Report

Project Title

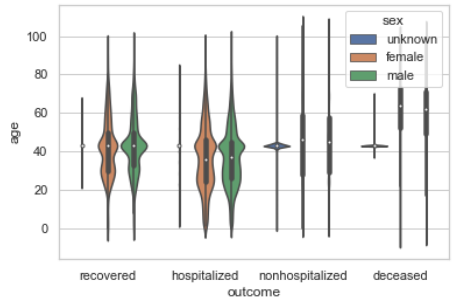
Predicting Outcome of COVID-19 Patients

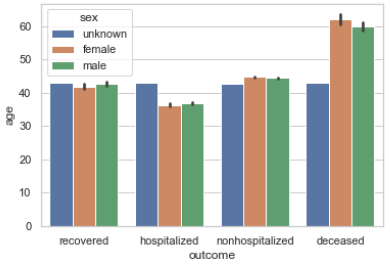
Problem statement

The problem that we are trying to address is accurately predicting the outcome of COVID-19 patients. We measure accuracy using the accuracy score, precision, recall and the F-1 score of the predicted outcome. This will be achieved by building, training and tuning three different models that can be used to predict outcomes on a dataset of COVID-19 patients.

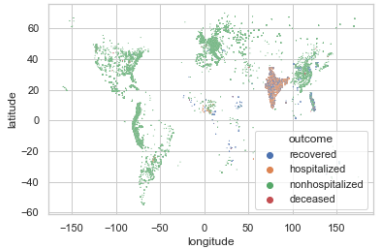
Dataset Description & EDA

We first explored to see if gender had an impact on the outcome by creating a box plot and a violin plot versus age. No trends were found by gender.

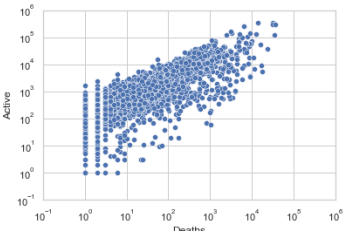


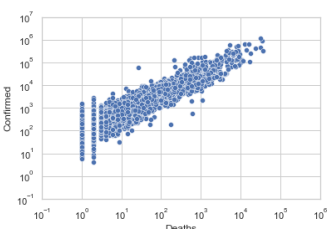


Then we created a scatter plot on lat and lon colored by outcome to see if any regions in the world showed a trend. It showed how there are some hotspots for certain outcomes



We then made a scatter plot on confirmed deaths, recovered, active, incidence_rate, and case fatality ratio. Since they showed an exponential growth trend, we took the natural log of all of those columns to see if there were any linear relationships between these columns.





Data Preparation

During data cleaning, the average age was imputed and invalid age format was transformed into an integer. The “source” and “additional_information” column was dropped as it was irrelevant. The “sex” column was transformed unkown into a third class. The location data set imputed missing values as zero or recalculated the value based on other columns. Outliers were not deleted.
The location data was joined on casesTest and casesTrain data set on “combinedKey” column, where combinedKey column is in the format of “{province},{Country}” merging the two tables.

Classification Models

Decision tree is a supervised machine learning algorithm where data is continuously split according to a certain parameter to obtain a classification outcome. It was chosen as one of the classification models because it is resilient to outliers, which are quite prominent in the dataset provided to us. Moreover, decision trees are an ideal choice for predicting categorical labels while being easy to understand and interpret.

XGB boost was chosen to take advantage of its ensemble learning. Because there are lots of outliers due to the nature of the data being logarithmic ensemble learning can reduce the variance caused by the outliers.

KNN is a non-parametric (there is no assumption for underlying data distribution) and lazy learning (it does not need any training data points for model generation) algorithm. All training data is used in the testing phase. In KNN, K is the number of nearest neighbors. For finding closest similar points, you find the distance between points using distance measures such as Euclidean distance, Hamming distance, Manhattan distance and Minkowski distance. KNN first calculates the distance, then finds closest neighbors, and finally, votes for labels.

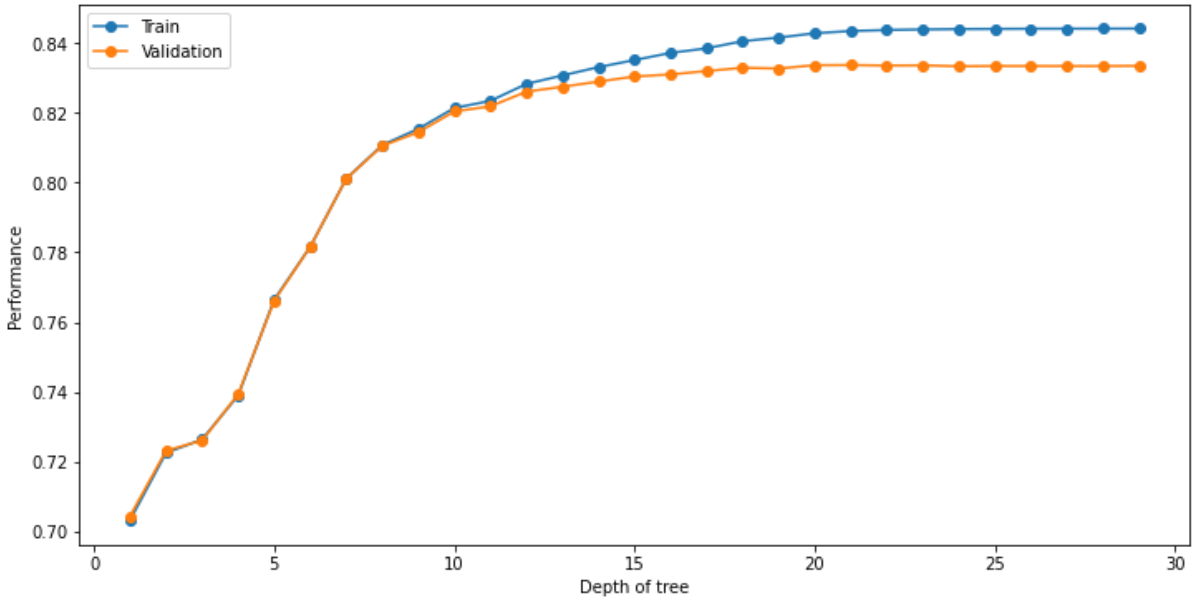
Initial Evaluation & Overfitting

Decision Tree

Dataset	Accuracy	Precision	Recall	F1- score	Support
Train	84.41	0.85	0.22	0.35	3578
		0.73	0.88	0.80	100088
		1.00	1.00	1.00	119881
		0.76	0.56	0.64	70561
		0.32	0.08	0.13	921

Validation	83.33%	0.72	0.88	0.79	24912
		0.99	0.99	0.99	30119
		0.74	0.54	0.62	17576

Overfitting was measured by using various max_depths and seeing where accuracy on the training dataset starts to diverge from the accuracy on the validation dataset:



From the above graph we can see that the two accuracies start to diverge from an approximate max_depth of 15.

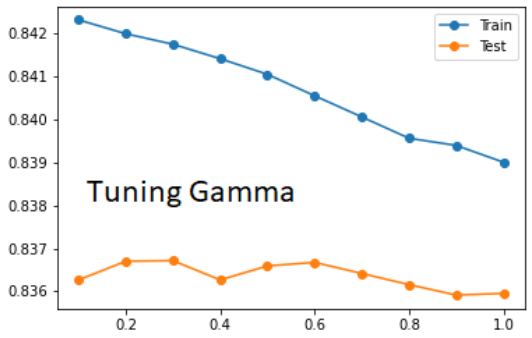
KNN

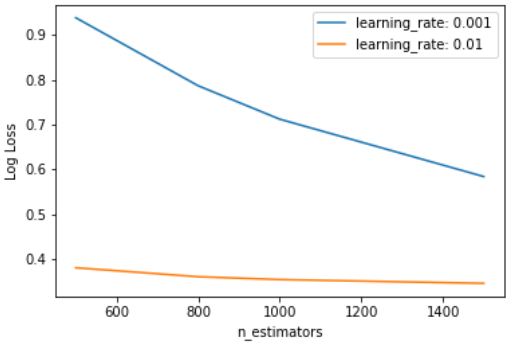
Dataset	Accuracy	Precision	Recall	F1- score	Support
Train	82.47%	0.62	0.04	0.08	3578
		0.72	0.84	0.78	100088
		0.99	0.99	0.99	119881
		0.69	0.55	0.55	70561
Validation	82.29%	0.52	0.03	0.06	921
		0.72	0.84	0.78	24912
		0.98	0.99	0.99	30119
		0.69	0.55	0.61	17576

Overfitting was measured by using various n_neighbors and seeing where accuracy on the training dataset starts to diverge from the accuracy on the validation dataset:

XGBoost

Overfitting seems to happen for xgboost when the trees are grown deep and when the number of gradient boosted trees is increased. The n_stimators vs log loss shows that as n_estimators are increased, the model is overfitting and performing worse on the validation data set compared to how well it predicts the training set. A larger gamma is also responsible for over fitting. Since the model considers more leaves, there is a higher chance of overfitting when there are too many leaves.





Similarly to the other models the group has explored, this model is also a bad predictor for deceased.

```
[[   492   2144    119    823]
 [     0  87747      3  12338]
 [    18      1 119425    437]
 [    14  30377    312  39858]]
```

The confusion matrix shows that the model is mistaking deaths for hospitalized and recovered. There are lots of false positives when predicting for the deceased.
The model uses ‘recovered’ and ‘case-fatality_ratio’ columns as the two most important features while ‘age’ and ‘date_confirmation’ are the two features that the model considers the least. They can be seen as noisy data to the model.

Hyperparameter tuning

A. Decision Tree Classifier

Below are the combinations of hyperparameters for the decision tree ordered in terms of best results for ‘recall on deceased’.

Hyperparameters	F1-Score on ‘deceased’	Recall on ‘deceased’	Overall Accuracy	Overall Recall
max_depth = 20 criterion = ‘gini’ splitter = ‘best’	**0.14**	**0.0783**	**0.8322**	**0.8322**

GridSearchCV was used to perform 5-fold cross validation on the several combinations of parameters to get the results above.

We can see that the best parameters include a max_depth = 20, criterion = ‘gini’ and splitter = ‘best’. Although a lot of the combinations of hyperparameters have the same F1-score on deceased, this combination has the highest recall on deceased.

B. XGBoost Classifier

Hyperparamter tuning done using XGBClassifier from xgboost library.
The training set was used to tune the parameters

Hyperparameters	F1-Score on ‘deceased’	Recall on ‘deceased’	Overall Accuracy	Overall Recall
n_estimators=2500 learning_rate=0.01 Max_depth=12 gamma=0.1	**0.26**	**0.95**	**0.84**	**0.86**

The best parameters are n_estimators=2500, learning_rate=0.0, Max_depth=12, gamma=0.1 giving the best f1-score on deceased but it's also tied with many other models, tie breaker broken by recall on deceased and overall accuracy and overall recall.

The technique used for hyperparameter tuning is gridsearchCV since we want to do 5-fold cross validation on many combinations of hyperparameters.

C. KNN Classifier

Below table shows the top 3 best results on **Recall on Deceased**. It can be observed that when we set the hyperparameters for KNN model to n_neighbors=3, algorithm=’brute’, and weight=’uniform’, we get the best result for recall on deceased. Similarly, setting hyperparameters to n-neighbors=3, algorithm=’ball_tree’ and ‘kd_tree’, and weight=’uniform’, we got the second and third best results for recall on deceased respectively.

Hyperparameters	F1-Score on ‘deceased’	Recall on ‘Deceased’	Overall Accuracy	Overall Recall
n_neighbors = 3 algorithm = ‘brute’ weight = ‘uniform’	**0.09**	**0.129961**	**0.794429**	**0.794429**

In addition to this, when setting hyperparameters to n_neighbors=50, algorithm=’ball_tree, and weight=’distance’ we obtained the best accuracy for our KNN model. For the second and third most accurate KNN model, we set the hyperparameters to n-neighbors=50, algorithm=’auto’ and ‘kd_tree’, and weight=’distance’ respectively. *RandomizedSearchCV* was used to perform 5-fold cross validation on the several combinations of parameters to get the results above.

Results

A. Decision Tree Classifier

Using the decision tree model to predict values on the validation leads to a total accuracy of 0.82 while the total recall is also 0.82. Meanwhile where the model struggles particularly is in the f1- score on ‘deceased’ which is extremely low at 0.14.

	precision	recall	f1-score	support
0.0	0.32	0.09	0.14	921
1.0	0.72	0.88	0.79	24912
2.0	0.99	0.99	0.99	30119
3.0	0.74	0.54	0.62	17576
accuracy			0.83	73528
macro avg	0.69	0.62	0.64	73528
weighted avg	0.83	0.83	0.82	73528

```
[[   80   581    44   216]
 [   39 21922     7  2944]
 [   78    19 29806   216]
 [   51  7830   223  9472]]
```

From the above confusion matrix and classification report we can see that the decision tree misclassifies a lot of values for deceased. This is demonstrated by the low values for f1-score and recall for deceased.

B. XGBoost Classifier

The tuned model trained on the train set and predicting on the validation data set, similarly to the other models is not very good at predicting deceased. The total accuracy is 0.84 and the total recall is 0.84. The f1-score on ‘deceased’ is 0.12 and the recall score on ‘deceased’ is 0.57. From the f1-score on ‘deceased’ the model’s precision on predicting deceased is not very good.
The confusion matrix shows that there are lots of misclassifications for deceased as shown by the f1-score and recall on deceased.

	precision	recall	f1-score	support
0	0.07	0.57	0.12	109
1	0.88	0.73	0.79	30135
2	0.99	0.99	0.99	30129
3	0.55	0.74	0.63	13155
accuracy			0.84	73528
macro avg	0.62	0.76	0.63	73528
weighted avg	0.87	0.84	0.85	73528

```
[[   62   575    46   238]
 [    5 21860     6  3041]
 [   19     3 29903   194]
 [   23  7697   174  9682]]
```

C. KNN Classifier

Using the tuned KNN model to predict values on the validation gives a total accuracy of 0.80 and a total recall of 0.80. The f1-score on deceased is 0.09 and has a recall score of 0.16. This shows that the model’s precision on predicting deceased is not good.

	precision	recall	f1-score	support
0.0	0.06	0.16	0.09	921
1.0	0.74	0.73	0.74	24912
2.0	0.99	0.99	0.99	30119
3.0	0.63	0.58	0.61	17576
accuracy			0.80	73528
macro avg	0.61	0.62	0.61	73528
weighted avg	0.81	0.80	0.80	73528

```
[[  148   399    55   319]
 [ 1206 18292     5  5409]
 [   53    13 29826   227]
 [  922  6094   282 10278]]
```

The above confusion matrix shows that the KNN model misclassifies a lot of the deceased values and thus produces a low f1-score for deceased.

Conclusion

It is apparent from all the models, that they are all poor at predicting deceased. The low recall and f1-score on deceased indicates that the models are not very good at predicting that label. This is further supported by the confusion matrix where there are more false negatives or false positives for deceased then there are true positives. This can be a result from the nature of the data where the classes are closely related to each other.
The XGBoost model is making predictions for the deceased when it is certain while the other two models have a loose criteria for what they label as deceased. This is a result of the bias variance trade off.
For the goal of the project xgboost would be the best since it labels deceased correctly the most out of the other models. It is emphasized that all three models are poor for this task and perhaps another model or method is more suited for the goal of the project.

Prediction on test dataset

Using the 3 tuned models that were built, we used a soft voting classifier to predict the test dataset. This method should reduce the variance but not fix the bias that leads to the prediction of the deceased.

Lessons learnt & Future work

A. Ritvik Shah

   Understanding your dataset is a central issue and primary element of progress. Discovering issues in data collection is significant. Visualizing the dataset can help to distinguish numerous issues. I learnt that data preparation is a very important task, which cannot be ignored. Therefore data understanding and cleaning is crucial as the resulting ML model would not be effective at all if the data is not clean and properly transforme. For the future work I aim at using this skill at the organization where I work, they have huge data sets which they have been collecting since years.

B. Jusung Lee
   I would have liked more time to tune multiple different models to see if any of the other models would perform better for this dataset. It was interesting taking real world data and going through the whole data preprocessing stage instead of being given cleaned data.

C. Alan Thomas

   This is my first machine learning project. This project helped me think outside the normal thinking paradigm. I was able to learn a lot about the various preprocessing

techniques to clean data and remove outliers and apply them into the project. This project has a huge learning curve to newcomers into the field of machine learning as it's a hands on experience on real life dataset. For my future work I plan on taking more Machine Learning courses and hopefully research in the field too.

Contributions

Milestone 1:

- Ritvik Shah - Transformation (1.4), Joining cases and Location Dataset (1.5). Same sections of Report
- Alan Thomas - Dealing with outliers (1.3), Transformation (1.4).
- Jusung Lee - 1.1, 1.2, 1.6

Milestone 2:

- Ritvik Shah - Building, training, tuning and evaluating the decision tree classifier model. Decision tree sections of Report
- Alan Thomas - Building, training, tuning and evaluating the KNN classifier model.
- Jusung Lee - Building, training, tuning and evaluating the xgboost classifier model.and the report

Milestone 3:

- Ritvik Shah - hyperparameter tuning for decision tree, predictions, test set and report
- Alan Thomas - hyperparameter tuning for KNN, predictions, test set and report.
- Jusung Lee - hyperparameter tuning for xgboost, predictions and test set, and report

References

- Sklearn documentation on the specific models

- J. Jordan. (), [Online]. Available: https : / / www. jeremyjordan . me / hyperparameter - tuning/. (accessed: 12.15.2020).

- GeeksForGeeks. (), [Online]. Available: https : / / www. geeksforgeeks . org / hyperparameter - tuning/. (accessed: 12.15.2020).

- Sciket. (), [Online]. Available: https://scikit-learn. org / stable / modules / grid search . html. (accessed: 12.15.2020).