Question – Tree Based Feature Selection

In the practical lab folder you will find a dataset called DataFile.csv. The dataset contains 5000 instances with 50 features and 1 binary class. The binary class is in the final column in the file.

Your objective is to perform tree-based feature selection on the dataset using a **RandomForestClassifier**.

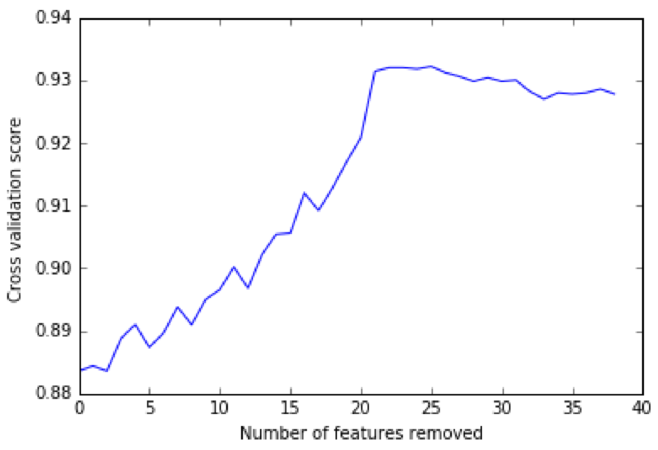
1. Read the dataset into a NumPy array (separate the class from the training data)
2. Perform **standardization** on the training dataset.
3. Split your dataset into training and test data using train_test_split (remember to set the random_state parameter to ensure reproducibility).
4. Create a kNN model and assess the accuracy of the kNN using the test data.
5. Create a **RandomForestClassifier** and run on the training data. This will score each of the features and will store this score in an array called feature_importances_. Remember, with a RandomForestClassifier, the higher the score the more important we deem the feature.
6. Running **np.argsort** on the feature importance array will provide you with the indices that specify an ordering on the feature importance array. In other words np.argsort will return an array of indices. The first index in this array will correspond to the feature that the RandomForestClassifier viewed as being the weakest feature (lowest value ranking).
7. Initialize a loop that begins to iteratively remove features from the dataset (starting with the weakest features). Remember you will need to remove the same features from the test and train data. For each iteration of the loop you build a kNN model and assess it accuracy on the new (reduced size dataset). The following method will allow you to delete columns from the dataset: **result = np.delete(X, indicesToDelete, axis = 1)** where X is the 2D feature training data and indicesToDelete is a list specifying the integer indices of the columns to be removed from the dataset.

Therefore, on the first iteration of the loop the weakest feature will be removed from the train and test feature data. Using this new data you should build a new kNN and assess it's accuracy. In the second iteration of the loop the weakest two features are removed and again you should build a new kNN model and assess its accuracy.

8. Each time around the loop record the accuracy achieved by adding to a list.

9. It might be helpful to produce a graph depicting the results of this process (see code below). In the code below the variable *numberOfFeatures* is a list containing the number of features removed for each iteration of the loop ([1, 2, 3…]). The *accurKNN* variable is a list that contains the accuracy of your kNN for each iteration of the loop.

```
import matplotlib.pyplot as plt

plt.figure()

plt.xlabel("Number of features removed")

plt.ylabel("Cross validation score ")

plt.plot(numberOfFeatures, accurKNN)

plt.show()
```


Cross validation score
Number of features removed