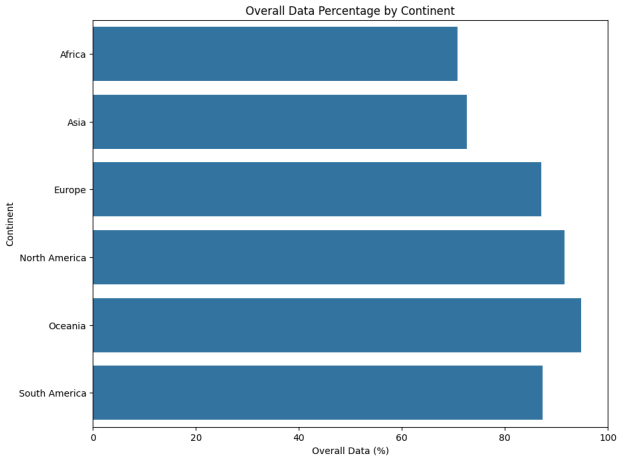
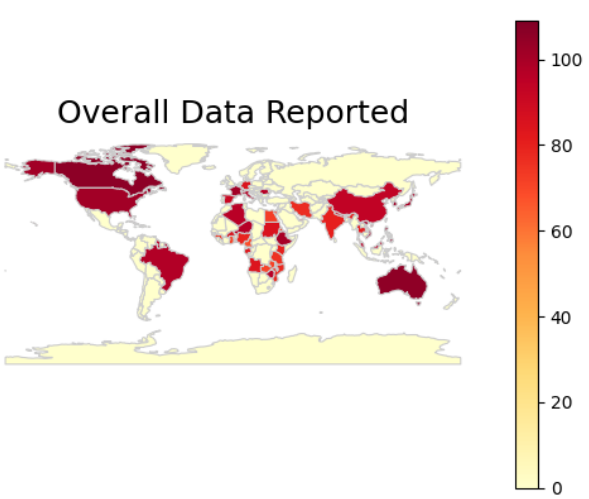
Problem Statement

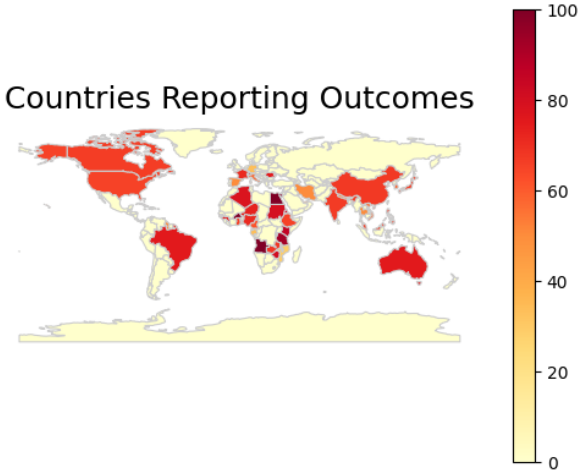
Data Visualization - Task 1

From the data being all continents held at least 65% of complete data that could be properly evaluated and used to train our models.With Africa and Asia providing the least amount of data.
This is not particularly surprising but certainly could be considered anomalous as they have considerably higher populations than other continents and so percentages of missing data represent much larger amounts of records in these regions. We know that during covid China, which holds nearly 20% of the world's population was particularly strict with both its covid reporting policies and social distancing practices, making the data that was recorded even more essential as it widely suspected that not all data was reported from regions of that side of globe in general.



We saw more consistency of data being reported in western countries, Specifically Canada, the US and Australia. It can be inferred that this may have to do with western powers leading the initiative to find the vaccine in the first place and having harder issues maintaining lockdows and so statistics were used to incentivise people to remain indoors. Overall however no countries were particularly keen on reporting their outcomes, whether this had to do with not wanting to scare the public or with uncertainty of results is still to be discovered.





Data Preprocessing - Task 2

We performed various operations such as cleaning, transformation, integration, reduction, and feature extraction during Data Preprocessing to merge two datasets. Initially, we reviewed the data column-wise to identify missing values, maximum and minimum values, and frequently occurring values for both the training dataset and the location dataset.

1. The country names in both the training and location datasets were standardized to follow RestCountries.com conventions, correcting 15 non-standard country names, such as "Korea, South" to "South Korea" and "Congo (Kinshasa)" to "Democratic Republic of the Congo." Records for cruise ships like "Diamond Princess" and "MS Zaandam" were also identified as invalid country records. An exception was made for Taiwan, treating it as a separate country despite naming conventions that vary internationally.
2. We combined and transformed the information from the Outcome and Outcome_Group columns from the training datasets. Based on the provided dataset for task 3 onwards, we manually mapped all the unique values of outcome to three different groups: "Hospitalized," "Deceased," and "Discharged," and updated the outcome_group column and dropped the outcome column.
3. We renamed the latitude and longitude columns from the training dataset to case_lats and case_longs to avoid conflict with the information in the location dataset. Additionally, we rounded all numerical values to 6 decimal places.
4. Missing values in additional information columns were set to "Not Available" and in the Confirmed, Deaths, Active, and Recovered columns to zero. An "Expected Mortality Rate" column was added to the dataset using the following formula:
    a. Expected Mortality Rate = Deaths / Confirmed Cases
5. The location dataset contained duplicate records for the same province and country key, each representing a different city. To address this, we merge these duplicates into a single record by aggregating confirmed, death, active, and recovered data. The most recent "last update" value was retained. Post-aggregation, incident rate, case fatality ratio, and expected mortality rate were recalculated using the updated figures.
6. Invalid values in the location dataset were addressed, and a list of unique countries was created. For each country, an API request was made to obtain its population, latitude, and longitude. With this data, the incident rate for records missing information was calculated using.
    a. Incident Rate = (Confirmed Cases / Population) * 100000
7. The missing latitude and longitude values were also added.
8. In the final processed dataset, we dropped the Country_Region and Province_State columns as they came from the Location dataset and were duplicates of the country and province columns from the Training Dataset. We also dropped the combined key as it was a derived column with no extra information.

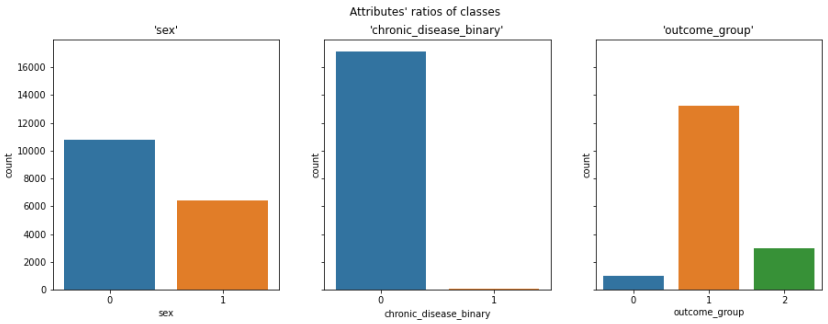
Data Preparation - Task 3, 4, 5

3. Feature selection: Report the final features that are used.
    - Final list of features used: age, sex, province, country, latitude, longitude, chronic_disease_binary, confirmed, deaths, recovered, active, incident_rate, case_fatality_ratio.

	column name	column types	NULL count
0	age	<class 'numpy.float64'>	0
1	sex	<class 'numpy.int64'>	0
2	province	<class 'numpy.int64'>	0
3	country	<class 'numpy.int64'>	0
4	latitude	<class 'numpy.float64'>	0
5	longitude	<class 'numpy.float64'>	0
6	date_confirmation	<class 'str'>	0
7	chronic_disease_binary	<class 'numpy.int64'>	0
8	Confirmed	<class 'numpy.float64'>	0
9	Deaths	<class 'numpy.float64'>	0
10	Recovered	<class 'numpy.float64'>	0
11	Active	<class 'numpy.float64'>	0
12	Incident_Rate	<class 'numpy.float64'>	0
13	Case_Fatality_Ratio	<class 'numpy.float64'>	0
14	outcome_group	<class 'numpy.int64'>	0

4. Mapping the features:
    - There are two groups of features that are mapped: binary and nominal. Both sex and chronic disease are binary so values in these columns are mapped to 0 and 1. For the two nominal features in the other group, each of their unique values is assigned a number between 0 and n, n is the number of unique values in the feature. A province needs to be assigned as a unique combination with their country even as NaN value.
    - The reason for binary mapping is self-explanatory.
    - While nominal attributes can be mapped using hot-one-encoder, direct numerical mapping has shown to be optimal and simple. With no country values missing and all of which are independent, it can be mapped directly without issue. Provinces, however, require a different solution. Since no country other than India has their provinces included in the dataset, we have to deal with both the missing value and how to map them. The solution was to create a unique combination for each province-country pair. This way, each value and row that it's in can be differentiated by the models.

5. Balancing the classes in the training dataset:
    - Main approach for class balancing was oversampling the minority classes. We considered undersampling in the case of the amount of data entries averaging over 100K. As a technique for oversampling we used a variant of the popular oversampling technique SMOTE. SMOTE allows the creation of synthetic entries for minority classes based on the original dataset examples. SMOTENC could be used while the dataset contains both categorical and numerical data types [2]. However, there are limitations to all techniques, and SMOTENC could not be applied to data attributes with classes that contain only 1 entry belonging to it. After discussing with the

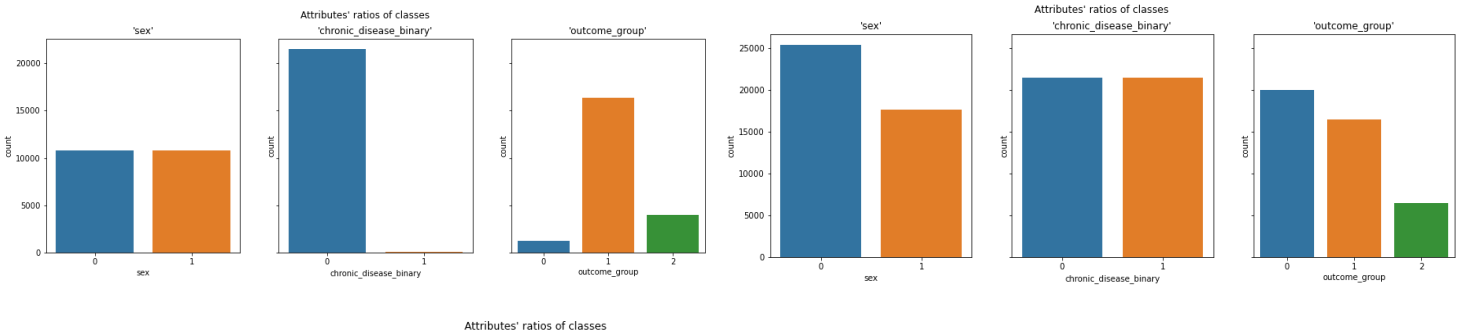


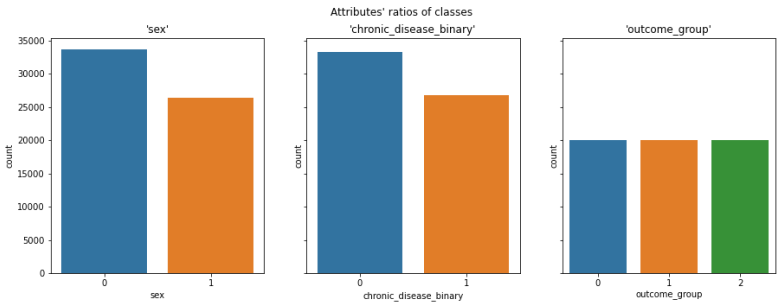
team, we decided to apply SMOTENC to features ['sex', 'chronic_disease_binary', 'outcome_group'].

- Following figure portrays the distribution of values in these features before oversampling. With overall data shape being (17212, 15) and class distributions before oversampling:

```
column 'sex':
0: 10799 (62.74%)
1: 6413 (37.26%)

column 'outcome_group':
1: 13241 (76.93%)
2: 2974 (17.28%)
0: 997 (5.79%)

column 'chronic_disease_binary':
0: 17131 (99.53%)
1: 81 (0.47%)
```

- Following figures show class distributions after each SMOTENC application. SMOTENC is being used on each of the selected features and thus oversampling the minority classes in these attributes to be equal to the majority class. It greatly increases the training set's size, but to a reasonable amount. Final data shape is (60096,15) and resulting in the final distributions:





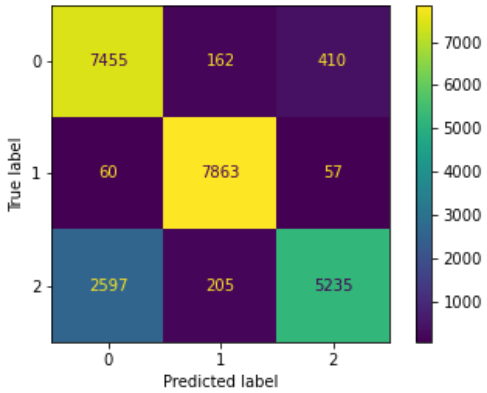

```
column 'sex':
0: 33671 (56.03%)
1: 26425 (43.97%)

column 'outcome_group':
1: 20032 (33.33%)
2: 20032 (33.33%)
0: 20032 (33.33%)

column 'chronic_disease_binary':
0: 33319 (55.44%)
1: 26777 (44.56%)
```

Classification Models - Task 6, 7, 8

6. Building models and hyperparameter tuning: For each model used, explain why you selected it over other models.

a. Logistic Regression

- Since all the features are either numerical or mapped as numbers, the dataset would be ready for logistic regression with some scaling. Even though logistic regression is usually used for binary classification, it performs relatively well on the train set with virtually no overfitting.
- Gridsearch is used for tuning C value and penalty. These two hyperparameters have the most potential to affect the model's performance, specifically overfitting. Since our models are

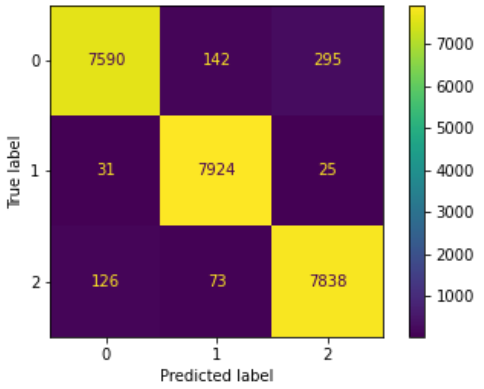


meant to be tested on unlabelled data, overfitting is crucial to the model's performance.

- Liblinear is chosen because the dataset we're working on is relatively small. It uses the one-vs-all schemes where it splits the classes into smaller binary problems to adapt to multiclass problems.
- L1 and L2 are supported by Liblinear and test for overfitting.
- C is from values that are logarithmically spaced between 10^(-4) and 10^4. A high C is more trusting of the training data while low C is not. With a wide range of C values, we can see which is more optimal.

b. Random Forest

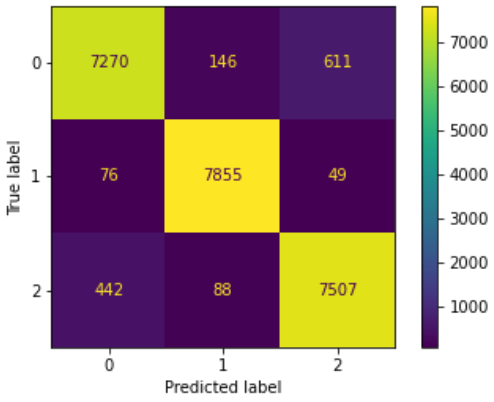
- The random forest classifier is known for delivering high performance. It also further enforces feature importance adding an additional layer of complexity to task 3, lastly it is known to be robust to overfitting making issues outlined in task 7 easier to avoid.



- Hyper parameter tuning is done through randomized search cross validation, I thought this to be the right choice as the hyper parameter space was quite large and Randomized Search CV is known to perform well in these situations. I wanted to make sure that I could capture a wide range of complexity. This was accomplished by creating a larger range for both n_estimators and max_depth. To avoid overfitting I had also set relatively high ranges for both min_samples_split and min_samples_leaf to ensure a good balance between bias and variance.

c. K-nearest neighbours

- The K-nearest neighbours algorithm is commonly used for classification problems and requires non-noisy data sets [3]. Considering that our training dataset is already balanced, it makes K-nearest neighbours to be a perfect candidate for our classification problem. Moreover, its performance to speed ration later on shows great potential.

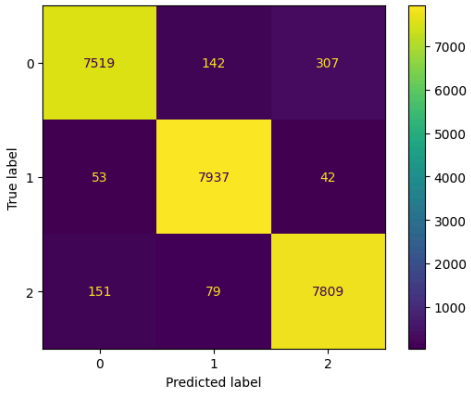


- GridSearchCV is chosen for the hyper parameter tuning of n-neighbours. With the range of (2, 30) with increments of 1, common k value is 3. In addition, in comparison to other models KNN is taking a fracture of the time (~5 minutes) it takes other models to hyper parameter tune (~60 minutes per model) with the same and sometimes better performance results.

d. Gradient Boosting Model

- The dataset's numerical nature suits the Gradient Boosting Classifier, adept at capturing complex patterns through sequential tree building. This method excels in both bias and variance reduction, essential for the dataset's varied features and target outcomes.

- Hyperparameter tuning focused on n_estimators, max_depth, learning_rate, and subsample via GridSearchCV, crucial for balancing model accuracy and generalization. The choice of parameters aimed at fine-tuning the model's learning capacity and fit to prevent overfitting.



- A 5-fold cross-validation strategy was employed, balancing thorough evaluation with computational efficiency. This approach ensures the model's robustness and its performance across different data segments, critical for our multi-class classification task. The parameter grid spanned n_estimators from 500 to 1000, reflecting the model's complexity and adaptability; max_depth at 3 and 2, to control overfitting; learning_rate values of 0.1 and 0.4, determining the step size in corrections; and subsample rates of 0.8 and 0.9, to manage sample variance and bias.

e. K-Fold Cross_validation

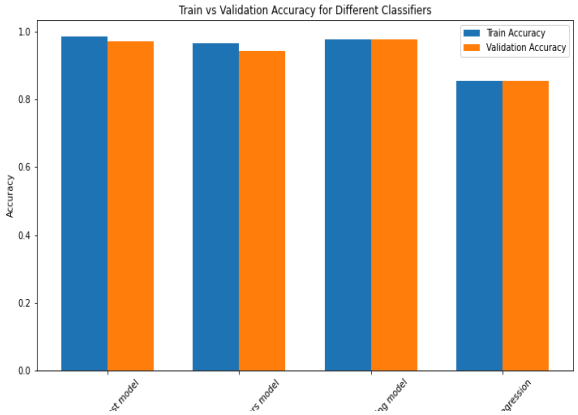
- After experimenting with the value of k and not finding many signs of either high variance or bias, we decided to settle on k = 5 to achieve performance consistency and time efficiency.
- Using best hyper-parameters and corresponding models, we got following statistics on F1-scores and accuracy from 5-fold cross-validation:

	Model	mean macro F1 scores	mean F1 scores over 'deceased' class	mean overall accuracy
0	Logistics Regression	0.8416138596841696	0.5431568406316388	0.8505862093413274
1	Random Forest	0.9668401476172089	0.6464172046523716	0.9679082830023891
2	K-Nearest Neighbors	0.9402564394441203	0.6241336503853276	0.941139743650796
3	Gradient Boosting	0.9583390177239026	0.6529749314549247	0.9592405299819754

	Model	mean F1 scores over 'deceased' class	mean F1 scores over 'hospitalized' class	mean F1 scores over 'non-hospitalized' class
0	Logistics Regression	0.5431568406316388	0.6889835603991659	0.6631658172800412
1	Random Forest	0.6464172046523716	0.7098471359548785	0.9429252580496678
2	K-Nearest Neighbors	0.6241336503853276	0.6809919237693676	0.8632493927724524
3	Gradient Boosting	0.6529749314549247	0.7061096070811167	0.9137375970873622

7. Overfitting: How did you check for overfitting? Explain using plots and/or evaluation metrics.

- To check for overfitting, each model is used to predict both the train and validation set and compared based on their accuracy.
- As the training dataset has been balanced and the features were carefully chosen, the models' performances on this set are expected to be similar to the unlabelled set.
- From the result, all models exhibit little overfitting.
- For a pattern of overfitting, each classifier should exhibit a clear bias when predicting its



training set instead of validation set.While random forest and K-Nearest Neighbors have an approximately .02 difference between the train and validation sets, Gradient Boosting and Logistic Regression have even less than that at approximately 0.001.

- Even though Random Forest has the highest accuracy on the train set, nearing 1, its performance is consistent across the sets. Similar pattern is observed in K-nearest Neighbors. For the other classifiers, their performances, while not as high as the former two, exhibit little difference in their accuracy, indicating that they have little potential for overfitting when used on an unlabelled set.
- Hyperparameter tuning is another metric for limiting overfitting. Grid search is performed on Gradient Boosting, K-nearest Neighbors and Logistic Regression. Randomized search was used for Random Forest. Logistic Regression was searched for a total of 200 fits, Gradient Boosting 80 fits and K-nearest Neighbors 140 fits. The highest score classifier is chosen from these fits.

Task 8:

Our team worked on four different models: Logistic Regression, Random Forest, K-Nearest Neighbors, and Gradient Boosting. It is evident from the results that Random Forest had the highest scores across all metrics without showcasing any significant overfitting.

Also, as the dataset was numerical, tree-based models perform better since they are inherently suited for handling numerical features and do not require feature scaling [1]. These models can capture non-linear relationships and interactions between features effectively. Our team's analysis revealed that while Logistic Regression offers simplicity and efficiency, making it a good baseline with adequate scaling, its binary nature required extension through one-vs-all schemes, potentially complicating multi-class problems. Random Forest stood out in terms of performance and robustness against overfitting, owing to its ensemble approach and intrinsic handling of feature importance, although it can be computationally intensive during hyperparameter tuning. On the other hand, Gradient Boosting, another tree based model performed well, it was slightly behind Random Forest on overall accuracy.

Task 9:

There is not too much to comment on with regards to the results produced, there were no errors or struggles when we implemented this and it surprisingly worked on its first run.

Task 10:

Although its performance was impressive, the Grid search fit model does not seem to be worth the time it takes to fit, considering Random Forest takes less time to train and performs better.

Surprisingly little data was reported on covid across the globe when compared to western countries which leaves the brain to wonder what the response to covid felt like at other parts of the globe

Conclusion

In summary, this report addressed crucial tasks in developing classification models for predicting COVID related medical outcomes. We selected features, where categorical features and outcome groups were mapped to numeric values for compatibility with machine learning algorithms. Balancing classes in the training dataset improved models' robustness, with original and final class distributions presented for comparison. We employed various classification models, selected based on suitability and performance. Hyperparameter tuning was conducted using K-fold cross-validation and appropriate search methods. Overfitting was assessed through several validation procedures. A comparative study evaluated model performances, identifying the best-performing model for our goal, which resulted in the Random Forest Classification model. Finally, we made predictions for the test dataset with the best performing model. This overall approach ensures reliable classification for predicting (medical) outcomes.

Lessons Learnt and future work

In this project, feature mapping and management of class imbalance proved crucial for boosting predictive accuracy. Careful validation procedures and a comparative study of model performances helped in selecting the most effective classification models. To further improve future projects, exploring advanced feature engineering techniques, ensemble methods, and thorough feature importance analysis could enhance model performance and interpretability, providing greater accuracy in classification problems.

References

[1] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on tabular data?," arXiv:2207.08815 [cs, stat], Jul. 2022, Available: https://arxiv.org/abs/2207.08815

[2] Mukherjee, M., & Khushi, M. (2021). SMOTE-ENC: A novel SMOTE-based method to generate synthetic data for nominal and continuous features. Retrieved from https://arxiv.org/ftp/arxiv/papers/2103/2103.07612.pdf

[3] Neptune.ai. KNN Algorithm: Explanation, Opportunities, Limitations. Retrieved from https://neptune.ai/blog/knn-algorithm-explanation-opportunities-limitations

[4] Devore, J. L. (2012). *Probability and statistics for engineering and the sciences / Jay L. Devore.* (8th ed.). Brooks/Cole.

Contribution

- Tristian Labanowich 301422226: Parts 1, 6(RF),9 as well as report composition and creation of various graphics included.
- Long Duong (301367862): Task 3,4,6(LR),7.
- Manvir Singh Heer (301448411): Task 2, partially Task 4, and Task 6 (Gradient Boosting) and Task 8.
- Dmitrii Beliaev, 301435234: Task 5 (including graphs and screenshots of class distributions), Task 6 (KNN, Cross-validation including score tables), confusion matrices, conclusion, lessons learnt and future work