One-Vs-Rest Multiclass Classification

Assignment 1_G4

Ahmed Yousry **Bassel Hamshary**
Mohamed El-Namoury

uOttawa
Faculté de génie
Faculty of Engineering

MAY 22, 2021

UNIVERSITY OF OTTAWA
Ottawa, Canada

Table of Contents

1. Objectives 2
2. Implementation 2
2.1. Loading Data 2
2.2. Binarization 3
2.3. Support Vector Classifier Model (SVC) 4
2.3.1. Training Model 4
2.3.2. Evaluating Model Using Argmax 4
2.3.3. Evaluating Model Using Ensemble algorithm 5
2.4. Logistic Regression Model (LR) 6
2.4.1. Training Model 6
2.4.2. Evaluating Model Using Argmax 6
2.4.3. Evaluating Model Using Ensemble algorithm 7
3. Conclusion 8

Table of Figures

Figure 1: Loading data from Iris dataset. 2
Figure 2: Excluding and renaming data 2
Figure 3: The Binarization function 3
Figure 4: Training of each binarized model using SVC Algorithm 4
Figure 5: The Performance metrics of the SVC model using argmax 4
Figure 6: Boundary selection of class '1'-vs-rest 5
Figure 7: Boundary selection of class '2'-vs-rest 5
Figure 8: Boundary selection of class '3'-vs-rest 5
Figure 9: The Performance metrics of the SVC model using Ensemble modeling 5
Figure 10: Training of each binarized model using Logistic Regression Algorithm 6
Figure 11: The Performance metrics of the LR model using argmax 6
Figure 12: Boundary selection of class '1'-vs-rest 7
Figure 13: Boundary selection of class '2'-vs-rest 7
Figure 14: Boundary selection of class '3'-vs-rest 7
Figure 15: The Performance metrics of the LR model using Ensemble modeling 7

1. Objectives

The purpose of this assignment to do multiclass classification on (“Iris-DATASET”) using 1-vs-ALL technique. Which work as the following:

- Define the target label from labels column.
- Then search for the desired label in the column and put it with value (1).
- Anything else is the label column is labeled to (-1).

At the end we have 3 binarized models for the Iris dataset. Train and evaluate them using support vector classifier SVC and logistic regression LR, and compare the two models.

2. Implementation

2.1. Loading Data

Firstly, we have downloaded the iris dataset from Sklearn library then we printed it. we found out its normalized and cleaned dataset does not need any effort in the data engineering part. Secondly, we found that the data has 4 features (0,1,2,3) and 150 rows. As shown in figure 1. After that, we dropped the first 2 features from the dataset. ("Assignment requirement ") We discovered that the label name (0,1,2) the dataset now is almost clear. but it needs some modification as changing column name in target and feature. So, we used the rename function in python to do this. we have now ("Petal_length", "Petal_width", "y"). As shown in figure 2.

```
iris = load_iris()
```

```
df_data = pd.DataFrame(iris.data)
```

```
df_data
```

	0	1	2	3
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2
...	...	...	...	...
145	6.7	3.0	5.2	2.3
146	6.3	2.5	5.0	1.9
147	6.5	3.0	5.2	2.0
148	6.2	3.4	5.4	2.3
149	5.9	3.0	5.1	1.8

150 rows × 4 columns

Figure 1: Loading data from Iris dataset.

```
df_data.drop(columns=[0,1], inplace= True)
df_data.head()
```

	2	3
0	1.4	0.2
1	1.4	0.2
2	1.3	0.2
3	1.5	0.2
4	1.4	0.2

```
df_target = pd.DataFrame(iris.target)
```

```
df_target
```

	0
0	0
1	0
2	0
3	0
4	0
...	...
145	2
146	2
147	2
148	2

Figure 2: Excluding and renaming data

2.2. Binarization

As shown in figure 3, This is the binarization function we had mentioned earlier in the first slide we did not want to use a ready-made function in python for this we wanted to take control of our code and dive deeper with writing some code. The function as we mentioned earlier takes the desired label and put it with (1) and anything else with (-1).

```
def Bin(df_target, label):
    df_copy=df_target.copy()
    if label == 0:
        df_copy[df_copy['y']==1]=-1
        df_copy[df_copy['y']==2]=-1
        df_copy[df_copy['y']==0]=1
    elif label == 1:
        df_copy[df_copy['y']==2]=-1
        df_copy[df_copy['y']==0]=-1
        df_copy[df_copy['y']==1]=1
    elif label == 2:
        df_copy[df_copy['y']==1]=-1
        df_copy[df_copy['y']==0]=-1
        df_copy[df_copy['y']==2]=1

    return df_copy
```

Figure 3: The Binarization function

2.3. Support Vector Classifier Model (SVC)

2.3.1. Training Model

Now, the data is ready to be trained, we trained our model using SCV classifier. As shown in the below figure, we have 3 different models (Binarized), so we have to train each one on its own. that why we have a loop that iterates 3 times for each SVC model. Then we used the **Argmax function** on the models.

```
models = []  # 3 bIN Model
svm_predictions = []
for i in range(0,3): # loop to No. of models
    df_label = Bin(df_target,i)

    X_train = df_data
    y_train = df_label
    svclassifier = SVC(probability=True)
    svclassifier.fit(X_train, y_train) #SCV Classifier
    y_pred = svclassifier.predict(X_train)
    svm_predictions.append(y_pred)
    models.append(svclassifier)
```

Figure 4: Training of each binarized model using SVC Algorithm

2.3.2. Evaluating Model Using Argmax

Now, we need to evaluate our model performance, therefore, we used the confusion matrix as a metric. We found that the model performs very good in predicting the labels as in the below figure. it missed only 2 points in the second class and 5 points in the third class, this is also shown in the graphs of each class.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.96      0.91      0.93        53
           2       0.90      0.96      0.93        47

    accuracy                           0.95       150
   macro avg       0.95      0.95      0.95       150
weighted avg       0.95      0.95      0.95       150

[[50  0  0]
 [ 0 48  5]
 [ 0  2 45]]
```

```
sn.heatmap(confusion_matrix(svmarg_predictions , df_target), annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0x1bba09d1580>
```

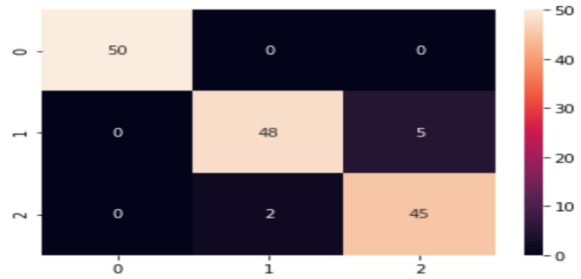

Figure 5: The Performance metrics of the SVC model using argmax

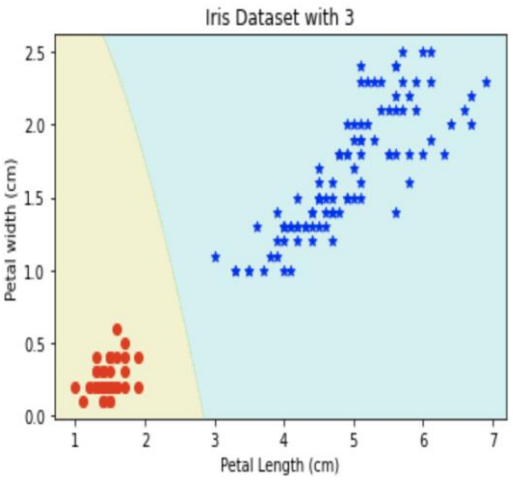

Figure 6: Boundary selection of class '1'-vs-rest

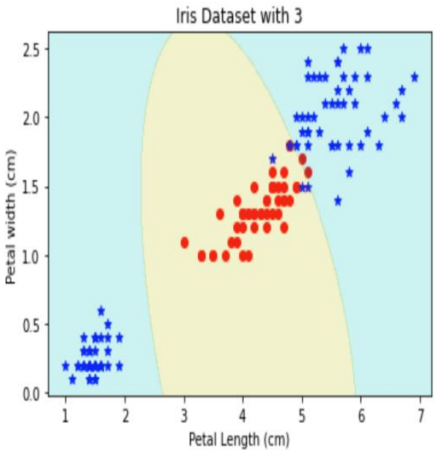

Figure 7: Boundary selection of class '2'-vs-rest

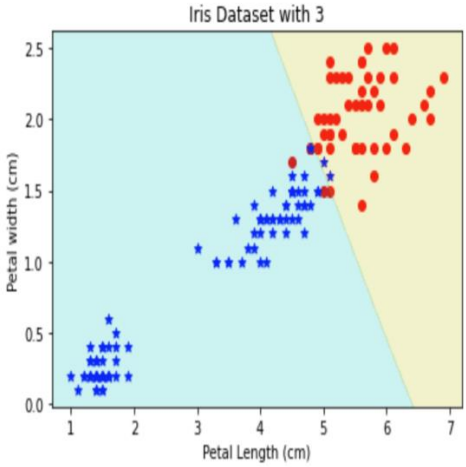

Figure 8: Boundary selection of class '3'-vs-rest

2.3.3. Evaluating Model Using Ensemble algorithm[1]

This method is built on ensembling model in machine learning the last thing we searched for an alternative to measure the performance we found one on towards-science website. The results for SVC is almost the same, as shown below.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.96      0.95        49
           2       0.96      0.94      0.95        51

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150

[[50  0  0]
 [ 0 47  2]
 [ 0  3 48]]
```

```
sn.heatmap(confusion_matrix(voting_svm_result , df_target), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bba14c49d0>
```

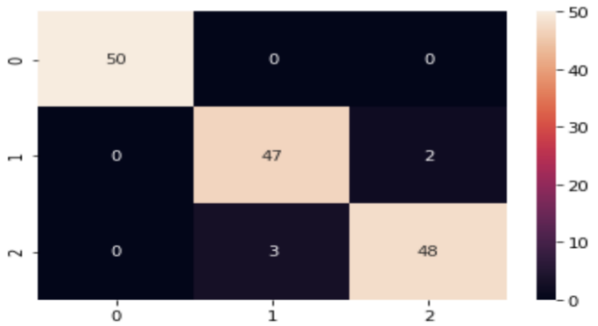

Figure 9: The Performance metrics of the SVC model using Ensemble modeling

[1] Ensemble modeling reference: https://towardsdatascience.com/ensemble-methods-in-machine-learning-what-are-they-and-why-use-them-68ec3f9fef5f

2.4. Logistic Regression Model (LR)

2.4.1. Training Model

We will re-train the model using logistic regression algorithm.

```
LR_predictions = []
LR_models = []
for i in range(0,3):
    df_labelLR = Bin(df_target,i)
    X_train = df_data
    y_trainLR = df_labelLR
    LR = LogisticRegression(random_state=0).fit(X_train, y_trainLR)
    LR_models.append(LR)
    y_predLR = LR.predict(X_train)
    LR_predictions.append(y_predLR)
```

Figure 10: Training of each binarized model using Logistic Regression Algorithm

2.4.2. Evaluating Model Using Argmax

After applying the same steps but this time using logistic regression, we found out the model has very bad performance, especially in the second class where it predicts only one correct point. This means that the problem cannot be implemented by Logistic Regression because it has very poor performance as shown in the below figure.

```
print(classification_report(LRG_predictions , df_target))
print(confusion_matrix(LRG_predictions , df_target))
```

```
              precision    recall  f1-score   support

           0       1.00      0.51      0.68        98
           1       0.02      0.08      0.03        12
           2       0.74      0.93      0.82        40

    accuracy                           0.59       150
   macro avg       0.59      0.51      0.51       150
weighted avg       0.85      0.59      0.66       150

[[50 46  2]
 [ 0  1 11]
 [ 0  3 37]]
```

```
sn.heatmap(confusion_matrix(LRG_predictions , df_target), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bba070ec40>
```

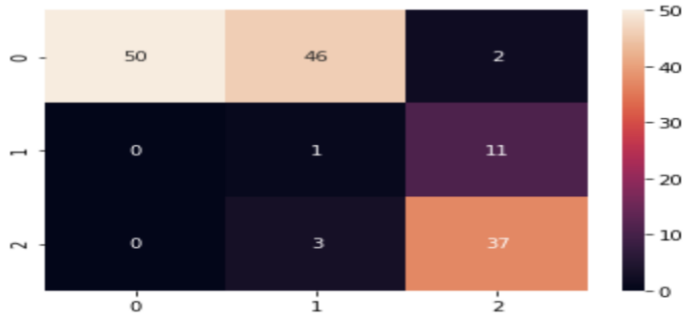

Figure 11: The Performance metrics of the LR model using argmax

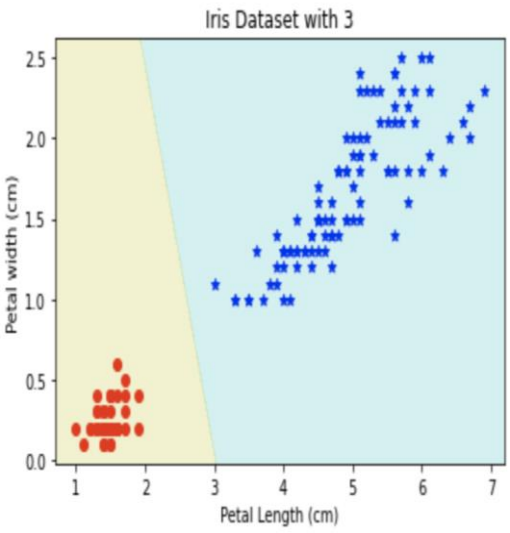

Figure 12: Boundary selection of class '1'-vs-rest

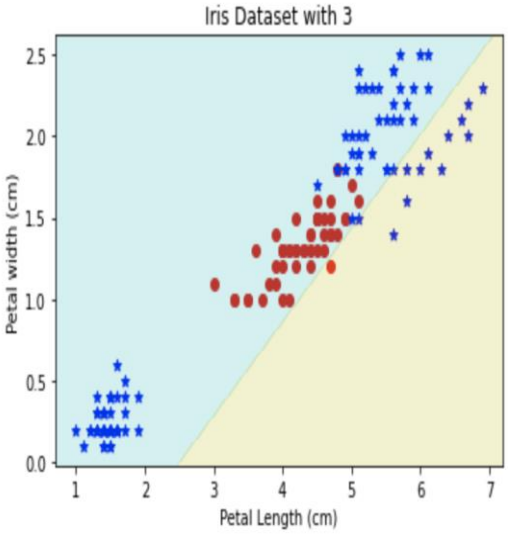

Figure 13: Boundary selection of class '2'-vs-rest

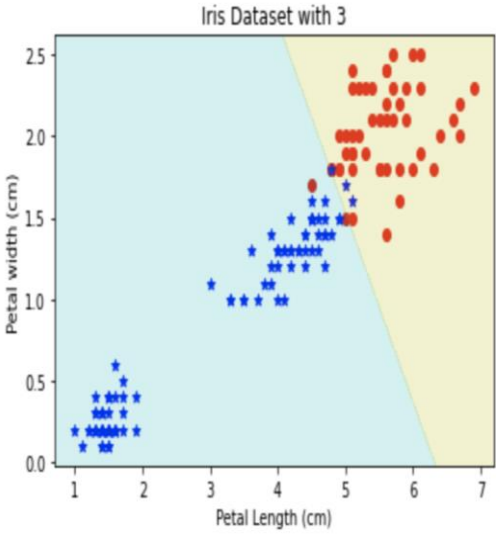

Figure 14: Boundary selection of class '3'-vs-rest

2.4.3. Evaluating Model Using Ensemble algorithm

As shown in the below figure, we noticed that the performance of the evaluation of the model is better when using the ensemble modeling than the Argmax modeling.

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        50
           1       0.94      0.96      0.95        49
           2       0.96      0.94      0.95        51

    accuracy                           0.97       150
   macro avg       0.97      0.97      0.97       150
weighted avg       0.97      0.97      0.97       150

[[50  0  0]
 [ 0 47  2]
 [ 0  3 48]]
```

```
sn.heatmap(confusion_matrix(voting_LR_Result , df_target), annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1bba1409100>
```

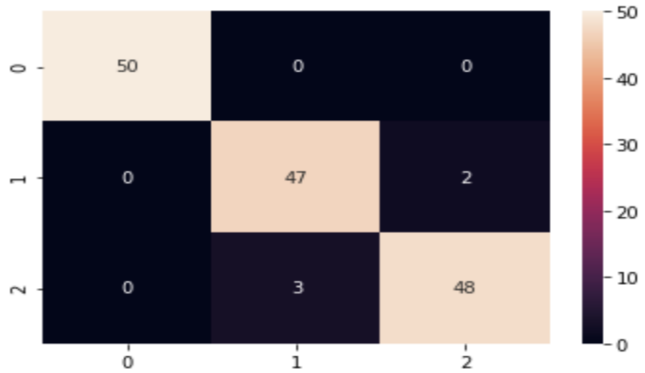

Figure 15: The Performance metrics of the LR model using Ensemble modeling

3. Conclusion

In conclusion, this report could be summed up by loading the data from Iris dataset, after that, the data was binarized to form three separate models for each class following the technique of one-vs-all, then the cleaned data enters two different models (SVC & LR) with two different techniques in evaluating the models (argmax & ensemble). Finally, it was found that using the ensemble modeling with both models (SVC & LR) will give a better performance of accuracy in predicting the class of each record (97%).

Python notebook