CMPT459 D100 Spring 2024

Special Topics in Data Mining

Course Project Final Report

Colton Blackwell, 301451278
Angie (Yoojin) Lee, 301365768
Mehreen Uzma, 301397070

1. Problem Statement

COVID-19, or coronavirus disease 2019, is caused by the severe acute respiratory syndrome coronavirus 2 (SARS-CoV-2). The contagious nature of COVID-19 has profoundly impacted all areas of the globe. It has led to rapid and widespread virus transmission, causing surges in cases that overwhelm healthcare systems and necessitate stringent public health measures such as lockdowns, travel restrictions, mask mandates, and social distancing guidelines.

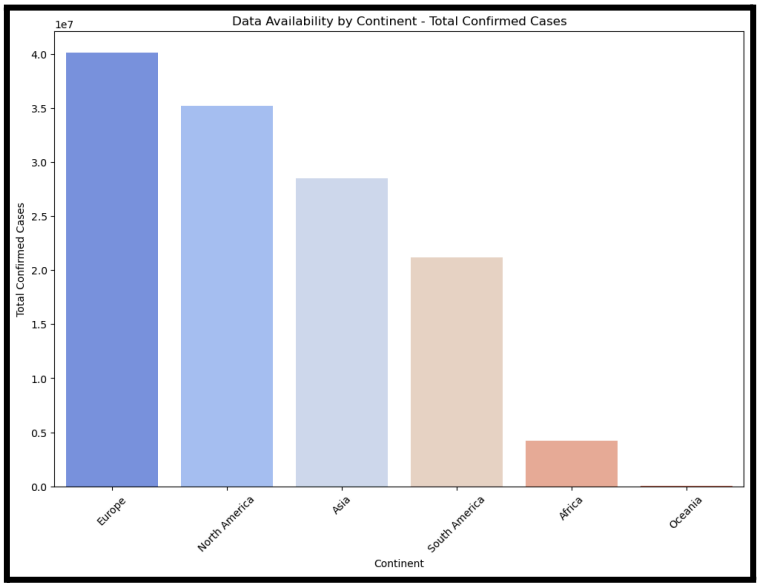
In this data mining project, we aim to explore several widely used machine learning models, optimize various hyperparameters, and utilize the best-tuned models to predict COVID-19 case outcomes. We hope that by predicting the outcome of a patient with COVID-related disease, we can understand the patterns and prevent further spread of this disease.

2. Data visualization

Interpretation of the Bar Plot on Data Availability by Continent:

This bar plot displays the total number of confirmed COVID-19 cases by continent. Europe and North America show the highest data availability, which may indicate a higher number of actual confirmed cases; however, it could also reflect the higher capability of data collection and reporting. Asia exhibits relatively lower data availability in proportion to its population, which could be due to several factors: accessibility to testing, differences in reporting standards, or the efficiency of the systems tracking and reporting confirmed cases.

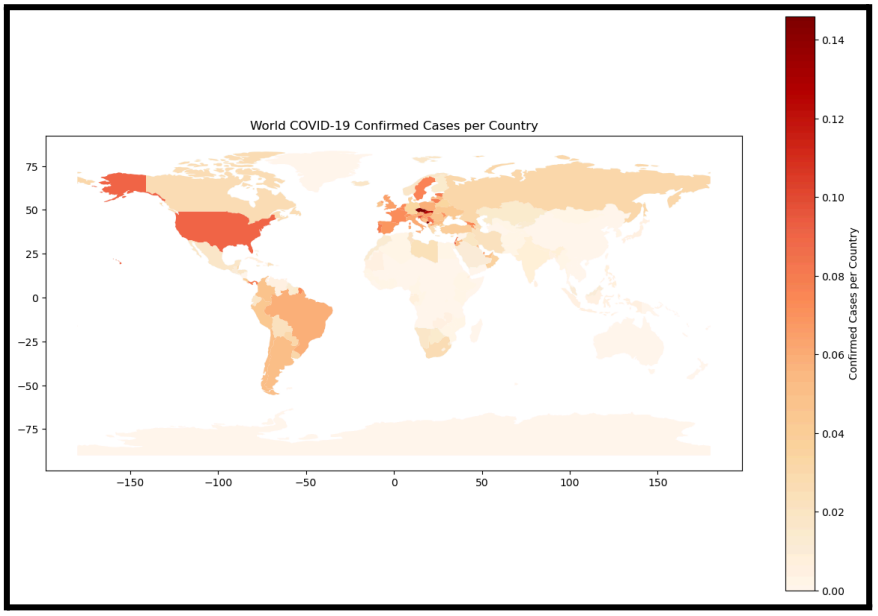
Africa and Oceania have the lowest data availability, which could stem from a lack of testing, differences in reporting systems, or a genuinely lower number of cases relative to their population. Moreover, Oceania's overall lower population might contribute to the lower absolute figures. The color spectrum was used to denote the level of data availability: continents with higher data availability are shown in redder shades, while those with lower availability are in bluer shades.



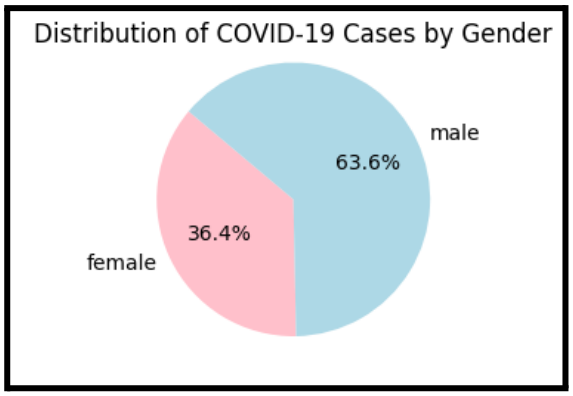
Interpretation of the World Heatmap:

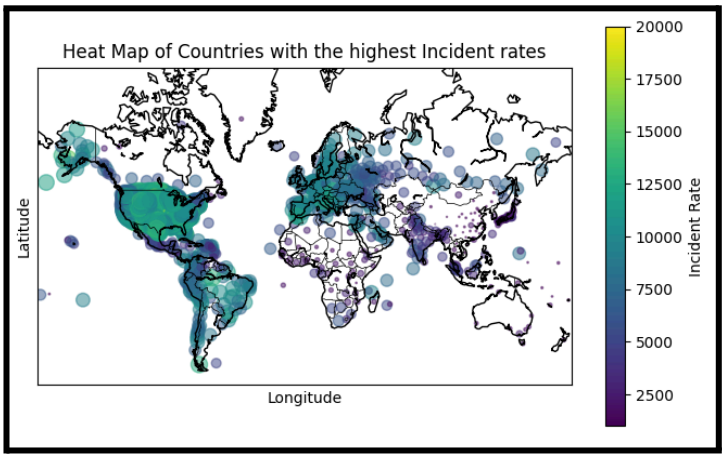
The World Heatmap presents a global map based on the proportion of confirmed COVID-19 cases relative to the countries' populations. This transcends the absolute number of cases, allowing for an assessment of the relative impact of the infection. Several countries in North

America and Europe exhibit high ratios, whereas most nations in Africa and Asia show relatively lower ratios. This provides insights into the testing capabilities and reporting systems of each country, as well as the actual extent of infection spread. The colors vary according to the ratio of confirmed cases in each country. Darker shades indicate higher ratios, while paler shades represent lower ones. This visually compares the impact of COVID-19 across countries.



COVID-19 cases show a significant gender ratio, with 63.6% male cases and 36.4% female cases. The United States and its states dominate the top 15 regions globally for confirmed cases. The MS Zaandam had the highest fatality ratio among confirmed cases due to reported deaths onboard. The United States, Brazil, Mexico, and India lead in death tolls, possibly due to economic challenges and healthcare system limitations. China's surprisingly lower reported cases might be attributed to lower testing rates and stricter lockdowns.





In the data preprocessing phase, we prioritized the management of missing values in critical features for model accuracy. For 'Age', where imputation strategies compromised model accuracy, we chose to drop records with missing values to maintain the quality of our input data, reinforcing 'Age' as a critical predictive feature. Missing values in 'Sex' were filled with -1, anticipating the conversion of categorical data to numerical, allowing these records to be utilized in model training without compromising data integrity.

We determined that preserving the existing records with missing 'date_confirmation' and 'source' would be more practical, as their quantities are negligible, rendering their impact on the analysis

minimal. Furthermore, if these features are deemed non-essential for predictive modeling, they can be excluded or ignored without affecting model performance.

We also implemented stringent cleaning for geospatial consistency, discarding records outside valid longitude and latitude ranges and those without specific province information. Moreover, we ensured 'Age' values were within a plausible range for human lifespan [0,105].

4. Data Preparation

4.1. Feature Selection

Features in testing and training datasets include:
['age', 'sex', 'chronic_disease_binary', 'latitude', 'longitude', 'Deaths', 'Recovered', 'Confirmed']

However, the following categorical features were converted into numerical representations:
Age, sex, country, Chronic_disease_binary, Outcome_group

4.2. Mapping the features

During the data preprocessing phase, transforming various categorical features into numerical form is an essential step for model training since machine learning models cannot process categorical data in string format. In this process, features such as 'sex', 'chronic_disease_binary', 'province', among others, and the response variable 'outcome_group' were mapped to numerical values. The mapping for each feature is as follows:

Sex: Sex was transformed into numerical data by mapping it to 0 or 1. This simple yet effective mapping allows the model to distinguish genders.

Chronic Disease Binary: The binary feature chronic disease presence was converted into numerical data by mapping ‘true’ and ‘false’ to 1 or 0, respectively. This is included in the classification model since the presence of a chronic disease can be a significant factor affecting patient outcomes.

Province: Province was mapped to unique integer values. Each province name is assigned a unique number, allowing the model to learn regional characteristics. This is because geographic location can provide valuable information for outcome prediction. Similarly, country was also mapped.

Outcome_group_mapped: The response variable 'outcome_group' was also mapped consistently to numeric values to maintain consistency and comparability. Specifically, 'deceased' was mapped to 0, 'hospitalized' to 1, and 'nonhospitalized' to 2. This standardized approach ensures that predictive modeling is aligned and can be benchmarked accurately.

This mapping implementation makes the models learn patterns through numbers instead of text data, leading to more efficient predictions. Additionally, pandas' categorical data type is more memory-efficient compared to string types, which helps optimize memory usage and improve performance and scalability when dealing with large datasets.

4.3 Balancing the classes in the training dataset

We balanced the dataset via **oversampling**. Undersampling involves the removal of data points from the majority class, which may result in the loss of valuable information contained within those instances. On the contrary, oversampling focuses on the minority class by

generating synthetic samples, ensuring that all available information from the minority class is retained. This approach not only helps in maintaining the variability and richness of the minority class, crucial for building robust and generalizable models, but also contributes to the model's resilience against noise and outliers present in the minority class. The synthetic samples created during oversampling aid in capturing the true underlying distribution of the data, enhancing the model's ability to learn and generalize effectively.

	Number of Cases	
outcome_group	**Before Balancing**	**After Balancing**
hospitalized	13241	13241
nonhospitalized	2974	13241
deceased	997	13241

5. Classification models

5.1 Building Models and Hyperparameter Tuning

5.1.1 Building Models

KNN

The K-Nearest Neighbors (KNN) classifier model was selected due to its ease of implementation, simplicity, and effectiveness. One of its strengths lies in its ability to capture complex decision boundaries that are not necessarily linear, making it suitable for scenarios with nonlinear relationships between features and classes. Additionally, tuning the parameter in KNN is straightforward and does not demand extensive computational resources. Moreover, KNN exhibits robustness to outliers in the data since it considers multiple neighbors rather than depending solely on a single data point.

Random Forest

The Random Forest classifier ensemble nature reduces overfitting, provides feature importance insights, and ensures resilience to outliers and noise, contributing to reliable and interpretable predictions. The model's parallel training capability makes it efficient for large datasets, while its non-parametric approach and lack of assumptions about data distribution enhance its versatility across various classification tasks.

Categorical Naive Bayes (CatNB)

The Categorical Naive Bayes classifier model’s simplicity and computational efficiency make it suitable for large datasets with categorical features, offering quick training and prediction times. The assumption of feature independence can still yield decent results and provide insights into the importance of different categorical features in the classification process.

5.1.2 Hyperparameter Tuning

Our decided K value was 4. This strikes a middle ground between computational efficiency and the accuracy of bias/variance estimation in cross-validation. Higher *K* values demand greater

computational resources but yield more precise assessments of model performance. Conversely, lower *K* values lessen computational burdens but might introduce increased variance in performance evaluations. For moderately sized datasets, opting for *K* =4 strikes an optimal balance, ensuring each fold contains sufficient data for robust model training and validation without overwhelming computational demands.

We decided on GridSearchCV for hyperparameter tuning because of its use of a specified grid of hyperparameter values, ensuring no combination is disregarded. As with our case, this method is efficient when dealing with limited hyperparameter spaces, guaranteeing a comprehensive assessment without overlooking any combinations within the specified grid. We opted against using RandomizedSearchCV because it might not fully investigate the entire hyperparameter space, potentially overlooking optimal combinations.

KNN

The hyperparameter range for K in KNN was set as range(3,7) to mitigate overfitting/underfitting, as to be discussed in Section 5.2. Notably, the choice of distance metric was not a hyperparameter since it was decided that the KNN class was to already integrate the Euclidean distance measurement within its structure.

Random Forest

During the hyperparameter tuning process for the Random Forest model, I explored various hyperparameter values including the **number of trees (n_estimators)**, the **maximum depth of the trees (max_depth)**, and the **bootstrapping option (bootstrap)**. These parameters are crucial as they significantly influence the performance of the Random Forest model.

Number of Trees (n_estimators): This parameter directly impacts the model's performance and complexity. Using more trees can make the model's predictions more stable but at the cost of increased computational expense. I experimented with values of 100, 200, and 300 to find the optimal number of trees.

Maximum Depth of the Trees (max_depth): This limits how deep the trees can grow. Deeper trees can model more complex patterns and fit the training data better but also raise the risk of overfitting. I tested values of None (no limit), 10, 20, and 30 to determine the appropriate depth.

Bootstrapping Option (bootstrap): Bootstrapping refers to the method of selecting samples with replacement when building trees, which can increase model diversity and help prevent overfitting. I evaluated both the use (True) and non-use (False) of bootstrap sampling.

Categorical Naive Bayes Classifier

Hyperparameter tuning for Categorical Naive Bayes ranges from 1 to 10^-9, covering different smoothing levels. Starting at 1.0 with no smoothing, it gradually introduces regularization through values like 0.811 and 0.657, reducing sensitivity to outliers. Lower values like 0.432 offer moderate smoothing, balancing model flexibility and overfitting risks. Values of 10^−3 and 10^−6 provide significant to very small smoothing, beneficial for complex datasets but require caution for numerical instability. Extreme regularization at 10^−9 is also explored. This range helps in choosing the right smoothing parameter based on the dataset characteristics.

Best Performing hyperParameters per 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
KNN	'K' = 4	0.902337	0.866244	0.901693
Random Forest	'bootstrap': False, 'max_depth': None, 'n_estimators': 300	0.926390	0.898055	0.925766
Categorical Naive Bayes	'Var_smoothing' = 0.028480	0.59021	0.187803	0.6710931

5.2. Overfitting

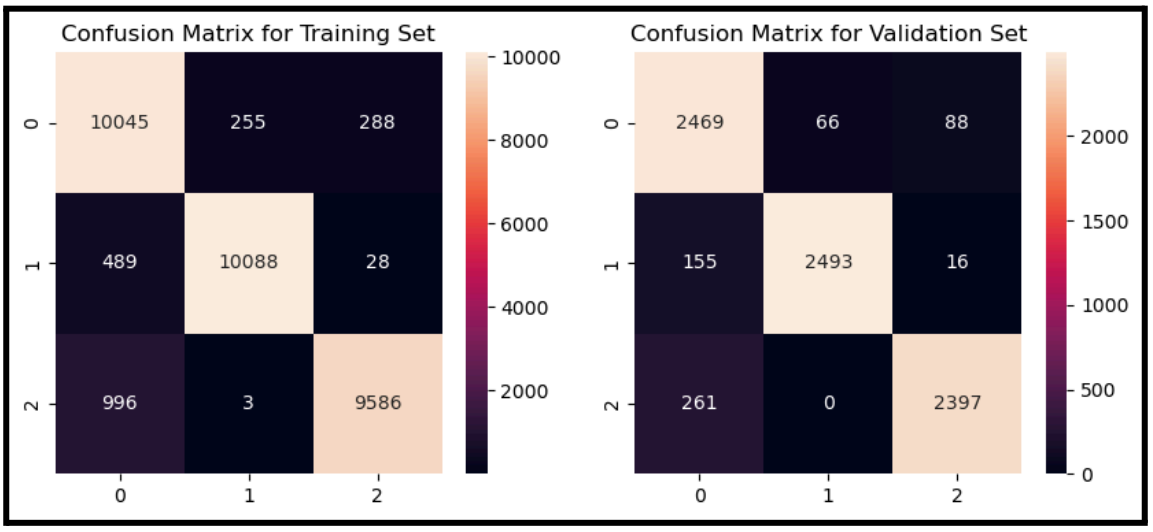
KNN

We checked for overfitting on the KNN model by measuring the mean error per K. A lower K value when it comes to KNN contributes to overfitting as the model becomes too sensitive to small fluctuations in the data, essentially memorizing the training set rather than learning general patterns. On the other hand, a higher K value can lead to underfitting, where the model oversimplifies and fails to capture the underlying complexity of the data. We observed that the pattern below displaying the error rate per K value, K=4 contributes to the best K value as it strikes a balance between overfitting and underfitting, making the model generalize well to new data while still capturing meaningful patterns from the training set.

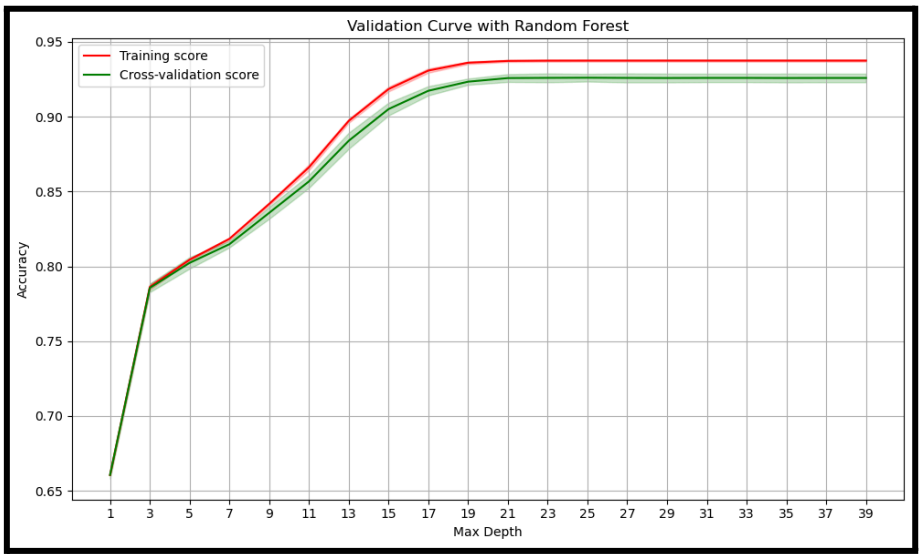
For GridSearchCV, we required the parameter grid to contain acceptable K values in range(3,12).

Random Forest

Confusion Matrix Analysis: The confusion matrices for the training and validation sets were compared to assess the predictive accuracy across classes. The similarity in patterns between the training and validation confusion matrices indicates effective generalization. Despite some instances of misclassification in the validation set, this is a common occurrence in complex models like Random Forest.



Validation Curve: A validation curve was utilized to evaluate performance variations with increasing 'max_depth' hyperparameter values. The curve depicts training and cross-validation accuracy scores, highlighting that beyond a certain 'max_depth', scores stabilize, suggesting no significant gain from deeper trees and mitigating the risk of overfitting. The convergence of training and cross-validation scores indicates that the model is appropriately tuned within the explored 'max_depth' range.



Naive Bayes Classifier

By comparing the mean performance metrics between training and validation sets and analyzing the learning curves, we can identify any signs of overfitting. If the model performs significantly better on the training set compared to the validation set or if there's a large gap between the training and validation curves in the learning curves plots, it suggests overfitting.

5.3. Comparative Study

Given outcome_group_mapping = {'deceased': 0, 'hospitalized': 1, 'nonhospitalized': 2}

Model	True Positives for ‘deceased’	True Positives for ‘hospitalized’	True Positives for Class ‘nonhospitalized’	Accuracy
KNN	2276	2467	2396	0.90396
Random Forest	2469	2493	2397	0.92624
Categorical Naive Bayes	1282	2640	2414	0.83931

The accuracy of KNN and Random Forest models shows similarity, whereas the accuracy of the Categorical Naive Bayes classifier slightly decreases. Interestingly, this classifier struggles with accurately recognizing the 'deceased' class labels.

In this Study, the Random Forest classifier emerges as the top-performing model within this specific context, demonstrating the highest number of true positives and the highest accuracy. This outcome is likely attributed to the fact that Random Forests utilize an ensemble learning technique. By combining multiple decision trees, this approach effectively mitigates overfitting and enhances generalization capabilities.

7. Conclusion

We carefully prepared our data, selecting relevant features and balancing class distributions for accurate model comparisons. After optimizing our datasets, we constructed and fine-tuned classification models using advanced techniques like hyperparameter tuning. Rigorous checks for overfitting and comparative analyses helped us identify the most effective model. Our best-tuned model successfully predicted outcomes on the test dataset, demonstrating the success of our approach in creating a robust classification solution.

8. Lessons learned and future work

Spending time on data cleaning and preprocessing is crucial as it ensures the model isn't learning from noisy or incorrect data. Handling missing values, outliers, and encoding categorical variables correctly helps in building a more robust and accurate model. Handling outliers can be improved further by exploring advanced outlier detection algorithms such as Isolation Forest to identify and handle outliers more effectively.

The feature scaling step is crucial for improving the performance and stability of classifier models, as it prevents the model from being influenced disproportionately by features with larger magnitudes. It also helps algorithms converge faster during training and makes them more robust to outliers and varying feature scales across different datasets.

9. References

Refer to page 9.

10. Contribution

Our group has demonstrated strong teamwork by supporting each other, addressing questions, troubleshooting, and solving problems collaboratively. While we advocate for equal recognition, we acknowledge that Colton and Angie contributed significantly more effort and, therefore, deserve additional marks for their contributions.

Colton - Report/Steps 1 - 8/ KNN model and hyperparameter tuning

Angie - Implementation of Steps 1 - 9 (including Random Forest Model, hyperparameter tuning and prediction on the test set), Report writing

Mehreen - Step 6: Implementation of the Naive Bayes Classifier, Step 7: Overfitting, reviewing report, reviewing code, troubleshooting and helping teammates.

References

Awan, Abid Ali, and Avinash Navlani. “Naive Bayes Classifier Tutorial: With Python Scikit-Learn.” DataCamp, DataCamp, 3 Mar. 2023, www.datacamp.com/tutorial/naive-bayes-scikit-learn.

GfG. (2024, January 25). *K-Nearest Neighbor(KNN) algorithm*. GeeksforGeeks. https://www.geeksforgeeks.org/k-nearest-neighbours/

“Naive Bayes Classifier in Python (from Scratch!).” YouTube, YouTube, 26 Feb. 2021, www.youtube.com/watch?v=3I8oX3OUL6I&t=491s.

Zwinck, J. (2016, April 1). *Pandas: Convert categories to numbers*. Stack Overflow. https://stackoverflow.com/questions/38088652/pandas-convert-categories-to-numbers

G. (2024, February 22). *Random Forest Algorithm in Machine Learning*. GeeksforGeeks. https://www.geeksforgeeks.org/random-forest-algorithm-in-machine-learning/

Koehrsen, W. (2020, August 18). *Random Forest Simple Explanation - Will Koehrsen - Medium*. Medium. https://williamkoehrsen.medium.com/random-forest-simple-explanation-377895a60d2d