APPENDIX

1. Install Jupyter Notebook.
2. Copy the folder containing source code files into the folder where Jupyter Notebook is installed.
3. Run Jupyter Notebook on your PC either by individually going to the installation folder or by searching it in the start menu and clicking the application icon.
4. After running the Application, search for the folder containing source code files.
5. Open the folder and click on the file which is to be run.

LOADING THE DATASET

In order to load the dataset into program there are two cases: -

1. **When your .csv file is in the same folder where Jupyter Notebook is installed.**

Suppose the name of the file is “data.csv”

There, this file can be easily read into source code data frames as follows: -

df = pd.read\_csv(“data.csv”)

1. **When your .csv file is in another folder**

Suppose the file with name “data.csv” is in the following directory: -

“C:\Users\ABC”

Then, this file can be loaded into dataframe in your source code as follows: -

Df = pd.read\_csv(“C:\Users\ABC\data.csv”)

RUNNING THE PROGRAM

* After loading the dataset in manner described above, the desired source code can be run y hitting “Run”.
* KNN and Random Forest Classifier takes less time and hence can be run on complete dataset to get the results.
* SVM (Support Vector Machine) takes time, so we have created a subset of data and .csv file of that subset is uploaded.

STEPS FOR RUNNING SUPPORT VECTOR MACHINE:

1. First run the file name svm\_preprocessing, in this file please enter the **path** of the csv files that are individual files, which already contains the important set of features.
2. Note: please don’t lease any tabbed space in path variable

E.g: The below one is wrong input can give error.



This below is the correct input:



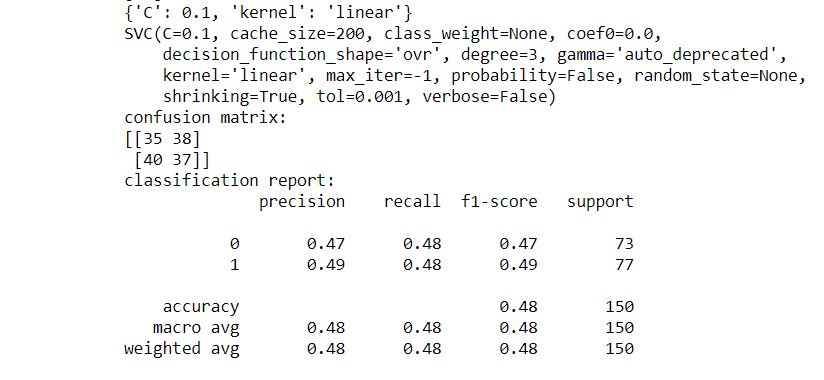
1. The files are: svm\_if = important features by information gain

svm\_re = important features by recursive feature elimination

svm\_et = important features by extra tree classifier

1. Now obtaining the C values from each of the above files **input the C** and **the path of this corresponding C** in the other file svm\_auc.

Eg.:This is where we find value of C in first programs: svm\_if, svm\_r, svm\_et for the second programs svm\_auc



1. If graph didn’t generated in the first attempt of execution then run the program svm\_auc, **another time**. There will be a graph as output
2. This file generates the Auc curve.
3. The outputs are in the folder.

PREPROCESSING

These files are there under preprocessing: -

1. Imputed Source Code: - In this code, we have dropped three columns “encounter\_id”, “patient\_nbr”, and “weight”.

The “encounter\_id” and “patient\_nbr” are just the serial number that just keep track of number of patients visiting the Hospital. “Weight” column has been dropped because it has 97% missing values and thus, would contribute little towards deciding “readmission” rate.

1. Encoding Source Code: - This code encodes the categorical columns into integer using Label Encoder. This was required because the libraries implementing “Random Forest Classifier”, “Support Vector Machine” and “KNN” cannot handle categorical data
2. Final Encoding Source Code: - In this code, we convert our classification problems into binary.

In original dataset, we have three values for class variable – [ ‘<30’, ‘>30’ , ‘No’ ]

That respectively describe that patient was re-admitted within 30 days, after 30 days and those who are not admitted.

To covert this into binary, we replace “<30” and “>30” with “Yes”.

There are two reasons for doing this: -

* The patient admitted within 30 days or after 30 days are re-admitted anyways.
* This also solves class imbalance problem

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | EXTRA TREES CLASSIFIER | | | | INFORMATION GAIN | | | | RECURSIVE FEATURE ELIMINATION | | | |
|  | **Accuracy** | **False Negative** | **Specificity** | **Sensitivity** | **Accuracy** | **False Negative** | **Specificity** | **Sensitivity** | **Accuracy** | **False Negative** | **Specificity** | **Sensitivity** |
| K NEAREST NEIGHBOURS | 0.57 | 8838 | 0.55 | 0.58 | 0.54 | 9045 | 0.51 | 0.56 | 0.54 | 9045 | 0.51 | 0.56 |
| SUPPORT VECTOR MACHINE (LINEAR) | 0.48 | 7342 | 0.59 | 0.46 | 0.58 | 8565 | 0.55 | 0.48 | 0.58 | 8621 | 0.54 | 0.55 |
| RANDOM FORSEST CLASSIFIER | 0.64 | 6620 | 0.63 | 0.64 | 0.59 | 7521 | 0.56 | 0.60 | 0.59 | 7532 | 0.56 | 0.60 |