**Assignment 1**

**DATA MINING**

**Tasks**

Your tasks are as follows. In your code, make sure random\_state is not set.

1) Use one classification method on the dataset "wine.csv". You can apply any of the methods explained in this instruction notebook or any other method in scikit-learn. You can even implement your own method. You can tune your model by using any combination of parameter values. Use 75% of the data for training and the rest for testing.

2) Print out the accuracy of the model in 1).

3) Print out the confusion matrix for the model in 1). Note that we are dealing with a multi-class (5 quality ratings of wines) classification problem. So the confusion matrix should be 5 x 5. (Actually 6 x 6 since we are also printing the numbers of "All". Refer to the earlier example.)

4) Use the same model with the same parameters you have chosen in 1). However, instead of using 75%/25% train/test split, apply 10-fold stratified cross-validation.

5) Print out the accuracy of each fold in 4).

6) Print out the average accuracy across all the folds in 4).

I tried many models but finally chose **Random Forest algorithm** as classification method for supervised learning on the data set "wine.csv" as it gave me maximum accuracy.

One of the reasons for choosing this model was that because in this algorithm rather than greedily choosing the best split point in the construction of the tree, only a random subset of features are considered for each split. Since the data set given to us had 5 different classifiers so I thought using **Random Forest algorithm** would be the best way to construct model on the training data.

To increase the efficiency I played with mostly **all the parameters** but below ones helped me in increasing the efficiency:-

1. **n\_estimators** :- It is basically the number of trees in the forest, more number of trees better the results , but it increases the execution time also. So by randomly checking for different values I sticked to **value 100** as this gave me good efficiency value and execution time was not too long.
2. **criterion** :- The function to measure the quality of a split. We have two choices for this either we can use **‘gini index’** or we can take ‘**entropy**’ as the criterion. I tried with both but got more efficiency with entropy so chose that parameter.
3. **max\_features** :- The number of features to consider when looking for the best split. So I applied brute force and found the value 3 giving the best results on the accuracy.
4. **max\_depth** :- The maximum depth of the tree. Again I applied brute force and found the value 19 giving the best results on the accuracy.
5. **n\_jobs:-** The number of jobs to run in parallel for both fit and predict. If -1, then the number of jobs is set to the number of cores. I put the value to -1 in-order to increase the performance as much as possible, but it depends on machine to machine.

**Stratification** :-

For **10 k-fold cross validation implementation** I have first performed **Stratified Shuffle Split** on the given data set so that all the classifications are represented equally i.e it returns stratified randomized folds. The folds are made by preserving the percentage of samples for each class. This helped me in increasing the accuracy for cross validation.

After that on stratified randomized folds I have performed cross validation which gave me much higher accuracy than simply applying cross validation without stratification.

Below is the screen shot for the implementation showing accuracy, confusion matrix, 10 Kfold implementation :-

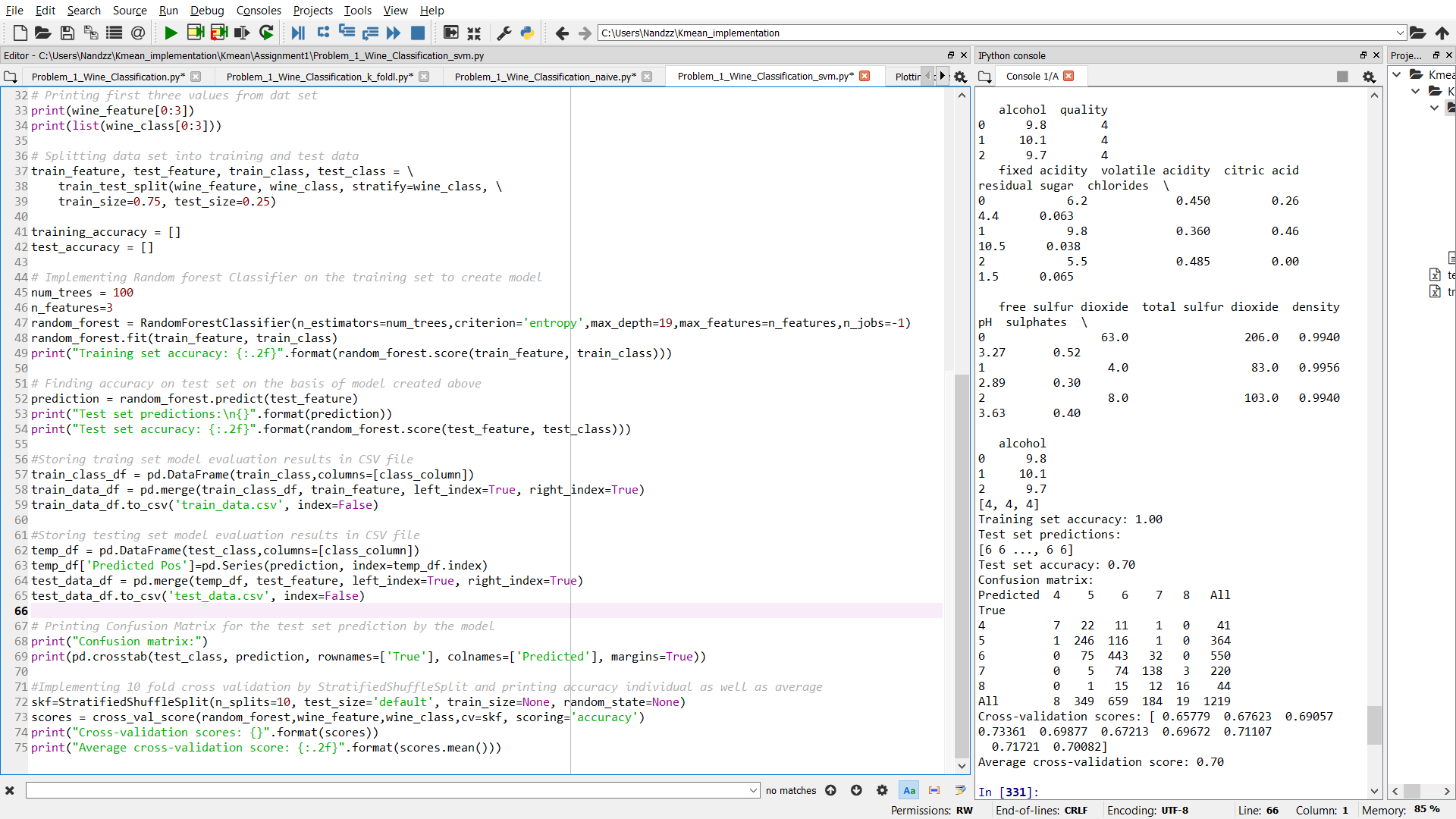
**Note :-**

**To run the code Wine.csv should be in the same folder as the code. Otherwise pandas will not be able to read the wine.csv file.**

**Version of python used 2.7 , IDE used by me SPYDER(Anaconda)**

I got test set accuracy as 70% as well as cross validation average accuracy as 70%.

Note :- Since random state is set to 0, hence we get different accuracy value each time when we run the code.



**References taken for help :-**

1. [http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#](http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html)
2. <https://machinelearningmastery.com/ensemble-machine-learning-algorithms-python-scikit-learn/>
3. <http://scikit-learn.org/stable/modules/generated/sklearn.model_selection.StratifiedShuffleSplit.html>
4. <https://www.datasciencecentral.com/profiles/blogs/random-forests-algorithm>
5. <http://localhost:8888/notebooks/Desktop/Data_Mining_Assignment/P.ipynb>
6. <https://www.mathworks.com/products/statistics/classification-learner.html>