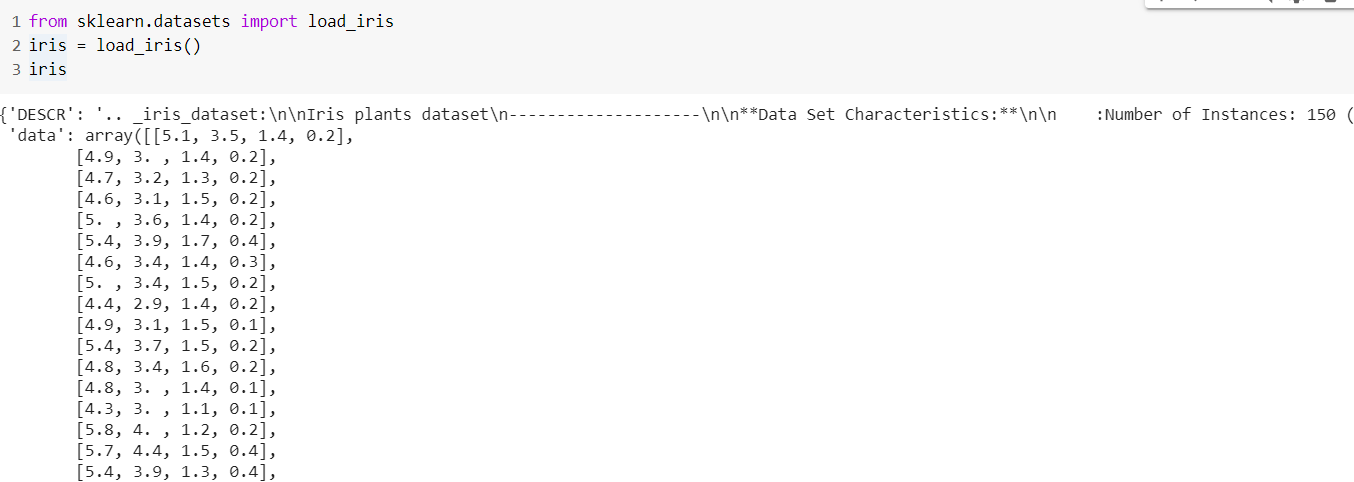
Aditya Pitchuka  
INFO 5502.002 (14676)  
Spring 2020

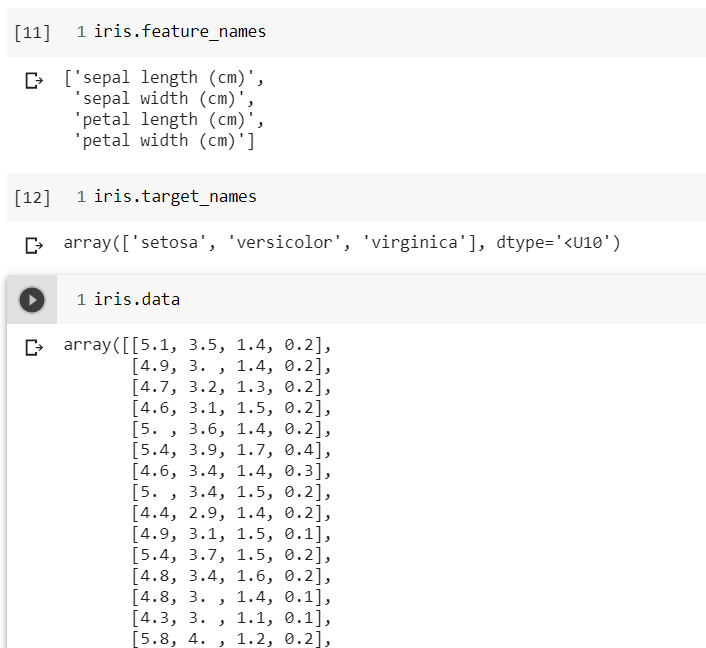
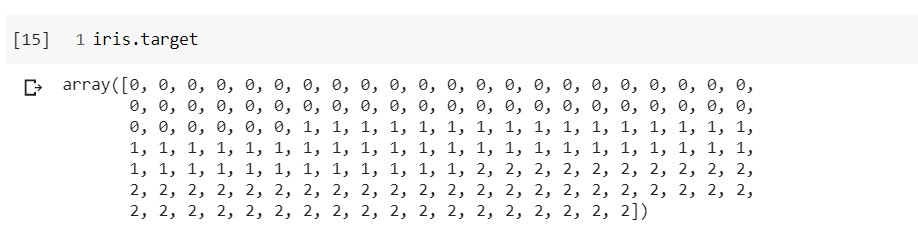
**Assignment – 7**

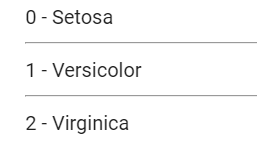
**EDA:**

Loading the dataset using the sklearn module.



Viewing the different rows and columns of the dataset.





We now use Classification as a supervised learning in which we use to categorise the different values.

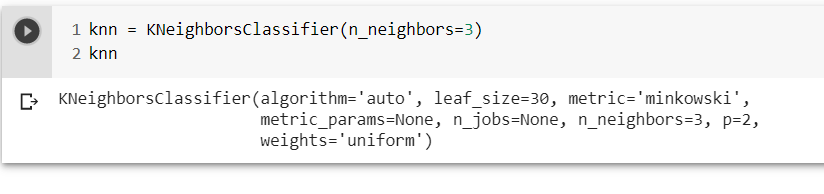
Using scikit-learn module we can implement the KNN algorithm that uses the numpy arrays for the different variables and they should be numeric. Hence, we classify them as 0, 1, 2 mapping to Setosa, Versicolor and Virginica respectively.

**KNN Algorithm:**

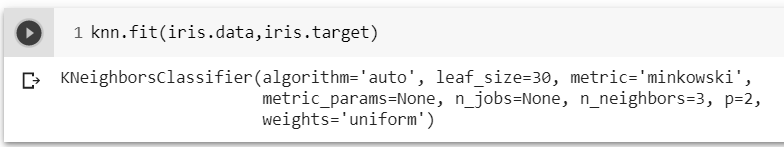
1. Pick a value of k
2. Search for k observations in the training data that are nearest to the measurements of the unknown iris.
3. Use the most popular response value from the k nearest neighbours as the predicted response value of the unknown iris.

We use the KNeighborsClassifier to create a classification model with neighbours as the argument.

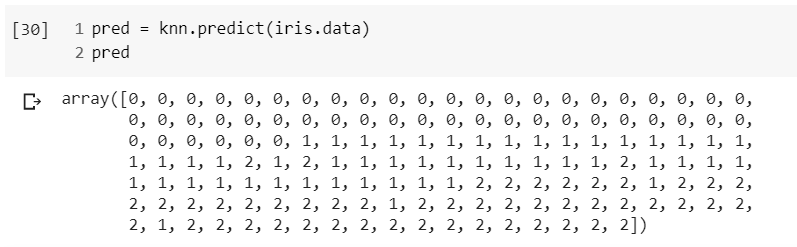
(k value is 3 in this case)



We train the model. We use the fit method with the two dependent and independent variables.



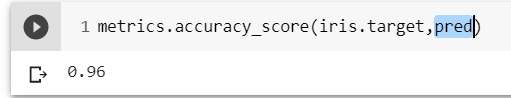
Now we predict the unknown variable using the predict function from the trained model.



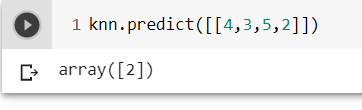
We feed the data from the table to the predict and returns the class of each observation.

We calculate the accuracy of the model now, 96%, as we trained the original this will be the accuracy of the original data.

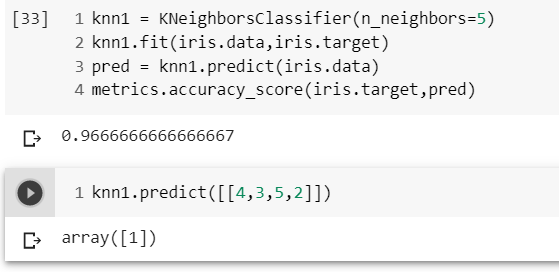
For this we use the metrics module.



For unknown values, we can now use this model to predict the class of these values. Here in this case, predict method returned an array of 2 which means the values belong to class 2 i.e., Virginica.



We can now repeat this with different values of k, say 5.



We can now see the accuracy, 96.67%, was even more increased and the unknown iris class is changed to class 1 i.e., Versicolor.

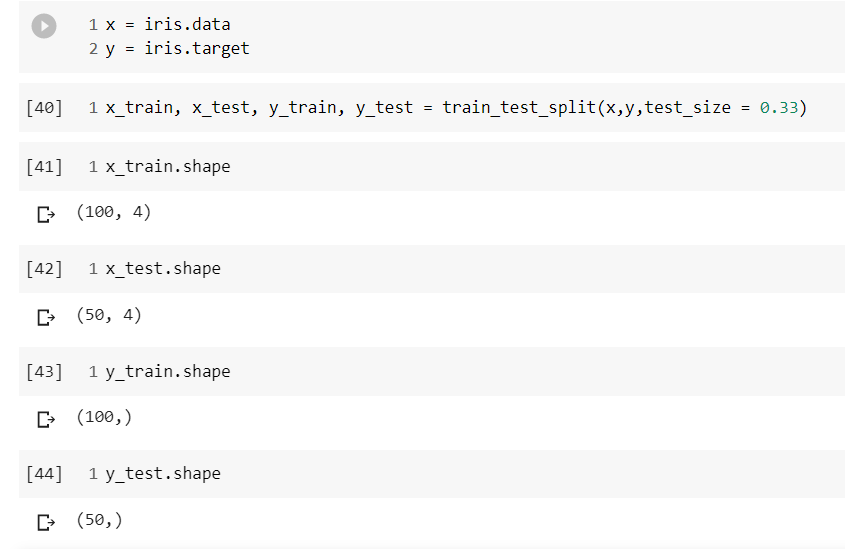
**Model Evaluation:**

We got two different values for our unknown iris, we need to know which one to trust and follow, so we have to evaluate/validate our training model by using a sample data from the original data and can choose the best value of k for KNN.

We chose the train/test/split procedure to evaluate our model. For this we use train\_test\_split class from sklearn module.

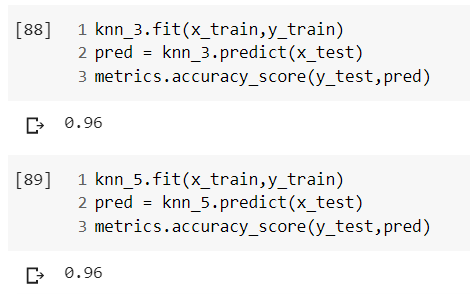
We train the model using the training set. Predict the values and test the results using the test set. Splitting is the test\_size of the testing set.

We use x as the independent values and y is our dependent values on the values of x which is the target array and the data respectively, we used earlier.

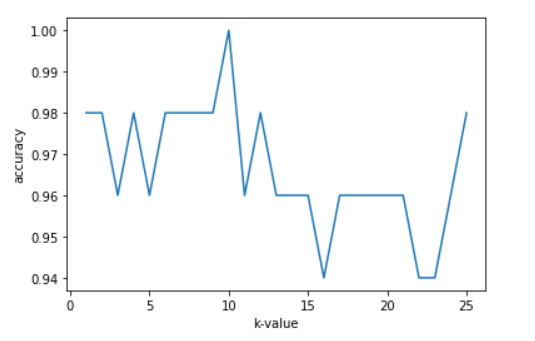


We split the data with 33% and 67% in this case and we check lengths in each of the training set and testing set.

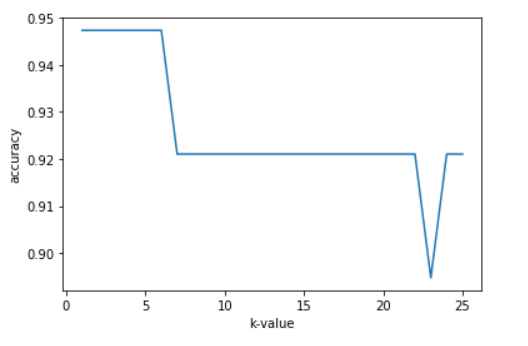
Now we use the above created classifiers and fit the models on our training set & testing set and check the accuracies.



We iterate for different values of k (from 1 to 25) and plot a graph for k values against the accuracies observed.



For splitting the data with 25% to 75% ratio we get as below.



From the above two graphs we see the value of k has the max accuracy for k = 9 and when splitting it 25% & 75% we see the value as k = {1,2,3,4,5}.

Testing accuracy is of high variance i.e., it can change a lot depending on the observations. This is way we say testing accuracy as high variance estimate.

To solve this problem, we can try averaging the testing accuracies for creating a bunch of train/test split. This can be done by using cross validation.

**K-fold Cross Validation:**

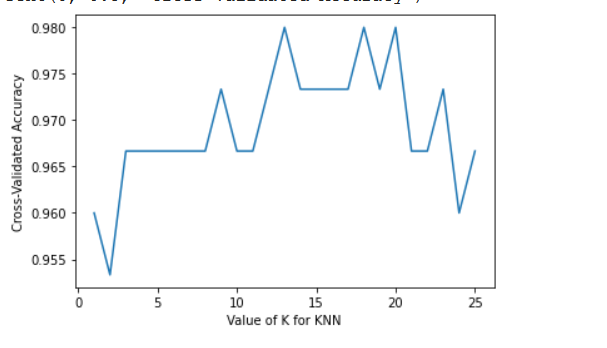
1. Split the dataset into K equal parts (folds).
2. Use one fold as testing and rest of them as training sets.
3. Calculate the accuracy for testing set.
4. Repeat the steps K times, so that each fold is used as testing set once.
5. Now we compute the average of the testing accuracies.

For example, if we have 10 observations in our dataset and we use a 5 fold then it works as follows:

1 2 3 4 5 6 7 8 9 10 🡪 Observations

|  |  |  |
| --- | --- | --- |
| Iteration | Testing set | Training set |
| 1 | 1 2 | 3 4 5 6 7 8 9 10 |
| 2 | 3 4 | 1 2 5 6 7 8 9 10 |
| 3 | 5 6 | 1 2 3 4 7 8 9 10 |
| 4 | 7 8 | 1 2 3 4 5 6 9 10 |
| 5 | 9 10 | 1 2 3 4 5 6 7 8 |

We implement the cross validation for k values from 1 to 25 find the scores and plot a curve.



We see high accuracies occur at k = 13 to 20. As it is a bias-variance trade off. Low value of k has low bias & high variance. And high k values have high bias and low variance. Best model is present in the middle as it balances the bias & variance.

When deciding best value of k, generally recommended to choose which has simple model. In KNN high values of k has low complex model. So, we chose as k = 20.

