strategy is to combine the confidence score of each model on predicting the positive class for the three models of SVM /Perceptron in one data frame the same structure for SVM model will be repeated with the Perceptron.

For SVM models we get data frame consists of 3 columns represent the three probabilities for each SVM (OvR) model for positive classes. In the screenshot below, each column name represents the positive class and its values indicates the confidence of its positive class among the other classes (-1), for example for first data point in testing set

model_1 predict the data point as *kame* with confidence *0.011*

model_2 predict the data point as *Rosa* with confidence *1.00*

model_3 predict the data point as *Canadian* with confidence *0.00*

so, we label this data point as *Rosa* or class 2

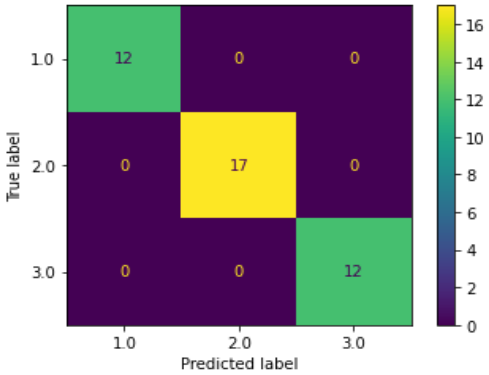
```
df2 = pd.DataFrame(data={'model_1':y_prob_perc1[:,1],'model_2':y_prob_perc2[:,1],
                         'model_3':y_prob_perc3[:,1],
                        'labels':y_test_all})
y_train =df2['labels']
X_train = df2.drop('labels', axis=1)
display(df2.head())
```

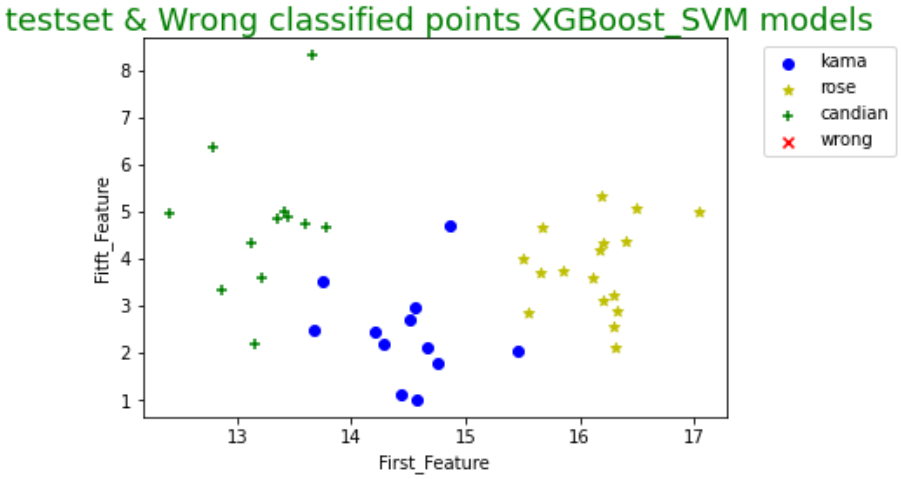
	model_1	model_2	model_3	labels
0	0.011807	1.000000	0.000000	2.0
1	0.392608	1.000000	0.000000	2.0
2	0.883333	0.014355	0.000000	1.0
3	0.358442	0.000000	0.385538	1.0
4	0.082284	0.963592	0.000000	2.0

*So, we use **XGBClassifier** as alternative method to aggregate all 3 models for both SVM and Perceptron and get more accurate model*.

5.1 – *XGBClassifier* aggregate the Three SVM Models

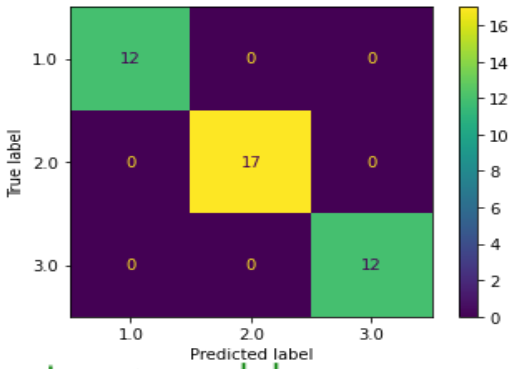
```
Classification Report:

              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        12
         2.0       1.00      1.00      1.00        17
         3.0       1.00      1.00      1.00        12

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41


Confusion Matrix:

[[12  0  0]
 [ 0 17  0]
 [ 0  0 12]]
```

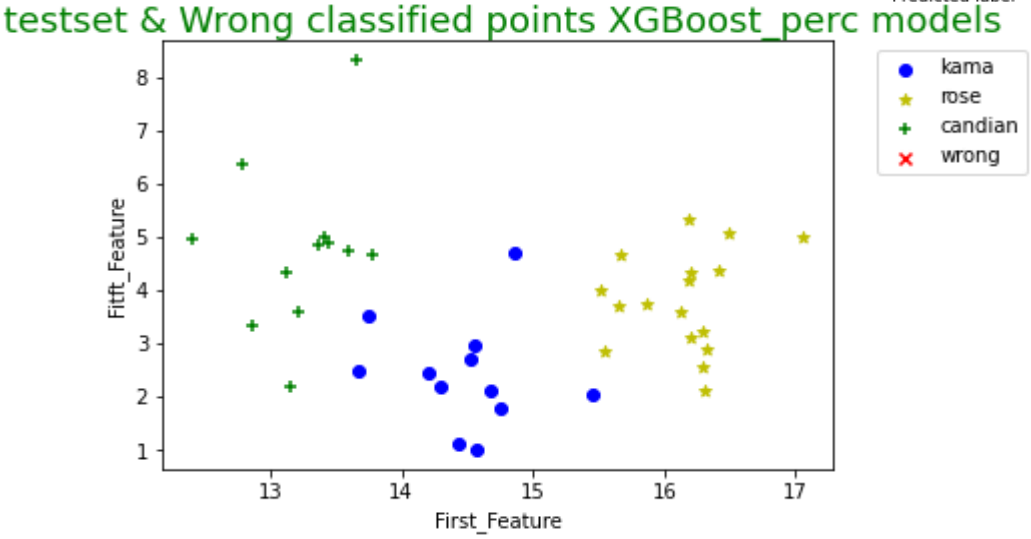




5.2 – *XGBClassifier* aggregate the Three Perceptron Models

```
Classification Report:

              precision    recall  f1-score   support

         1.0       1.00      1.00      1.00        12
         2.0       1.00      1.00      1.00        17
         3.0       1.00      1.00      1.00        12

    accuracy                           1.00        41
   macro avg       1.00      1.00      1.00        41
weighted avg       1.00      1.00      1.00        41


Confusion Matrix:

[[12  0  0]
 [ 0 17  0]
 [ 0  0 12]]
```





Comparing our strategy with argmax method we found that:

- SVM argmax misclassify 3 points out of 41 achieving 95% testing accuracy, while our strategy correctly classifies all 41 points achieving 100% accuracy

- Perceptron argmax misclassify 4 points out of 41 achieving 90% testing accuracy, while our strategy correctly classifies all 41 points achieving 100% accuracy

6 – Conclusion

- We Conclude that SVM works better if the data is linear separable than perceptron, while perceptron's performance increase when we adapt the hyperparameters like increasing number of iterations so it could achieve the same results in case that the data is linearly separable.

- We learnt how to split the multi-class dataset into multiple binary classification problems in order to make model with highest confidence score.

- We learnt How to use the argmax Function of class membership probability or a probability-like score (class index with the largest score) then use it to predict a class.

- We learnt how to use XGBClassifier as alternative method to aggregate all 3 models for both SVM and Perceptron and get more accurate model.