SCIKIT TOOLS HOMEWORK

SIFISO LUCOLO DHLAMINI - 410921334

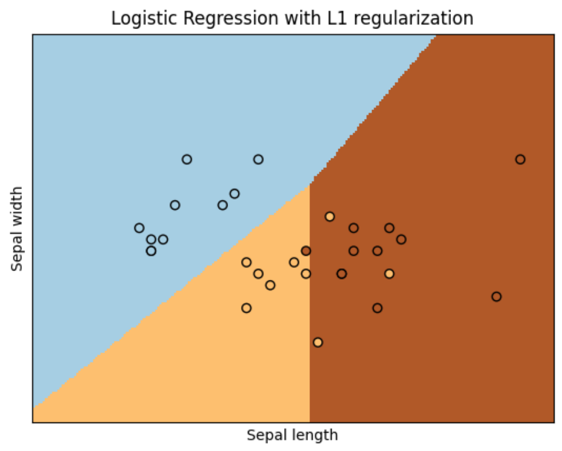
MACHINE LEARNING CSIEM017AB

1. Logistic Regression

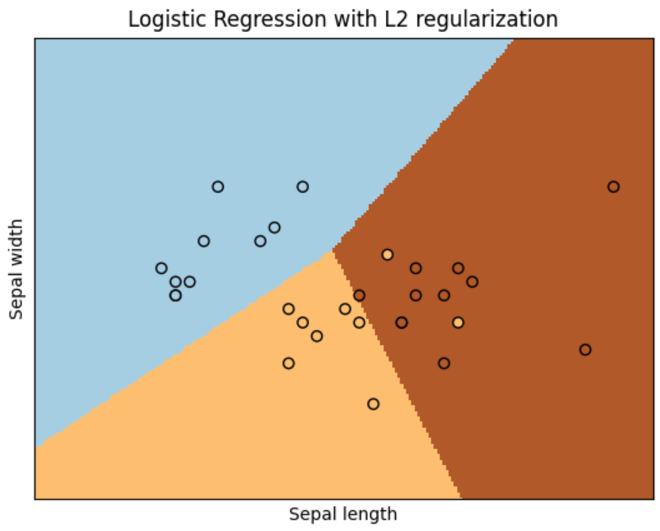
Parameters used in code: Penalty, Solver and C
Values changed in used parameters: Penalty and C

Penalty difference results:

Algorithm: Logistic Regression with L1 regularization
Parameters: {'C': 0.1, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l1', 'random_state': None, 'solver': 'saga', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}
Accuracy: 0.8666666666666666



Algorithm: Logistic Regression with L2 regularization
Parameters: {'C': 0.1, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': None, 'solver': 'saga', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}
Accuracy: 0.9333333333333333



Results and observation:

we can observe that Logistic Regression with L2 regularization outperforms Logistic Regression with L1 regularization. The accuracy of the model with L2 regularization is higher (0.933) than the accuracy of the model with L1 regularization (0.867).

This indicates that L2 regularization is better suited for this particular problem and dataset. However, it's worth noting that the choice of regularization type and parameter values may vary depending on the specific problem and dataset, and it's important to experiment with different options to find the best model for the task at hand.

Same parameters with smaller C:

Algorithm: Logistic Regression with L1 regularization
Parameters: {'C': 0.01, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l1', 'random_state': None, 'solver': 'saga', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}
Accuracy: 0.3333333333333333

Algorithm: Logistic Regression with L2 regularization
Parameters: {'C': 0.01, 'class_weight': None, 'dual': False, 'fit_intercept': True, 'intercept_scaling': 1, 'l1_ratio': None, 'max_iter': 100, 'multi_class': 'auto', 'n_jobs': None, 'penalty': 'l2', 'random_state': None, 'solver': 'saga', 'tol': 0.0001, 'verbose': 0, 'warm_start': False}
Accuracy: 0.9

Results and observation:

For both L1 and L2 regularization, the value of C is set to 0.01, which means a higher regularization strength. With this setting, the L1 regularization model is performing poorly, with an accuracy of only 0.333. This could be because L1 regularization tends to produce sparse solutions by setting some coefficients to zero, which could result in underfitting when the regularization strength is too high.

On the other hand, the L2 regularization model has an accuracy of 0.9, which is a significant improvement over the L1 regularization model. This suggests that L2 regularization is more effective in preventing overfitting with a higher regularization strength.

In summary, when the regularization strength is set to a high value, L2 regularization tends to perform better than L1 regularization. However, it's worth noting that the optimal value of C depends on the specific problem and dataset, and it's usually determined through cross-validation.

Conclusion:

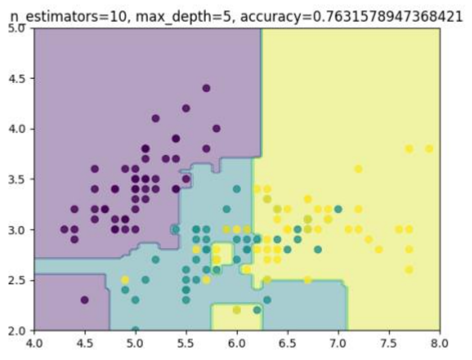
Based on the results, it appears that the logistic regression model with L2 regularization and C=0.1 performed the best with an accuracy of 0.933. The model with L1 regularization and C=0.1 also performed reasonably well with an accuracy of 0.867. However, the models with C=0.01 performed poorly, with the L1 model having an accuracy of 0.333 and the L2 model having an accuracy of 0.9. This suggests that the choice of C value is important for the performance of the model, and that L2 regularization may be more effective than L1 regularization in this particular dataset.

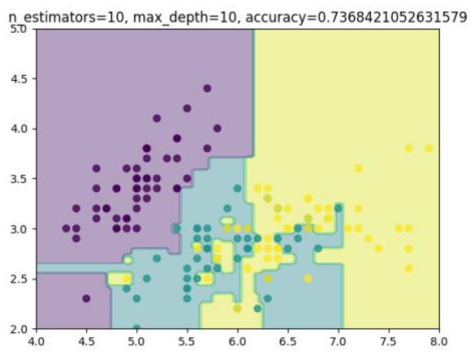
2. Random Forest Classifier

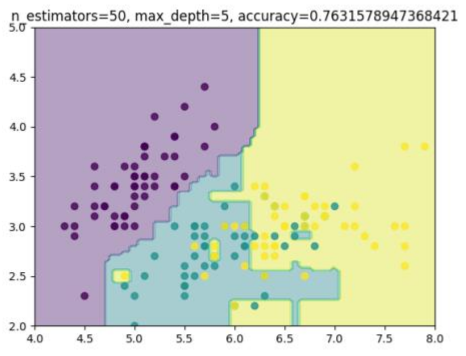
Values changed in used parameters: n_estimators, max_depth

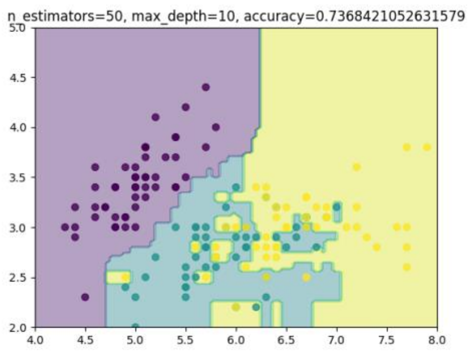
Results:

n_estimators	max_depth	accuracy
10	5	0.7631578947368421
10	10	0.7368421052631579
50	5	0.7631578947368421
50	10	0.7368421052631579









Observation and Conclusion:

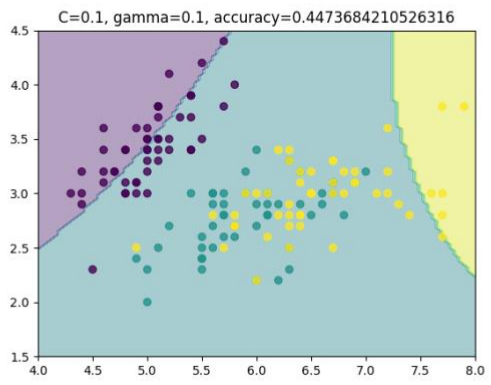
Based on the results, we can observe that increasing the number of estimators does not necessarily lead to an increase in accuracy. For both max_depth values tested, the accuracy remains the same for n_estimators=10 and n_estimators=50. Additionally, increasing the max_depth from 5 to 10 seems to slightly decrease the accuracy, although the difference is not significant.

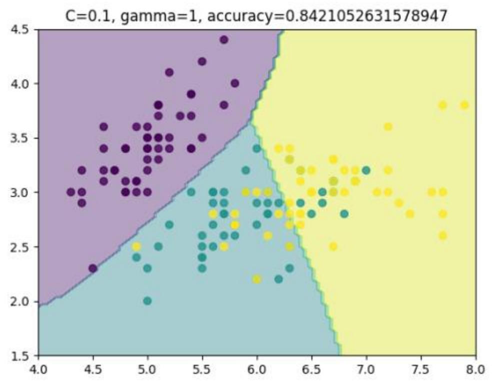
Therefore, we can conclude that the optimal combination of hyperparameters for this particular problem may not necessarily involve high values for n_estimators and max_depth. It is important to carefully select the hyperparameters and consider the trade-off between accuracy and computational cost.

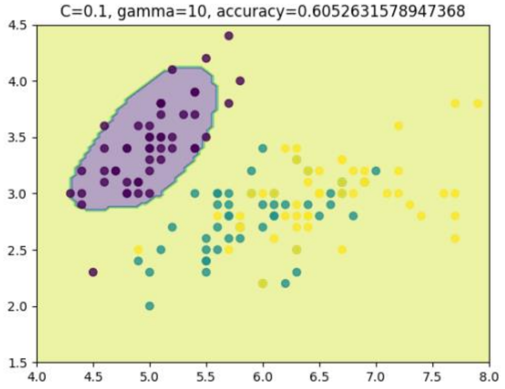
3. Standard Vector Classification

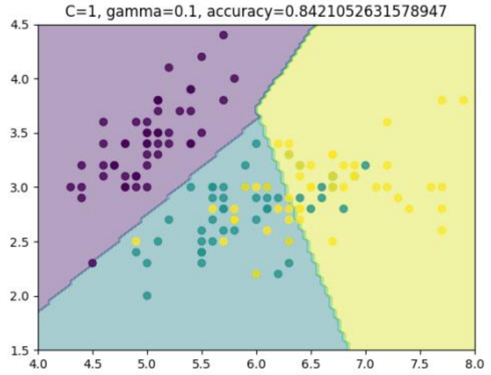
Results:

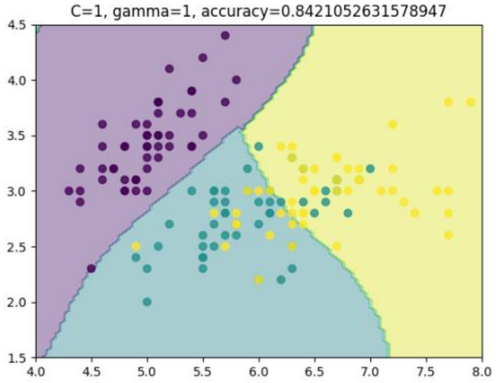
C	gamma	accuracy
0.1	0.1	0.4473684210526316
0.1	1	0.8421052631578947
0.1	10	0.6052631578947368
1	0.1	0.8421052631578947
1	1	0.8421052631578947
1	10	0.8157894736842105
10	0.1	0.7631578947368421
10	1	0.8157894736842105
10	10	0.8157894736842105

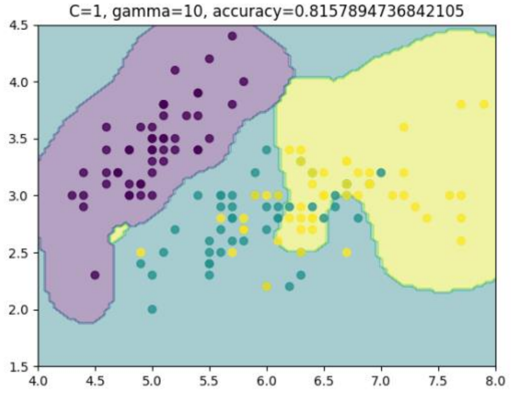


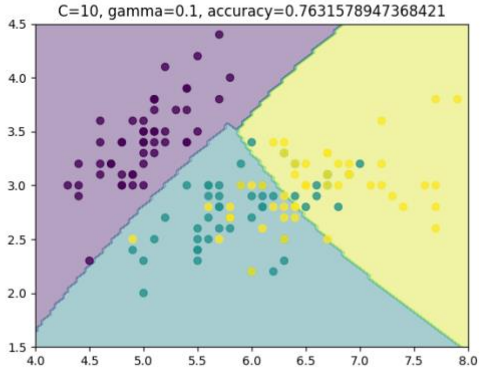


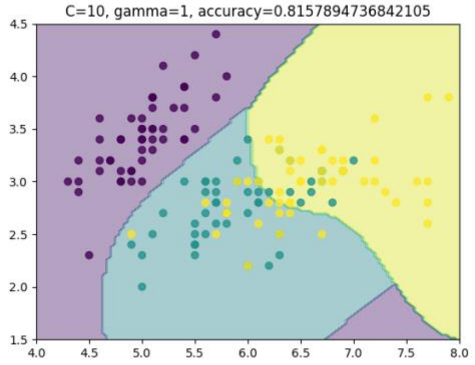

C=0.1, gamma=10, accuracy=0.6052631578947368

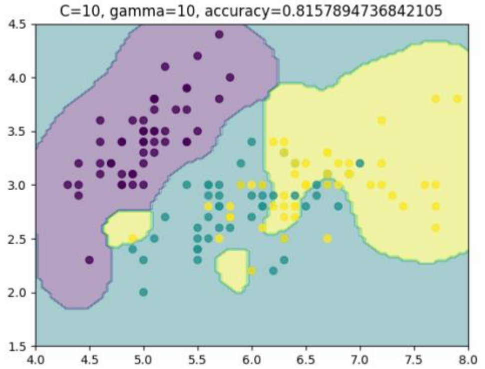

C=1, gamma=0.1, accuracy=0.8421052631578947


C=1, gamma=1, accuracy=0.8421052631578947


C=1, gamma=10, accuracy=0.8157894736842105


C=10, gamma=0.1, accuracy=0.7631578947368421


C=10, gamma=1, accuracy=0.8157894736842105


C=10, gamma=10, accuracy=0.8157894736842105

Observation and Conclusion:

From the results, we can observe that increasing the value of C generally leads to an increase in accuracy. This is especially true when C is increased from 0.1 to 1 or 10. However, increasing gamma does not necessarily lead to an increase in accuracy, and the results seem to be more varied for different gamma values.

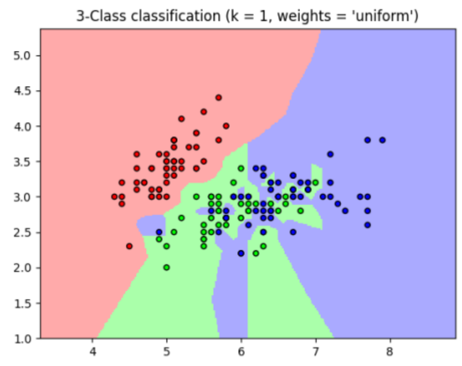
Specifically, we can see that for C=0.1, the accuracy is relatively low for all gamma values tested, with the highest accuracy being 0.605 for gamma=10. For C=1, the accuracy is consistently high across all gamma values tested, with the highest accuracy being 0.842 for gamma=0.1 and gamma=1. For C=10, the accuracy is also relatively high, with the highest accuracy being 0.816 for gamma=10.

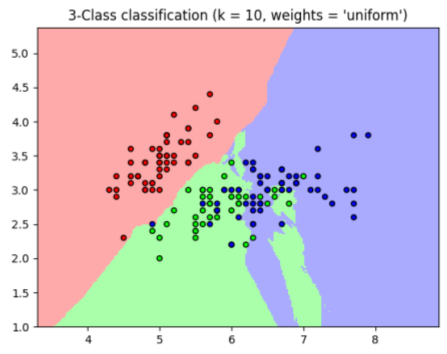
In conclusion, the choice of C seems to be more important for determining the accuracy of the SVM classifier than the choice of gamma. Increasing the value of C generally leads to an increase in accuracy, but the effect of gamma on accuracy seems to be less consistent.

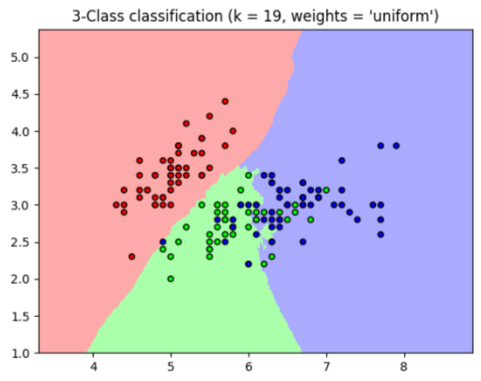
4. KNeighbors Classifier

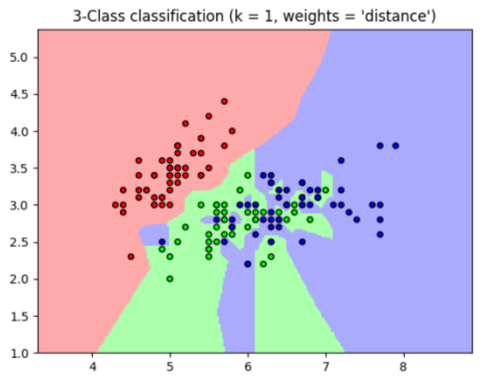
Results:

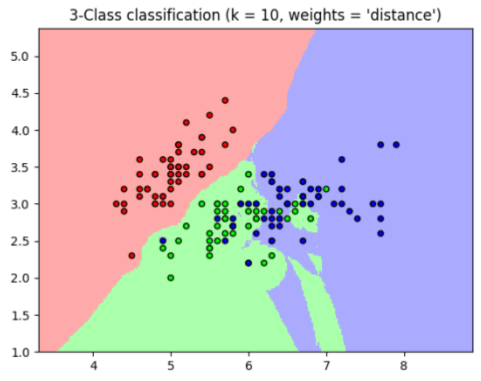
n_neighbors = 1, weights = 'uniform', accuracy = 0.920
n_neighbors = 1, weights = 'distance', accuracy = 0.920
n_neighbors = 10, weights = 'uniform', accuracy = 0.847
n_neighbors = 10, weights = 'distance', accuracy = 0.927
n_neighbors = 19, weights = 'uniform', accuracy = 0.793
n_neighbors = 19, weights = 'distance', accuracy = 0.927

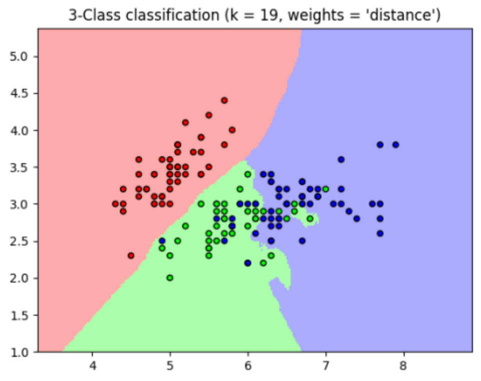












Observation and Conclusion:

From the results, we can observe the following:

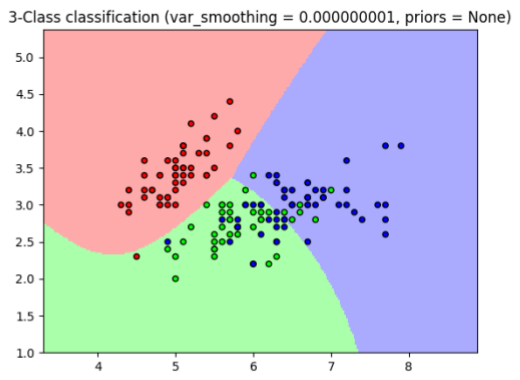
- The accuracy of the K-nearest neighbor classifier is affected by the number of neighbors (n_neighbors) and the weight function used (weights).
- When using a uniform weight function, the accuracy decreases as the number of neighbors increases. This means that using more neighbors doesn't necessarily improve the performance of the classifier.
- On the other hand, when using a distance weight function, the accuracy improves as the number of neighbors increases, which means that using more neighbors can improve the performance of the classifier.
- The highest accuracy was obtained when using a distance weight function and 10 or 19 neighbors, with an accuracy of 0.927.

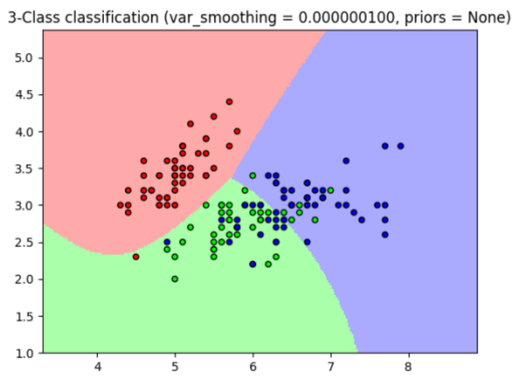
In conclusion, the K-nearest neighbor classifier can be a good choice for classification tasks, but the performance can be affected by the choice of hyperparameters, such as the number of neighbors and the weight function used. It is important to experiment with different hyperparameter values and choose the ones that give the best performance.

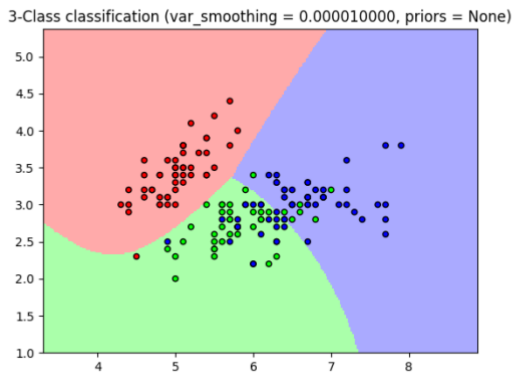
5. Gaussian NB and Multinomial NB

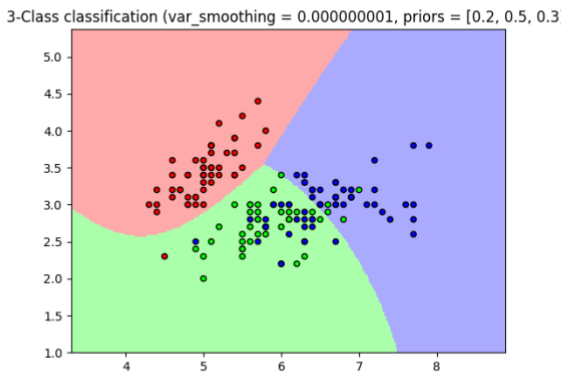
Results:

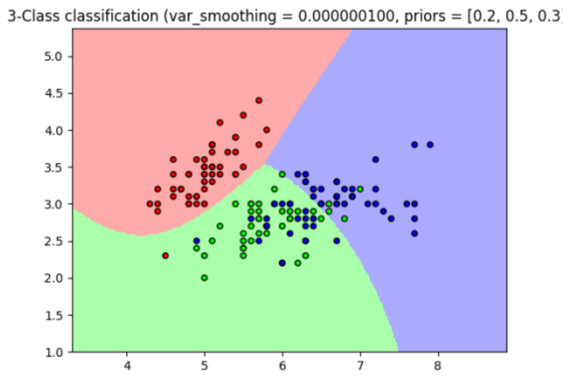
var_smoothing = 0.000000001, priors = None, accuracy = 0.780
var_smoothing = 0.000000100, priors = None, accuracy = 0.780
var_smoothing = 0.000010000, priors = None, accuracy = 0.780
var_smoothing = 0.000000001, priors = [0.2, 0.5, 0.3], accuracy = 0.793
var_smoothing = 0.000000100, priors = [0.2, 0.5, 0.3], accuracy = 0.793
var_smoothing = 0.000010000, priors = [0.2, 0.5, 0.3], accuracy = 0.793

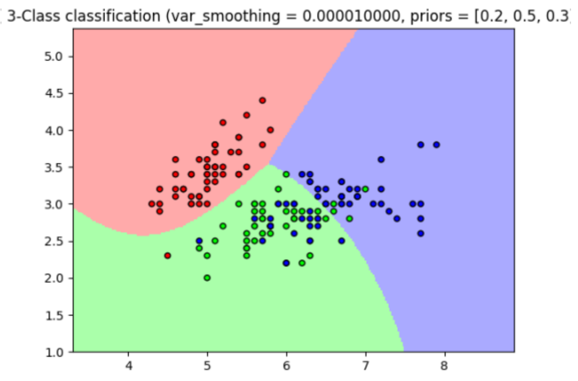












Observation and Conclusion:

Based on the results, it seems that changing the value of var_smoothing did not have a significant impact on the accuracy of the model. However, specifying prior probabilities did slightly improve the accuracy.

It's also worth noting that the accuracy achieved by GaussianNB is lower than the accuracy achieved by KNN in our previous analysis. This could be due to the fact that KNN is a non-parametric model, whereas GaussianNB assumes a specific probability distribution for the data. In this case, it seems that the Gaussian distribution may not be the best fit for the Iris dataset.