CMPT459 Spring 2021
Data Mining
Martin Ester
TAs: Madana Krishnan Vadakandara Krishnan and Arash Khoeini

Milestone 3 and report of the Course Project

Deadline: April 26

Total marks: 100

Introduction

In the second milestone, you have built 2 or 3 baseline classification models. In the final milestone you will be tuning the hyperparameters, performing cross validation, comparing the models based on their performances and predict the result on the test dataset that you generated from milestone 1. You will also be submitting a complete project report which includes all the steps that you have performed as a part of the project.

More often than not, the baseline model is not the best performing model. Different algorithms have different hyperparameters that can be adjusted to get a better performing model. Depending upon the problem statement, a particular metric (ex: Recall) might be more important than the other (ex: Accuracy). For this project, the goal is to predict the outcome (hospitalized, nonhospitalized, recovered, deceased) of a case. It is crucial to predict the 'deceased' label correctly, with few 'false negatives', i.e. the 'recall' for class 'deceased' should be high. At the same time, the number of false positives should not be too high, i.e. you should not simply predict every case to be 'deceased'.

IMPORTANT

In your report, briefly describe the contributions and tasks completed by each group member, from the start to the end of the project.

Tasks

3.1 Hyperparameter tuning (**30 marks**)

Using the *training* dataset from milestone 2, perform hyperparameter tuning for all of your classification models. To do so, perform cross validation on the training dataset. Report the results (as a table, csv, txt or illustrative plots) for **all** the hyperparameter combinations that you have evaluated in this form:

Hyperparameters	F1-Score on 'deceased'	Recall on 'deceased'	Overall Accuracy	Overall Recall
hyp1=val1, hyp2=val2	---		---	---
hyp1=val3, hyp2=val4	---		---	---
.	.		.	.

.	.		.	.

Also, mention the parameters of the model that give the **best** result for F1-Score on 'deceased' in the report. Discuss the technique used (ex:GridSearchCV, RandomSearchCV, BayesianOptimization etc) for hyperparameter tuning, along with the reasoning.

Note that the overall recall of the model and recall for 'deceased' are different. The documentation link below provides a nice example of how to use multiple metrics and 'refit'.
https://scikit-learn.org/stable/auto_examples/model_selection/plot_multi_metric_evaluation.html

3.2 Comparative Study (**5 marks**)

From task 3.1, you now have 2 or 3 tuned models. Evaluate the performance of each of these tuned models. Use accuracy, precision, recall (from confusion matrix and/or classification report), and any other metrics that you think are relevant as performance metrics. Compare the performance of the different models and discuss their advantages and disadvantages. Finally, discuss which is the best model for the goal specified in the introduction of this document.

3.3 Prediction on test set **(15 marks)**

Predict the outcome label on the *test* dataset using the best tuned model. Save the prediction as 'predictions.txt', with every prediction beginning in a new line. Read the 'NOTES' section for more instructions. The predictions will be evaluated on F1-Score on 'deceased' using the ground truth labels that are held privately by the TA.

The marks for this task will be assigned according to the rank of the performance of your best model.

The evaluation criterion is relative.

- Relative – predictions will be evaluated with respect to how other groups perform. You will be assigned a rank based on ascending order of F1-Score on 'deceased', i.e, better predictions will have a higher rank.

Ex: There are 32 groups in total. If your F1-Score on 'deceased' is 2nd best among all submissions, you will have a rank of 31. Your mark will be (31/32)*15 = 14.53.

3.3 Report **(50 marks)**

As a part of the final milestone, you need to submit a project report. The project report should cover all tasks covered in the 3 milestones. Use the following outline for the report.

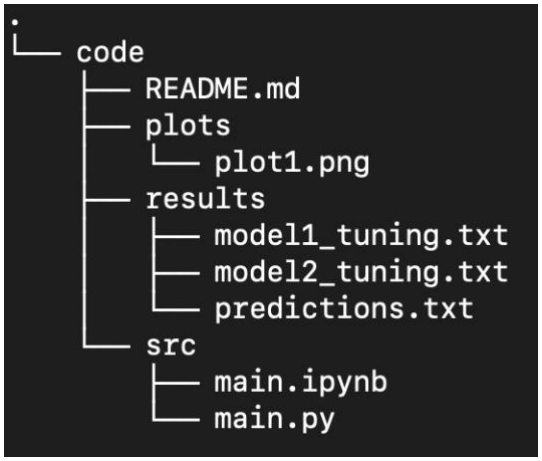
- **Project Title**
- **Problem Statement** – Define the goal of this project
- **Dataset description and EDA** – Insights derived from the dataset using visualizations or statistical techniques
- **Data preparation** – Discuss steps taken in data cleaning, outlier detection, merging datasets
- **Classification models** – Which classification techniques were used in this project? Justify why you chose a particular model
- **Initial evaluation and overfitting** – Brief description about results obtained for baseline models and checking for overfitting
- **Hyperparameter tuning** – Fine tune models and discuss approaches taken

- **Results** – Report and discuss results obtained during model tuning (accuracy, overall recall, F1-Score for 'deceased', recall for 'deceased')
- **Conclusion** – Compare performance of the models using standard metrics and report the best fit
- **Prediction on test dataset** – Discuss briefly how you use models to predict outcome labels on test dataset
- **Lessons learnt and future work** – What lessons did you learn during this project? How can it be improved further?
- **References** – Any references that you use (research papers, blogs, code references, tools etc)
- **Contributions** – Briefly describe the contributions and tasks completed by each group member, from the start to the end of the project.

In the report, please include plots, diagrams and tables that will help understand the work done. Ex: plots from EDA, training and evaluation. The page limit for the report is 10 pages. Reports should **NOT** exceed the page limit. The font size should be 12, and the page margins at least 2 cm.

Submission (Code + Report)

A). Code
Submit a 'code.zip' file with the following contents. It should contain the code that you have written for the hyperparameter tuning and comparative study task. You can include any plots that you might generate. Include the results from hyperparameter tuning task for all combinations within the 'results' folder in the form of .txt, .csv or images. Submit the **'predictions.txt'** file in the results folder. You do NOT need to save and submit the models. The structure can look like this.



- *plots/* folder contains any plots that you wish to generate
- *results/* folder contains hyperparameter tuning scores and the test data results as 'predictions.txt'
- *src/* folder contains code and scripts
    - *src/main.py*
    - *src/main.ipynb*
- *README.md* for any special instructions for code execution (ex: if main.py or main.ipynb should be executed, if third party APIs are used, high running time, links to dataset or saved models)

B). Report
Submit a 'report.pdf' file, as described above.

NOTE:

- If the model has many hyperparameters, you need not tune more than 3 hyperparameters (this will help reduce the runtime).
- In addition to hyperparameter tuning and cross validation, you are free to perform any other steps that you think will be benefit the final predictions.
- Read the following instruction very carefully about the test dataset clarification - https://coursys.sfu.ca/2021sp-cmpt-459-e1/pages/test_data_clarification/view
- If you performed additional preprocessing steps on train dataset in milestone 2 (ex: OneHotEncoding), you would have to perform similar steps on the test dataset as well, so that you can perform the predictions.
- Important points to note 'predictions.txt' are:
    - Every prediction should be written in a new line
    - There should be a total of 46,500 rows
    - Original test dataset order must be preserved
    - Prediction outcome names should match original dataset (nonhospitalized, hospitalized, recovered, deceased), and should not be 0, 1, 2 and 3
    - Do not add a header line in the predictions.txt file
    - Finally, before submitting, run through this file_verifier code snippet. If this fails, the scores for part 3.3 will not be marked.

```
def check_if_file_valid(filename):
    assert filename.endswith('predictions.txt'), 'Incorrect filename'
    f = open(filename).read()
    l = f.split('\n')
    assert len(l) == 46500, 'Incorrect number of items'
    assert (len(set(l)) == 4), 'Wrong class labels'
    return 'The predictions file is valid'

check_if_file_valid('predictions.txt')
```