Some Important Question regarding classical methods

Q1: Explain why you had to split the dataset into train and test sets?

Answer:

Whole Dataset = Training Data + Testing Data

- **Training data** includes the sample of data with their corresponding outputs (expected). In other words, Labelled data which is used to fit/construct the model for learning and to measure model performance (Ex: Accuracy score, Sensitivity, F-score etc.). Clearly, training data provides biased accuracy score as labels are already provided.
- **Testing Data** contains data sample of whole dataset (without expected results) on which fully trained model performance is assessed and validated. Clearly, it provides unbiased estimation of the accuracy scores as model has never seen this data.
- Once a classifier is trained with training data, it produces a function which can be used for mapping test data with their corresponding outputs. So, in order to allow an algorithm to make new predictions based on new data, splitting the whole dataset into training and testing data is considered as a good practice.
- Same training data or even subset of training data should not be used for testing purpose because it will lead to overfitting as model has already learned the training data and will perform extremely well on testing data (if testing data = training data/subset of training data) however will give poor performance while dealing with new data. Both training and testing data should never overlap with each other.

To summarize, main idea behind splitting is, it is done to ensure that the model actually "learns" how to predict an answer and not just "memorizing" the training data results.

Helpful Links:

- https://en.wikipedia.org/wiki/Training,_validation,_and_test_sets
- https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7
- https://developers.google.com/machine-learning/crash-course/training-and-test-sets/splitting-data

Q2: Explain why when finding the best parameters for KNN you didn't evaluate directly on the test set and had to use a validation test?

Answer:

Validation Data is the data sample which fits to the trained model in order to tune the hyperparameters of the classifier (in this case KNN) to achieve maximum performance. It provides unbiased estimation of model's performance while tuning hyperparameters of a classifier. Using Validation data, model make certain predictions on the data and these predictions are used to measure classification accuracy or classification error.

Below mentioned is the reason why I used validation set for finding the best parameters instead of using directly test data.

1) The performance of KNN classifier is evaluated by tuning different hyper-parameters using validation data. All different combinations of hyper-parameters are then evaluated.
2) KNN classifier with best hyper-parameters giving least amount of error and maximum accuracy is then selected.

3) In order to fine tune the hyper-parameters, I had to run the model multiple times on validation data and there's a possibility of overfitting (model might pick noise or random data fluctuations and learn them as concepts, which will degrade the performance of the model). So, in order to confirm the performance of KNN classifier and avoid overfitting on test data, another data sample is used which is known as "test data".

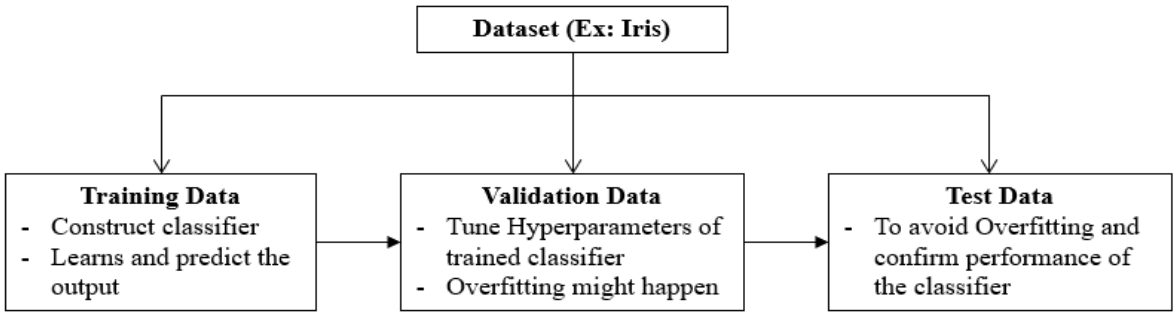


Figure 1

Helpful Links:

- https://machinelearningmastery.com/difference-test-validation-datasets/
- https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7
- https://www.quora.com/Why-do-we-have-separation-of-data-into-training-held-out-and-test-data

Q3: What was the effect of changing k for KNN? Was the accuracy always affecting the same way with an increase of k? Why do you think this happened?

Answer:

- In KNN**,** number of neighbors (K) is considered as a controlling hyperparameter which affects model performance and prediction. Just to make things clear, there exists no optimal value of K, it depends on provided dataset. Depending on the value of K, you'll end up with different classification.
- **Pattern:** From the figure 2.1, We have two values of K (K = 25 and 30) which provided maximum accuracy of 96.6667%. Lowest Accuracy has been obtained for K=1 and for all other K values (K = 5, 10 ,15, 20, 35) accuracy of 93.333% has been observed.

    1) When K =1, each data point will take only 1 nearest neighbor to predict the class which results in low bias but high variance i.e. Overfitting.
    2) As K increases, it starts to consider average of nearest data points for each prediction. This means model become resilient to outliers and will have smooth/stable decision boundary (bias starts increasing, whereas variance starts decreasing with increase in K) and ultimately providing better accuracy than K =1 i.e. 93.33%. In other words, increment in K leads to classifier generalization capacity.

3) At K =25 and K =30, classifier gave maximum accuracy.
4) At K =35, decision boundary starts becoming transparent which negatively impacts the classification performance of the classifier, this leads to high bias between expected and original output along with low variance and in turn accuracy will start to decrease again. i.e. from 96.67% to 93.33%. Misclassification has been observed from confusion matrix of K =35. (Refer to figure 2.2)

- When both K =25 and 30 tested with test data, it provided 100 % accuracy. So, in order to pick one K value following things has been to be kept in mind:
    1) K = 30 is an even number and with this K, classifier may face a situation when 15 of the nearest neighbors belong to setosa class and remaining 15 to versicolor or each 10 neighbors belong to three classes, in this tie situation classifier will not be able to do classification properly.
    2) K =25 has less bias as compared to K =30. Even though K =25 has high variance than 30 that means there exists more variability with 25, still K =25 will be able to classify the datapoints more accurately because of less bias.

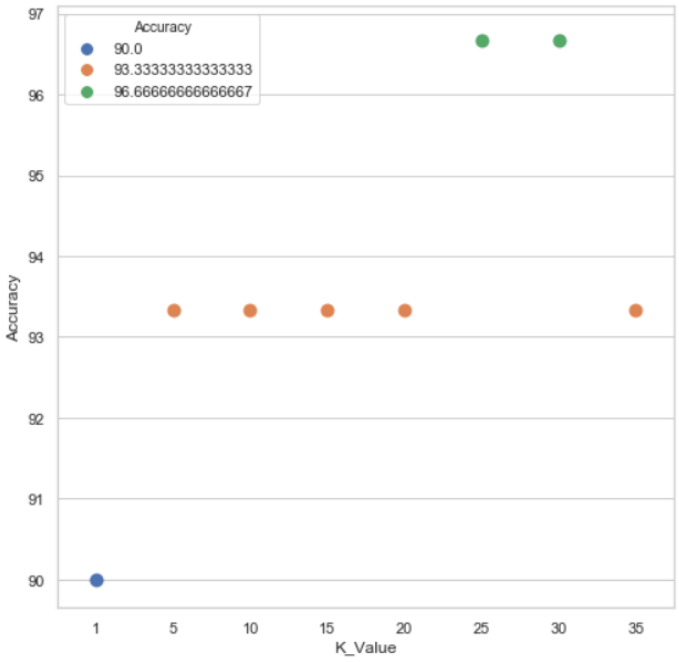


Figure 2.1

```
For K =  35

Confusion Matrix:
 [[10  0  0]
 [ 0  9  0]
 [ 0  1 10]]


Classification Report:
              precision    recall  f1-score   support

      setosa       1.00      1.00      1.00        10
  versicolor       0.90      1.00      0.95         9
   virginica       1.00      0.91      0.95        11

    accuracy                           0.97        30
   macro avg       0.97      0.97      0.97        30
weighted avg       0.97      0.97      0.97        30


Time taken by K = 35:  0.03195691108703613

Accuracy of KNN Model:
 96.66666666666667 %
------------------------------------------------------------------
```

Figure 2.2

Helpful Links:

- https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
- https://www.datacamp.com/community/tutorials/k-nearest-neighbor-classification-scikit-learn
- https://en.wikipedia.org/wiki/Bias%E2%80%93variance_tradeoff
- https://www.quora.com/Why-does-the-variance-decreases-in-KNN-algorithm-when-we-increase-the-K

Q4: What was the relative effect of changing the max depths for decision tree and random forests? Explain the reason for this.

Answer:

- Maximum "number of depths" criterion is an important factor in evaluating the classifier's performance.
- As per my understanding, "*the deeper you allow your tree to grow, it will have more splits which eventually captures more information about the data. This might lead to overfitting as model will perfectly fit over the training data and will not be able to generalize itself over the test data.*"
- Same has been proved for below mentioned classifiers.
- For **Decision Tree Classifier**, I achieved the max. accuracy of 93.33% for depth =3 (smallest) and with highest speed of convergence. (Refer to figure 3.1).

```
For depth = 3
Mean Accuracy is: 0.933
Time:  0.01795196533203125   <-- Lowest
--------------------------------------------------------------------------------


For depth = 5
Mean Accuracy is: 0.917
Time:  0.03690147399902344
--------------------------------------------------------------------------------


For depth = 10
Mean Accuracy is: 0.925
Time:  0.05684971809387207
--------------------------------------------------------------------------------


For depth = None
Mean Accuracy is: 0.925
Time:  0.07679510116577148
--------------------------------------------------------------------------------
```

Figure 3.1 (Decision Tree Output)

- For **Random Forest Classifier**, if you'll refer figure (3.2), it is clearly visible that in all the cases, maximum accuracy is obtained when classifier had least number of depths (n_depths = 3 and number of trees =50). As depth increases to 150 and 200, bias amount will increment because of increase in random samples, classifier will start performing misclassification and will eventually lose the generalization capability.

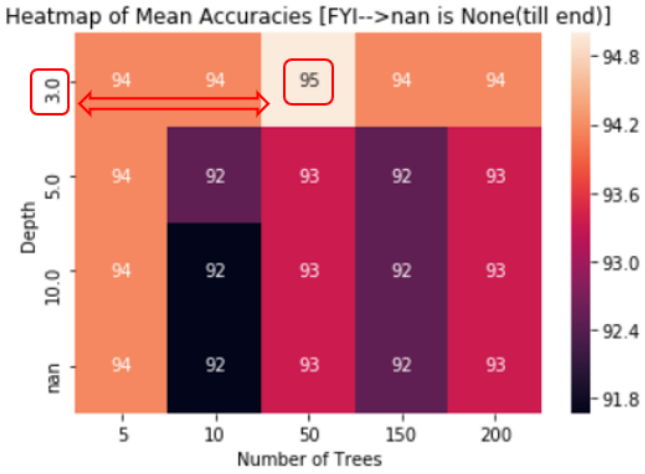
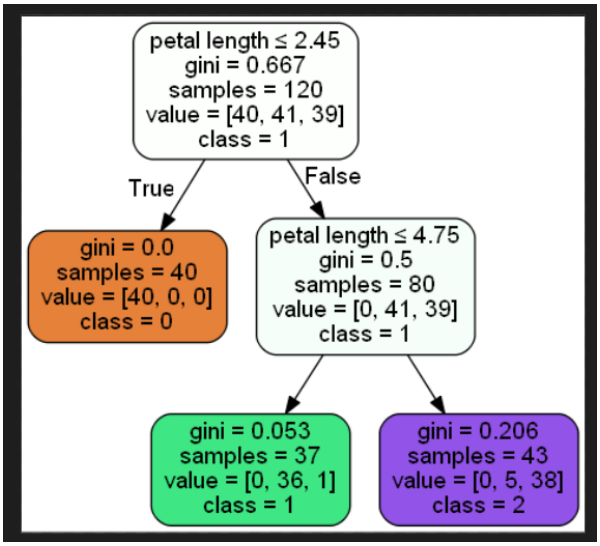


Figure 3.2 (Random Forest Classifier Heat Map)

Some Important Question regarding classical methods

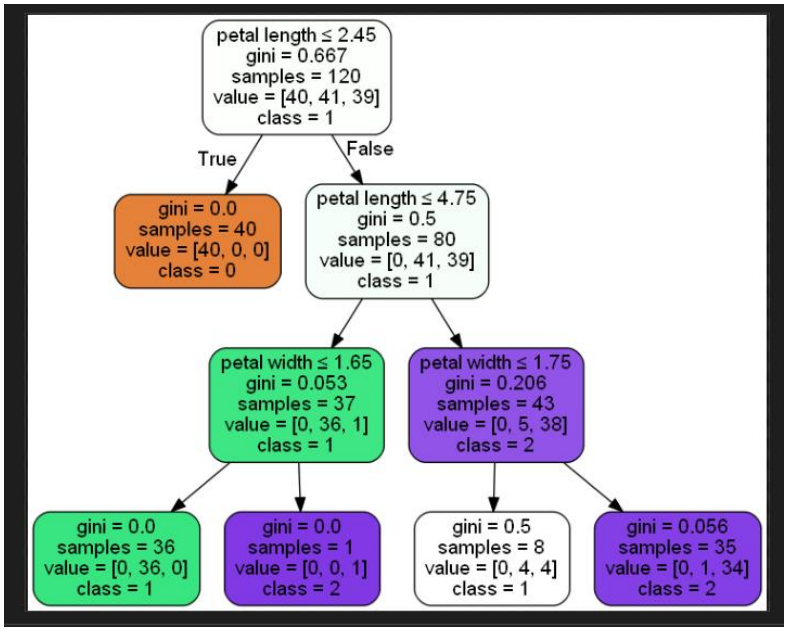
Decision Tree Classifier: In-depth Analysis using visualizations:

- At depth = 1,2, Decision Tree classifier produces a tree with one decision function (**Petal Length**) which is not enough for the classifier to perform classification and end up with an accuracy of 90% on Validation Data. In another words, classifier doesn't have enough data to learn and predict the results i.e. Underfitting has occurred.



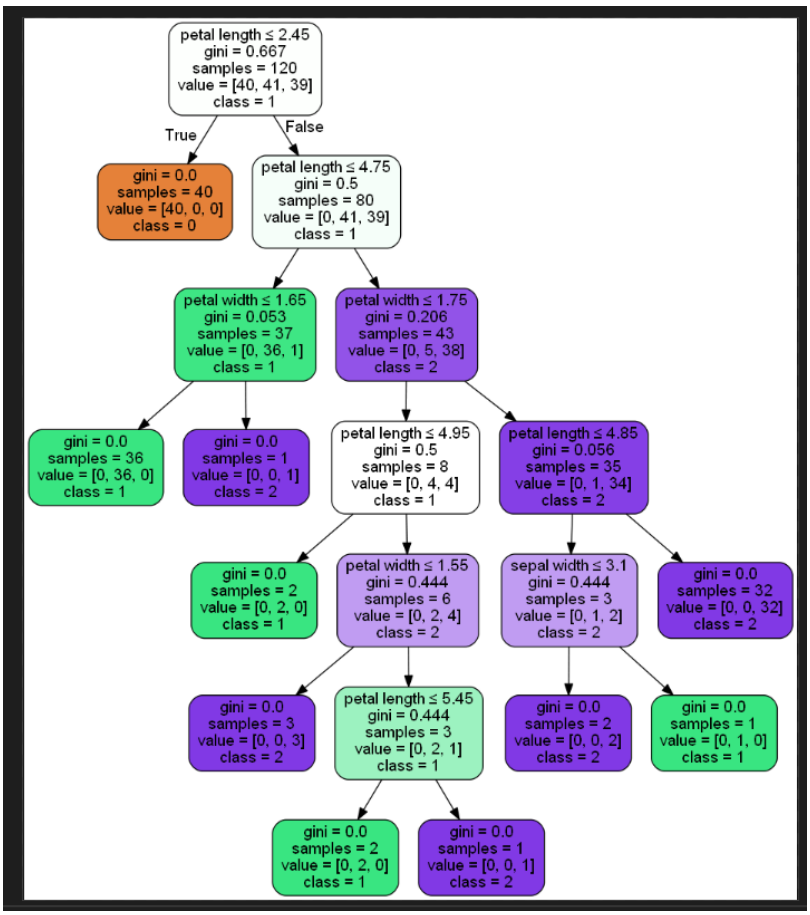
- At depth = 3, Decision Tree Classifier takes into account **petal length** and **petal width** decision functions which were considered as best features to perform the classification and obtained an accuracy of 93.33% on validation data (though there will be misclassification as there was minor overlapping between Versicolor and Virginica) [Refer to Pair plots comments]

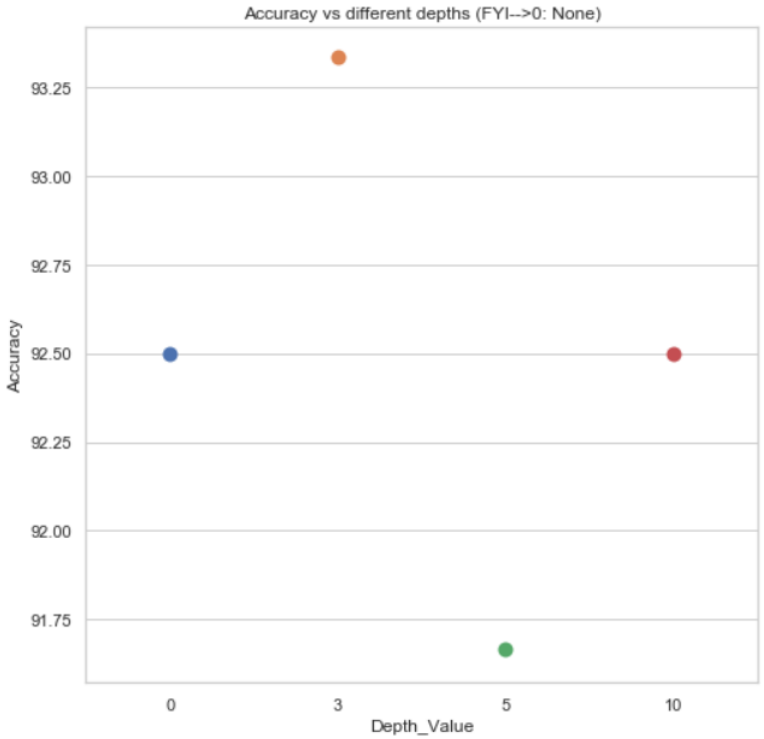
Some Important Question regarding classical methods



- At depth= [10, None], classifier would go into too much depth creating multiple decision functions, that means classifier would narrow down the validation data and will make too much accurate predictions/decision boundary will be extremely smooth or almost transparent, so when tested on an unseen data, classifier will give 100 percent accuracy which is the case of Overfitting.
  Accuracy with depth on Validation Data: [10, None] = 92.5%.

Some Important Question regarding classical methods





[Overall, in all three tree-based classifiers, maximum accuracy of model is obtained at least number of depths i.e. 3 and as you increase the depth, model accuracy starts to decline and starts becoming computationally expensive.]

Q5: Comment on the effect of the number of estimators for Gradient Tree Boosting and what was the relative effect performance of gradient boosting compared with random forest. Explain the reason for this.

Answer:

- As per my understanding, "*higher is the number of estimators, more will be the trees and as the number of tress increases it helps the model to learn the data in a better manner as noise effect gets reduced with the aggregation of trees. On the other hand, having too many trees slows down the model performance also.*"

- **For Random Forest Classifier:** As the number of trees starts to increase, model accuracy also increased but it reached a point where it provided max accuracy (Number of Trees = 50) and if we go beyond that model accuracy starts declining again.

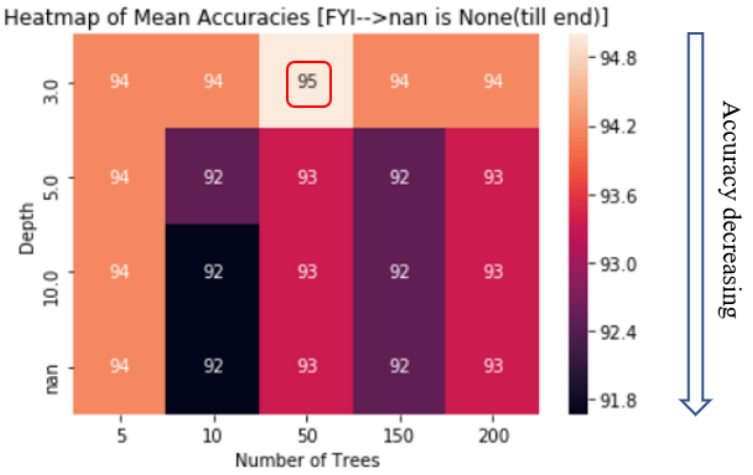


Figure 4.1

- In Gradient Tree Boosting classifier plot (refer Figure 4.3), "**number of grown trees = n_classes * n_estimators"** and I have got maximum accuracy for two different number of estimators (5,10). Now obviously value 10 will have more trees and 5 will have less, but since we know that both are providing same accuracy, we must look at which one is taking less time than other. And if you refer figure 4.2, clearly value 5 is taking significantly less time as value 10 i.e. speed of convergence for 5 is high.
- So, keeping in mind the above explanation, I would be choosing n_estimators =5 instead of n_estimators=10, as it is giving max accuracy in less time. As we increase the value of estimators, classifier's taking more time and providing less accuracy.

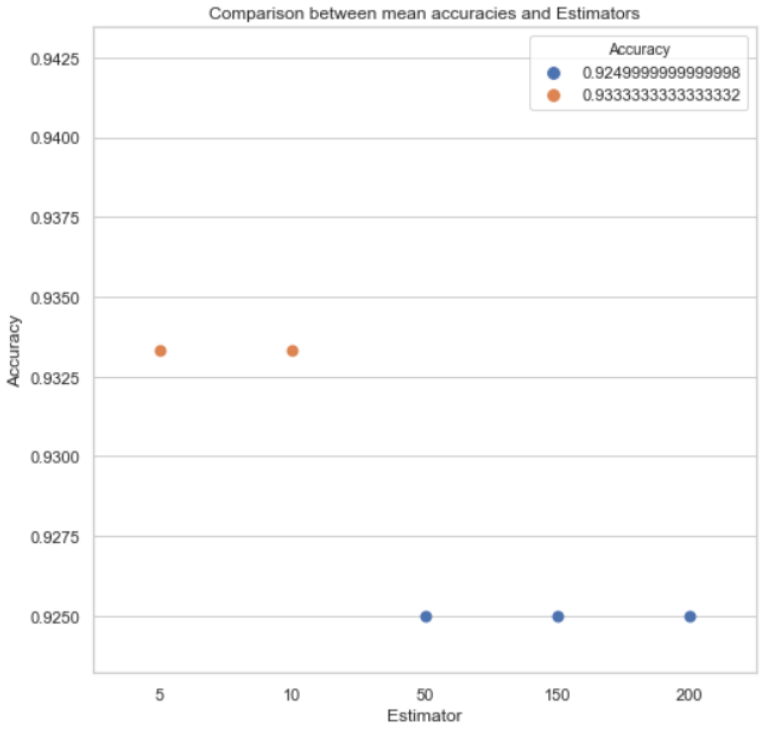


Figure 4.3 (Gradient Tree Boosting Plot)

```
For 5 Estimators
Classifier Accuracy is: 0.9333333333333332
Time:  0.07878994941711426
----------------------------------------------------
For 10 Estimators
Classifier Accuracy is: 0.9333333333333332
Time:  0.23232793807983398
----------------------------------------------------
For 50 Estimators
Classifier Accuracy is: 0.9249999999999998
Time:  0.9644060134887695
----------------------------------------------------
For 150 Estimators
Classifier Accuracy is: 0.9249999999999998
Time:  2.855318069458008
----------------------------------------------------
For 200 Estimators
Classifier Accuracy is: 0.9249999999999998
Time:  5.187086582183838
----------------------------------------------------
```

Figure 4.2 (Gradient Tree Boosting Classifier Output)

[To compare accuracy, Random Forest Classifier (95%) performed better than Gradient Tree Boosting (93.33%) however Random Forest Classifier takes 50 trees to produce that same accuracy level which is computationally expensive as compared to Gradient Tree Boosting.]

Q6: What does the parameter C define in the SVM classifier? What effect did you observe and why do you think this happened?

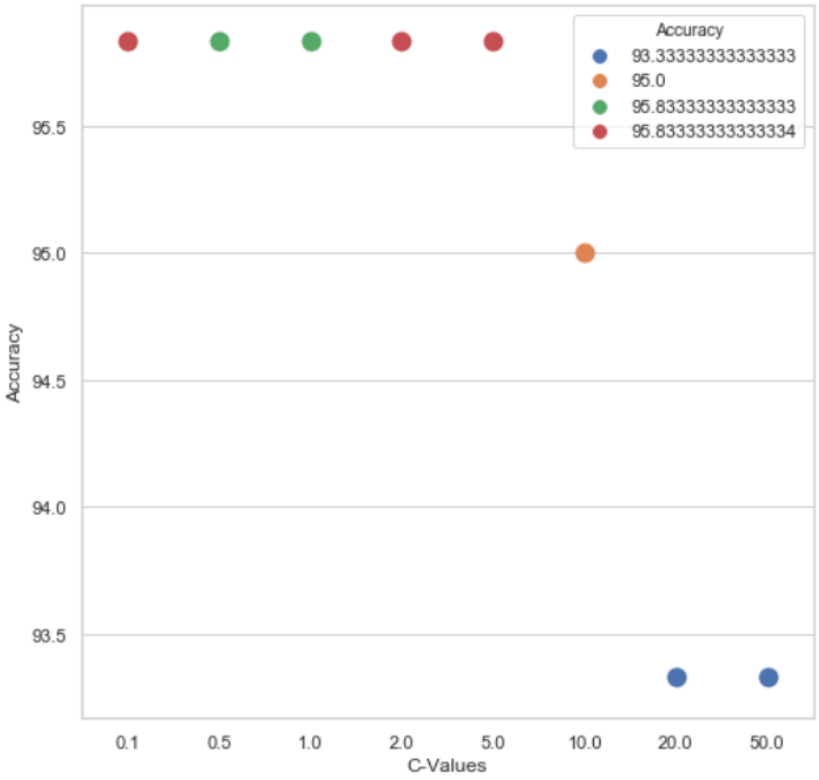
Answer:

In SVM, C refers as Regularization parameter. Main goal of SVM is to increase the separation between the classes using margin hyperplane and minimize the degree of misclassification.

- C or Regularization Value decides what kind of hyperplane your classifier would be constructing i.e. for higher values of C, classifier creates a small margin hyperplane which is vulnerable to overfitting, on the other hand for lower values of C, classifier creates a large margin hyperplane. When the value of C is too low, classifier neglects the outliers that causes misclassification between classes.
- C measures the tradeoff between largest margin of separation and smallest number of incorrect classifications.

Explanation of Plot:

- At C = 0.1, at this low value SVM classifier model will not consider the outliers (that causes misclassification) and provided a decent classification accuracy of 95.83333333333334 %.
- As C value increases, classifier won't be able to neglect the outliers and eventually classifier performance is dropped slightly. i.e. from 95.83333333333334 % to 95.83333333333333 %.



Some Important Question regarding classical methods

- At C = 2 and C =5, classifiers starts to produce smooth decision boundary, performs classification properly and again gave an accuracy of 95.83333333333334 % (this is like C =0.1 but here our classifier is not neglecting any outliers).
- As the value of C increases further, decision boundary becomes more and more smooth (almost transparent), which in turns effects model classification capacity and accuracy starts decreasing.
- Out of these 3 Values C = [0.1, 2, 5], when tested on test data results, C = 0.1 and C = 2 gave maximum accuracy. (Refer to notebook for detailed explanation of testing scenario).

Useful Links:

- https://stats.stackexchange.com/questions/31066/what-is-the-influence-of-c-in-svms-with-linear-kernel
- https://medium.com/@pushkarmandot/what-is-the-significance-of-c-value-in-support-vector-machine-28224e852c5a
- https://www.datacamp.com/community/tutorials/svm-classification-scikit-learn-python