CMPT459 D100 Spring 2024

Special Topics in Data Mining

Instructor: Martin Ester

TAs: Ali Alimohammadi, Arash Khoeini

Course Project Final Report

Deadline: Tuesday, April 9, 2024, 11:59 PM

Total Marks: 70

Introduction

Using the COVID-19 case data provided (download here), your goal is to build various classification models and evaluate the performance of the models. Each group member has to build one classification model. Thus, each group will have 2 or 3 models built, depending on the group size.

This is a dataset related to COVID-19 cases, specifically from the year 2021. It contains several columns with information about individual cases, including:

- **Age:** The age of the patient, although this column contains many missing values.
- **Sex:** The gender of the patient, also with missing values.
- **Province and Country:** Geographic information about where the case was reported.
- **Latitude and Longitude:** Specific geographic coordinates.
- **Date Confirmation:** The date on which the COVID-19 case was confirmed.
- **Additional Information:** This column seems to contain additional details about the case, but it also has many missing values.
- **Source:** The source of the information, likely where the data was obtained from.
- **Chronic Disease Binary:** A Boolean value indicating whether the patient has a chronic disease.
- **Outcome:** This column describes the outcome of the case, such as "Hospitalized" or "Recovered".
- **Outcome Group:** This column is present but appears to be empty or not applicable in the dataset.

The data seems to be primarily focused on tracking the status and outcomes of COVID-19 cases, with a particular emphasis on geographic and demographic information. However, there are notable gaps in the data, particularly in the age and sex columns.

More often than not, the baseline classification model is not the best-performing model. Different algorithms have different hyperparameters that can be adjusted to get a better-performing model. You will be tuning the hyperparameters, performing cross-validation, comparing the models based on their performances, and predicting the result (outcome) on the test dataset. The predicted results for the test dataset need to be submitted to CourSys for grading; both the base and relative performances will be considered for the grading of the test predictions (more on this in Task 8).

Depending upon the problem statement, a particular metric (e.g., F1-Score) might be more important than the other (e.g., Accuracy). For this project, the goal is to predict the outcome (hospitalized, non-hospitalized, deceased) of a case correctly with a small number of false negatives and false positives. For this reason, we will primarily focus on the macro F1-Score (mean F1-Score across all classes) as the evaluation metric for our models. For more information about the F1-Score, you can read it here.

You will also be submitting a complete project report which includes all the steps that you have performed in the project.

IMPORTANT

In your report, briefly describe the contributions and tasks completed by each group member for the course project's final milestone and report. We expect all group members to contribute equally and by default give the same marks to all group members. If it is evident from the contributions statement that some group members significantly under-contributed to the project, then those group members will receive only partial marks (for example, 50% of what the group received).

Getting Started

Python Packages

You might find the following Python packages useful for the tasks in this course project:

1. **scikit-learn**
   A Python machine learning library that includes various classification, regression, and clustering algorithms. You will also find this package useful for creating train/validation splits, evaluating your models, and hyperparameter tuning.
2. **Pandas**
   Python library for data manipulation and analysis.
3. **NumPy**
   Python library for scientific computing and working with arrays.
4. **Matplotlib**
   Python library for creating plots.
5. **Seaborn**
   Python library for creating prettier plots.

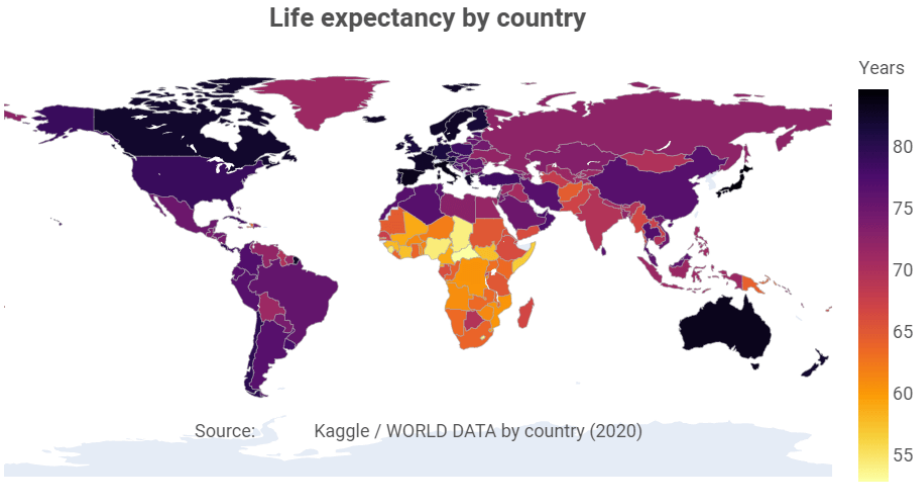
You are not limited to the Python libraries listed above. Feel free to use any Python packages/libraries you think are useful for your project.

Tasks

Note: responses/results that you will need to include in the report are highlighted by blue-colored text in the instructions below.

1. Data visualization (8 marks)

First, you have to visualize the data. Create a bar plot of data availability based on continents. Are there any anomalies? How do they compare to the continents' overall populations? Use a color spectrum for different proportions; for example, the bars for the continents with a higher data availability should be redder and bluer otherwise. Create a colored heatmap of the world, concerning the percentage of available data on the countries' populations. An example of this is life expectancy based on countries. **Note that you have to use both the location and cases csv files and combine them. Report all the plots and your interpretation of them.**



2. Data Preprocessing (5 marks)

Handle the missing values. Join both location and case data on both country and province together as a primary key and add an 'Expected_Mortality_Rate' to the cases dataset. **Find out what is the best strategy for dealing with missing values and mention and explain your choice in the report.**

Using the knowledge from the last two parts, continue with this dataset, a subset of the former dataset and complete the following tasks using ONLY this subset.

3. Feature selection (1 mark)

Select reasonable features to use for the train and test datasets. Make sure that the train and test datasets contain identical selected features. Throughout the project, you may add/remove features as you see fit. Note that adding/removing features may help you train better-performing models. **Report the final features that are used.**

4. Mapping the features (2 marks)

Map the categorical features (e.g., sex, chronic disease binary, etc.) and the outcome_group (the response variable you will be predicting) to numeric values. For example, 0 can be used to represent 'male' and 1 can be used to represent 'female' for the 'sex' feature. **In the report, explain how you mapped the features**

and justify your approach. Finally, for the 'outcome_group', everyone must use the same mapping, which is **0: 'deceased', 1: 'hospitalized', 2: 'non_hospitalized'**. You may find this resource helpful for mapping the 'outcome_group' response variable.

Note: Make sure you map the categorical features of both the train and test datasets.

5. Balancing the classes in the training dataset (4 marks)

You will notice that the classes in the training dataset are heavily imbalanced, meaning that some classes have significantly more cases (examples) than others. When the data is imbalanced, it can be challenging to train models that can accurately predict the outcome of cases, especially those belonging to the minority class(es). Therefore, it would be reasonable to balance out the classes in the training dataset first before training the classification models. **Discuss how you balanced the training data and justify your approach. Report the original number of cases in each class (before balancing) and the final number of cases in each class (after balancing).**

6. Building models and hyperparameter tuning (16 marks)

As mentioned in the introduction, each individual in a group has to build a classification model. The models can be of any type and you can use any existing Python libraries (e.g., Scikit-learn, PyTorch, etc.) to build them. The models could include boosting trees, SVMs, KNN, Decision Trees, Random Forests, MLPs/Neural Networks, Naive Bayes or any other classifiers that you think could be appropriate for the problem statement. **For each model used, explain why you selected it over other models.** You can split the *training* dataset further into *training* and *validation* sets with an 80:20 train-to-validation ratio. You will then train your model on the training set and evaluate your model on the validation set. Doing so allows you to get a sense of how your model performs based on the results you obtain on the validation set since you do not have access to the ground-truth test data labels.

To improve your models, hyperparameter tuning is an important procedure. Using the *training* dataset provided in the download link in the Introduction section, perform hyperparameter tuning for all of your classification models. To do so, perform K-fold cross-validation on the training dataset. You may choose a value of K that you think is reasonable. **Report and justify the value of K selected. Discuss the technique used for hyperparameter tuning (e.g., GridSearchCV, RandomSearchCV, BayesianOptimization, etc.) along with the reasoning. Report the hyperparameter values (e.g., ranges) you tried for hyperparameter tuning and explain why they are selected. Save all results (i.e., different hyperparameter combinations and their corresponding model performance evaluated using the three metrics shown in the table below) from hyperparameter tuning of each model as .txt files (more instructions in the Submission section). Finally, report the results and the hyperparameter combination for the best-performing model per group member in the following form:**

Model	Hyperparameters	Mean macro F1-score across the validation sets	Mean F1-score on 'deceased' across the validation sets	Mean overall accuracy across the validation sets
[Model 1]	hyp1=val1, hyp2=val2, …			
[Model 2]	hyp1=val3, hyp2=val4, …			
[Model 3]	hyp1=val5, hyp2=val6,			

7. Overfitting (6 marks)

It is not uncommon for the classification models to overfit. **How did you check for overfitting? Explain using plots and/or evaluation metrics.** You can vary the values of at least one hyperparameter, train models for different values of that hyperparameter and then compare the performance.

8. Comparative study (5 marks)

From Task 6, you now have 2 or 3 tuned models as well as their performances on the validation sets. **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 and why.**

9. Prediction on the test set

Predict the outcome label on the test dataset using the best-tuned model(s). Using the snippet of code provided below, save the predictions as a .csv file with columns "Id" and "Prediction". DO NOT shuffle/reorder the rows in the test dataset, otherwise, your predictions will be graded incorrectly. The predictions will be evaluated on the macro F1-Score using the ground-truth labels that are held out by the TA.

Code snippet:

```
def create_submission_file(y_preds, file_name):
    with open(file_name, "w") as csvfile:
        wr = csv.writer(csvfile, quoting=csv.QUOTE_ALL)
        wr.writerow(["Id", "Prediction"])
        for i, pred in enumerate(y_preds):
            wr.writerow([str(i), str(pred)])

create_submission_file(y_preds, "submission_[model_name].csv")
```

In the code snippet above, `y_preds` should be a NumPy array or list containing the predicted classes of the test cases/examples. `[model_name]` in `'submission_[model_name].csv'` is the name of your model (e.g., 'svm','rf', 'nn', etc.).

The marks for this task will be assigned based on base and relative performances.

Base performance (3 marks)

Each of your models should achieve a mean macro F1-Score of **0.70** on the validation set across all of the cross-validation splits and a mean F1-Score of at least **0.40** for the 'deceased' class to receive full marks (3 marks). **In your report, attach a screenshot of the output macro F1-Score and F1-Score for all three classes for each of the 2-3 models (one model per group member).**

Relative performance (10 marks)

The marks for this portion will be assigned according to the performance ranking of your best model based on the macro F1-Score on the test data predictions. Predictions will be evaluated with respect to the performance of other groups. The rank-to-marks conversion is shown in the table below.

Rank	Marks	Rank	Marks
Top 10%	10	Top 60%	4
Top 20%	8	Top 70%	3
Top 30%	7	Top 80%	2
Top 40%	6	Top 90%	1
Top 50%	5	Remaining	0

Each group needs to submit **one** set of predictions for the test data as a .csv file to CourSys for the relative performance grading. Please choose the best model from your group to make the predictions for submission.

Please double-check to make sure that your predictions .csv file has a total of 4304 rows (predicted classes for the test data). You will be responsible for ensuring that the format of your prediction file is correct. Incorrectly formatted prediction files will not be marked.

10. Report (10 marks)

The project report should cover all tasks outlined above. One mark is given per section of the report (excluding the project title) based on *readability* and *quality*. For report sections that contain at least one task (e.g., Data preparation contains tasks 3,4 and 5), make sure to clearly label the task (e.g., '3 feature selection') and place all relevant responses/discussions under their corresponding tasks.

Use the following outline/checklist for the report.

- Project Title
- Problem Statement (1)
- Data visualization (1)
    - Task 1
- Data preprocessing (1)
    - Task 2
- Data preparation (1)
    - Task 3 feature selection
    - Task 4 mapping the features
    - Task 5 balancing the classes in the training dataset
- Classification models (1)
    - Task 6 building models and hyperparameter tuning
    - Task 7 overfitting
    - Task 8 comparative study

- Model predictions (1)
  - Task 9 prediction on the test set
  - Additionally, discuss the results reported in task 9
- Conclusion (1)
  - Briefly summarize what you did in tasks 3-9
  - Summarize any interesting or key findings
- Lessons learnt and future work (1)
  - What lessons did you learn during this project? How can it be improved further?
- References (1)
  - Any references that you used (research papers, blogs, code references, tools, etc.)
- Contribution (1)
  - 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 all relevant plots, diagrams, and tables that will help the TA understand the work done. For example, plots from training and evaluation, etc. **Figures and tables not included in the report will not be marked.** The page limit for the report is 8 pages (not counting the title page for those who want to include one). Reports should NOT exceed the page limit. The font size should be at least 12. The font in the body of the report should be 12pt and titles/headers should be larger than 12pt. Finally, the page margins should be at least 2cm.

Submission

Submit your **code** (as a .zip), **report** (as a PDF file), and **predictions** (as a .csv file) to CourSys no later than Tuesday, April 9, 11:59 PM. **No late submissions will be accepted, and no grace days will be given.**

Code

Submit a 'code.zip' file with the contents specified below. You can include any plots that you might generate. Include the results from the hyperparameter tuning task for all combinations within the 'results/' folder in the form of .txt. You do not need to save and submit the models. The structure should look like this:

```
.
└── code
    ├── README.md
    ├── plots
    │   └── plot1.png
    ├── project.ipynb
    └── results
        ├── model1_tuning.txt
        └── model2_tuning.txt
```

- *plots/* folder
    - Contains any plots that you want to generate
- *results/* folder
    - Contains hyperparameter tuning results for each model
    - Groups of three should have model3_tuning.txt (not shown in the sample structure above)
    - '`model[1/2/3]`' in '`model[1/2/3]_tuning.txt`' should be replaced with the actual model names (e.g., 'randomforest_tuning.txt', 'svm_tuning.txt')
- *project.ipynb*
    - This Jupyter notebook should contain ALL of the code you have written for Tasks 1.1-1.7
- README.md
    - Include any special instructions for code execution (e.g., if 3rd party APIs are used, high running time, etc.)
    - Include the Python version and the Python libraries/packages used.

Report

Submit a 'report.pdf' file as described above.

NOTE:

- In addition to hyperparameter tuning and cross-validation, you are free to perform any other steps that you think will benefit the final predictions. Make sure you discuss it in your report.
- If you performed additional preprocessing steps on the training dataset (e.g., one-hot encoding or converting categorical values to numerical values), you would have to perform similar steps on the test dataset as well so that you can apply the trained models to make predictions.