Final Project Report

CMPT 459 D100

Instructor: Martin Ester

Yuhang Yuan (301357634)
Sunny Yang (301351209)
Zhe Liu (301449316)

Problem Statement

In December 2019, the first cases of Coronavirus disease 2019 (Covid-19) were reported in Wuhan, China. The high prevalence of Covid-19 has made it a new pandemic in most countries. Hence for the longevity of the world, it is imperative for researchers to understand and predict the transmission of the virus.

Operating upon two provided COVID-19 case data sets, the goal of this project is to evaluate various models on their performance in predicting Covid-19 outcome cases. Through tuning hyperparameters, cross-validations, and comparing various model predictions against each other, the aim is to find the best-performing model based on macro-F1 values for Covid-19 outcome predictions.

Data Preparation

1.1 Feature Selection

Through the EDA assessment done in Milestone one, it was determined that other than '*outcome_group*' used as class labels, the "best" features to use are **[*'age,' 'sex,' 'province,' 'country,' 'chronic_disease_binary,' 'Case_Fatality_Ratio*']**

1.2 Mapping the Features

Categorical values such as *'sex,' 'province,' 'country,' 'chronic_disease_binary'* were encoded and mapped into numeric values using the *preprocessing* library provided within sklearn. The library automatically encodes by assigning a unique numerical value to each categorical value. The library can take any input and accurately convert it to numerical without compatibility issues which justify the use of this library.

Outcome_grouped mapped following ***0: 'deceased,' 1: 'hospitalized,' 2: 'nonhospitalized.'*** which was provided in the project description by Dr.Ester

1.3 Balancing the classes in the training dataset

Between *Undersampling* where the majority class labels are balanced to equal the minority class and *Oversampling* where the values within minority class are repeatedly sampled to match the majority class labels; ***we opted for undersampling the dataset*** as it improves the F1 values of models seen in later parts of this report. Specifically, all the samples within the minority class were kept, while the same amount of values was resampled from the majority classes so that the number of samples within each class was equal. See **figure 1** below for the number of cases(*before balancing*) and (*after balancing*)

Original Data

Labels	Counts
0	997
1	13241
2	2974

Undersampled Data

Labels	Counts
0	997
1	997
2	997

Figure 1

Classification Models

1.4 Models, and Hyperparameter tuning

Among the numerous classification models at our disposal, our group opted to choose Random Forest, KNN, and SVM to predict outcome groups:

Reason for choosing Random Forest:

- Provide easy-to-understand predictions
- Handle large datasets efficiently
    - The trees are independent of each other during training
- Compared to decision trees it solves the issue of overfitting
    - Establishing root nodes and the separation of the nodes are performed randomly.
- Reduce the impact of outliers
    - Random forest use the data to build multiple decision trees; even if some individual decision trees may cause inaccurate predictions due to the influence of outliers, the prediction results are obtained by referring to multiple decision trees. Therefore, the influence of outliers is reduced.

Reason for choosing KNN:

- Fast training prediction
    - KNN is a lazy learning algorithm. Instead of all data points to predict, it only uses the K-Nearest neighbours. Therefore, it is much faster than other algorithms.
- Short prediction time
    - KNN is a memory-based approach; it allows the algorithm to respond quickly to changes in the input during real-time use.
- Easy to hyper-tune due to limited amount of parameters

Reason for choosing SVM:

- Low computational overhead (handle feature sets efficiently)
    - SVM algorithm is only determined by a small number of support vectors; the complexity depends on the number of support vectors rather than the dimension of the entire sample space
- Difficult to overfit
    - SVM uses regularization to be resistant to over-fitting
- Productive in high dimensional spaces
    - SVM works comparably well when there is an understandable margin of dissociation between classes

K values:

Through online resources, it was recommended that the best K value exists within a range from 5-10. Through some testing in our models, it was discovered that having K = 5 produces the highest validation score among available K values.

Hyperparameter tuning:

We used *GridSearchCV,* which exhaustively searches through all combinations of provided values in each hyperparameter until the best combination is found. GridSearchCV was chosen for ease of setup, giving us control over the combination of parameters to test. This was done for all three models within this project, while the hyperparameters tuned within each model were selected based on our belief of the most important and effective influence of knowledge learning with the class.

Reason for Hyperparameter range:

*Specific Hyperparameters used in each model are chosen through the recommendation of online sources, while some are chosen due to being specially mentioned during lectures which elude to their importance

- Random Forest
    - ‘Criterion’- [‘Gini, 'entropy’]:
        - Only two criterions available for Random Forest
        - We choose to tune this parameter due the influential effects seen in the outcome of evaluation functions used in the random forest.
    - N_estimators':[50,100], 'min_samples_split':[2,10,20], 'max_depth':[100, None]
        - After reading the documentation of Random Forest, values chosen in the hyperparameters were within range of the recommended values by experts within this field.

- KNN
  - 'N_neighbors': range(5, 56, 5)
    - k = 5 was chosen as it is the default recommended value provided within the documentation, while research suggests an N value of $\sqrt{total\ sampled} \approx 3000 \approx 55$)
    - This hyperparameter is essential to make sure that the model doesn't overfit or underfit during training.

  - 'weights':('uniform,' 'distance'),
    - Only two weights possible

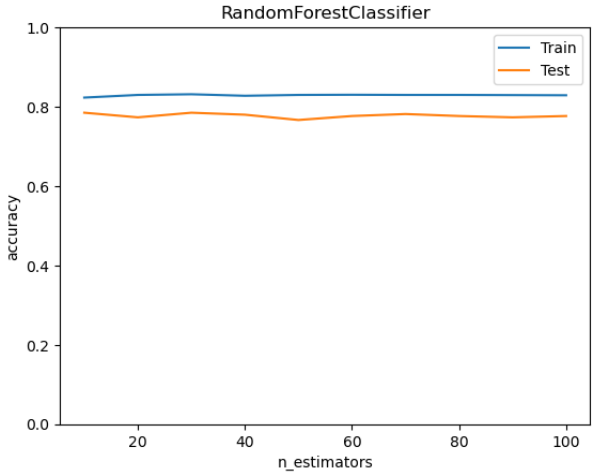
  - 'P':[1,2]
    - Only two distance functions available (manhattan_distance) and (euclidean_distance)

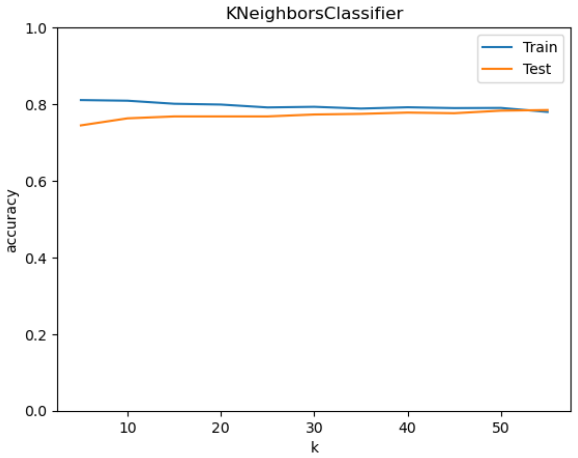
- SVM
  - 'kernel':('rbf,' 'poly,' 'linear'),
    - Among the recommended kernels found within online sources, we discovered that these three possible hyperparameters values for kernel provided the best results hence the reason consideration within GridSearchCV
    - The kernel is the core mathematical function behind how SVM transforms data into a required form thus elevating its importance in tuning

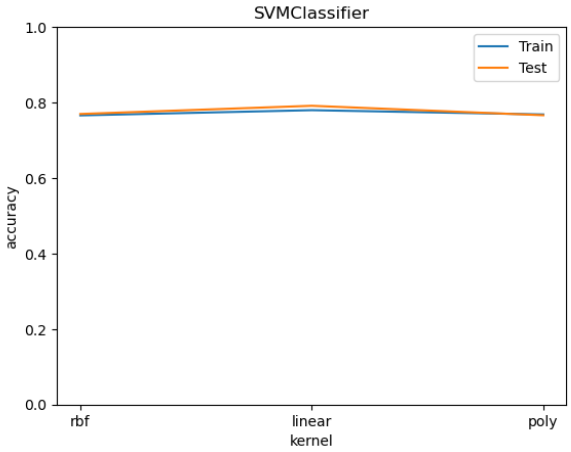
  - 'Cache_size':[200,500]
    - Range based on the available memory that our hardware had at disposal for generating the model

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
Random Forest	Criterion =Gini N_estimators = 50 Min_samples_split = 20 max _depth = None	0.792376	0.699489	0.793070
KNN	N_neighbors = 20 P = 1 Weights = uniform	0.783261	0.692122	0.785545
SVM	Kernel = linear Cache_size = 200	0.772266	0.674752	0.775931

1.5 Overfitting







We attempted to identify overfitting by plotting the accuracy points of training and testing against changes in one hyperparameter per model, such as *n_estimators* for **Random forest**, *K* for **KNN** and *kernel* for **SVM**. The conclusion we are looking for follows that if training and test accuracy deviate from each other too much, it would be a sign of overfitting. However, from the graphs, it can be seen that among the three models, all training and test plots seem to be within close range of each other. Therefore, we can conclude that overfitting did not occur in our models. Though it was noticed that changes in overall accuracy could be seen depending on how we balanced the data.

1.6 Comparative study

- Comparision of Random Forest, KNN, SVM
    - Speed: The computation of Random Forest is less intensive than KNN and typically runs faster than SVM
    - Random forest covers the problem of low bias/high variance of decision trees by generating an entire forest of decisions
    - Random Forest's prediction is easier to explain the reason for the decision made as opposed to SVM
    - Random forests are robust to outliers since they get averaged out by the aggregation of multiple trees. Meanwhile, KNN is highly sensitive to outliers
    - Random Forests are unaffected by whether data is categorical or numerical and thus do not have to worry about data loss when converting data type
- Comparing the best results of the three models, Random forest leads the scores for all three metrics. The goal of this project is to find and tune a classification technique so that it can accurately predict the outcomes of Covid-19 cases. This is achievable through the random forest model, which can reach a decently mean accuracy of 0.79 across validation sets.

Model predictions

1.7 Prediction on the test set

Base Performance: Screenshots

- Random Forest

	params	mean_test_Macro F1	mean_test_Deceased macro F1	mean_test_Accuracy
10	{'criterion': 'gini', 'max_depth': None, 'min_...	0.792376	0.699489	0.793070

- KNN

	params	mean_test_Macro F1	mean_test_Deceased macro F1	mean_test_Accuracy
12	{'n_neighbors': 20, 'p': 1, 'weights': 'uniform'}	0.783261	0.692122	0.785545

- SVM

	params	mean_test_Macro F1	mean_test_Deceased macro F1	mean_test_Accuracy
2	{'cache_size': 200, 'kernel': 'linear'}	0.772266	0.674752	0.775931

Conclusion

Machine learning is the concept of using classification models built using different algorithms to analyze the intrinsic connections of data. Such models can help people make predictions or decisions about subsequent data sources of the same type. The COVID-19 pandemic has had a catastrophic impact around the world. By collecting data and using machine learning, we can accurately diagnose COVID-19 victims.

The first step of building classification models for the COVID-19 dataset is feature selection. We selected the features we needed through EDA assessment. And then, in the data preprocessing step, we performed data mapping, through which we can map complex data formats into model-friendly numeric values. In addition, we ensure class balance by undersampling the dataset. Based on several criteria in 1.4, we chose Random Forest, KNN, and SVM over the other models. We determine the values of hyperparameters by referring to online resources and using GridsearchCV. Overfitting is common in classification models. To avoid overfitting, we plot the data to observe if the model is overfitting visually. Finally, by comparing the performance of the three models in the validation dataset, we have determined the Random Forest obtained the highest F1 Macro score.

An interesting finding of this project is when we checked whether the model is overfitting, we found that the accuracy of the SVM classifier is almost the same in the training dataset and testing dataset; therefore, we believe that SVM will have the highest accuracy for the validation dataset. However, the final data shows that SVM performs the worst among the three models.

Lessons Learnt and future work

Within the project across milestone one and this final project, we learned methods in data preprocessing, outlier detection, hyperparameter tuning and most important of all, the building of classifiers models. The biggest struggle within this project was finding the “best” classifier to use for predicting COVID-19 outcomes. Additionally, determining the best hyperparameter to set for each chosen model.

Though difficult, this project did help us gain experience and insight into what classification methods are available at our disposal. In future, this experience should help us create better and more accurate models in future data mining projects.

References

[1] Naukri.com. (n.d.). Retrieved December 6, 2022, from https://www.naukri.com/learning/articles/how-to-set-the-value-of-k-in-k-fold-cross-validation/

[2] *Sklearn.ensemble.randomforestclassifier*. scikit. (n.d.). Retrieved December 6, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[3] *3.2. tuning the hyper-parameters of an estimator*. scikit. (n.d.). Retrieved December 6, 2022, from https://scikit-learn.org/stable/modules/grid_search.html

[4] *Sklearn.neighbors.kneighborsclassifier*. scikit. (n.d.). Retrieved December 6, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[5] *Sklearn.svm.SVC*. scikit. (n.d.). Retrieved December 6, 2022, from https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

Contributions

Yuhang Yuan (301357634):

- Built Random Forest Model + hyperparameter tuning of own model
- Drafted outline and contents of project reports in milestone one and final project
- Data visualization for overfitting section

Sunny Yang (301351209):

- Built + Hyperparameter tuning KNN model
- Dataset preprocessing, class balancing
- Finalised Code formatting and bug fixes

Zhe Liu (301449316):

- SVM model construction and hyperparameter tuning
- Finalised projects reports (grammar)
- Feature selection and hyperparameter research