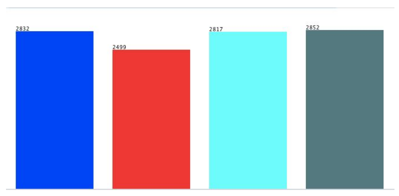
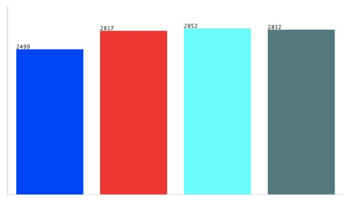
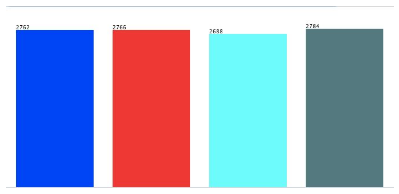
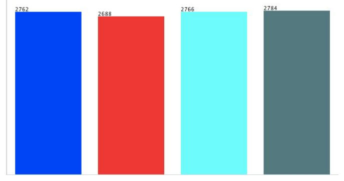
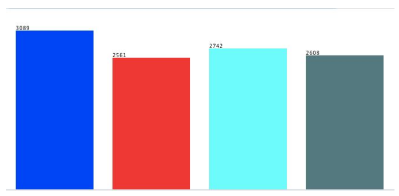
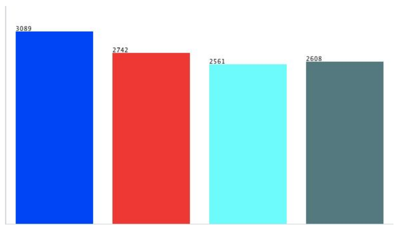
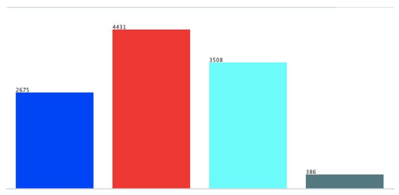
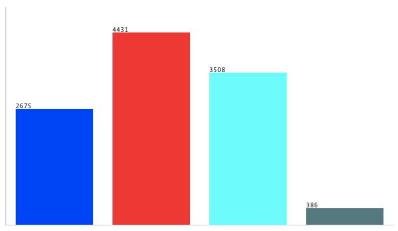
DMG Assignment 3 Report

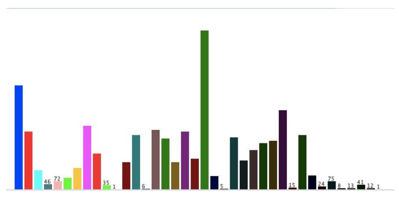
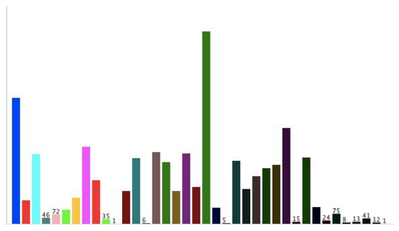
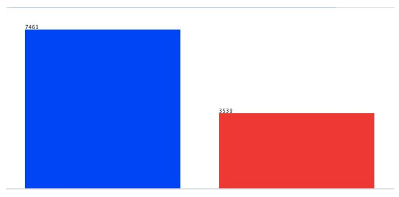
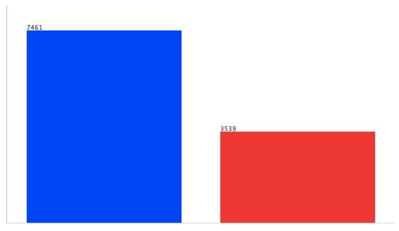
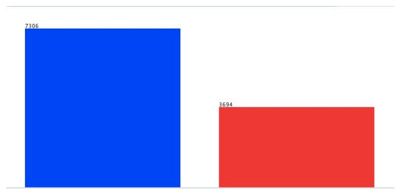
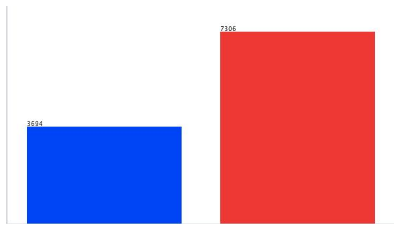
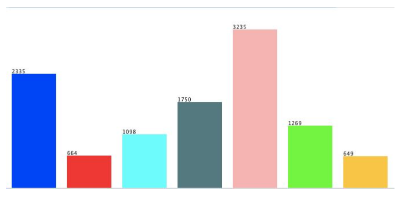
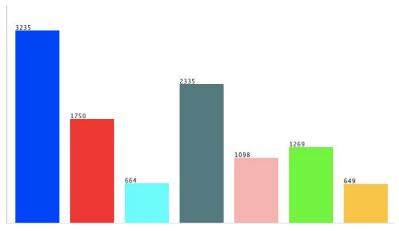
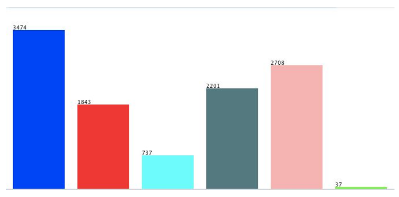
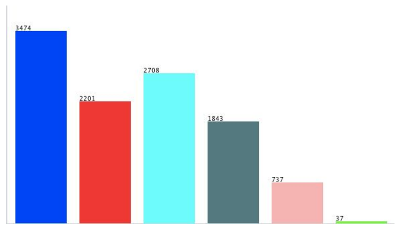
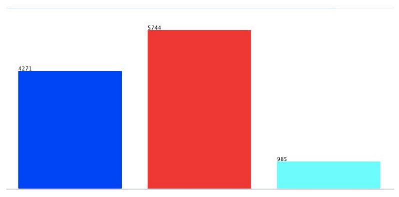
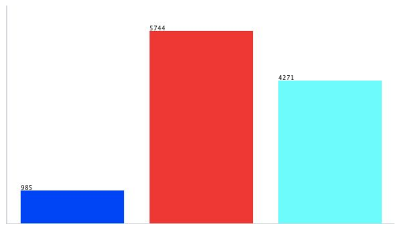
Navneet Agarwal
2018348

Aditya Singh
2018378

Data Visualization

(Each colour represents a unique value of an attribute)

Attribute	Original	After balancing the dataset
Elevation		
Aspect		
Slope		
Wilderness		

Soil_Type		
Hillshade_9am		
Hillshade_noon		
Horizontal_Distance_To_Hydrology		
Vertical_Distance_To_Hydrology		
Horizontal_Distance_To_Fire_Points		

Methodology

Preprocessing:

- In the given data 3 columns namely Horizontal_Distance_To_Hydrology, Vertical_Distance_To_Hydrology and Label consisted of numeric attributes(int datatype), and apriori algorithm(in weka wrapper) works on strings, so to handle this issue we mapped these columns data from int to string.
- We analysed the data and it consisted of different numbers of rows for each Label and there were a lot of duplicate records for Class 4, so we undersampled the data. After undersampling, we had an equal number of instances for each class.
- We saved this data frame to a CSV so that it can be loaded by python weka wrapper to create a dataset that can be fed into **weka.associations.Apriori** model. (We used weka wrapper for python)

Algorithm:

- We use the above preprocessed data set to extract the Apriori rules from the python-weka-wrapper. There is a -A parameter which helps to extract the Class-Association-Rules. The Labels appear on the consequence side while the attributes occur on the antecedent side.

- After extracting the rules, we pruned these rules according to the algorithm mentioned below.
    - For rule$_i$ in Extracted_Rules:
        - If the length of the rule(antecedent part) > min_len_threshold(set by us) :
            - Find all the instances in the training data which satisfy the antecedent and the consequence of the rule$_i$.
                - If none of the instances satisfies the rule$_i$ then continue.
                - Else, Remove all the instances which satisfy this rule and rule$_i$ to the final list of pruned rules.
        - Else:
            - Continue
- Open the test_X.csv and apply the first step of data preprocessing on this.
- Classification Algo:
    - For every instance$_i$ in test_data:
        - For every rule$_j$ in Pruned_Rule_List:
            - If instance$_i$ satisfy antecedent of rule$_j$:
                - Then assign the label of rule$_j$ (consequence) to instance$_i$.
                - continue
        - If instance$_i$ does not satisfy any of the above rules:
            - Assign a default class to instance$_{i.}$

Analysis of data and Algorithm:

- There were a lot of instances that were duplicate (3k), but removing them does not make sense, as there can be duplicate instances in this scenario.
- There was an imbalance between different class labels, which affected the kind of rules we extracted. So, we balanced the data set by undersampling the data.
- We modified different parameters under this model like modifying the minimum confidence, minimum support and number of rules to be extracted.
- We observed that if the number of rules extracted were very less (100~200) and minimum confidence was too high (0.80) then we had majority rules of only Class 4. So, we modified these values to extract more numbers of rules which covered all the classes.
- Under the pruning part, we tried different values for min_len_threshold (minimum length a rule should have). This was done because our data set contains an almost equal distribution of labels for different values of each attribute. If the length of a rule was very less like 1 then it may misclassify many instances.
- By varying these parameters we tried to improve our macro-f1 on kfold data and the model with the best parameters was used for Kaggle submission.
- The default label was assigned according to the class label which had the least covered instances after the pruning step.

Comparison of macro F1 scores with Random forest-based baseline:

K-fold Macro f1 score	Kaggle macro f1 score	Baseline RF macro f1 score
0.7130	0.68506	0.47451
0.6990	0.68500	0.47451
0.6930	0.68090	0.47451

We observe that we have achieved a higher macro f1 score from the baseline Random Forest. This means that we had higher recall and precision than the baseline Random Forest. This also suggests that we were able to fine-tune the rule-based classifier we made to perform better than the baseline model.

Learning:

- We learnt how to make a rule-based classifier using class association rules and for that, we used Apriori Algorithm in the python-weka-wrapper.
- We learnt the usage of a new python library called weka.
- We learned about various techniques (weka, SPMF, orange) to extract the rules using the Apriori algorithm.

- We learned about visualizing the data given to us in Weka's GUI.
- We learnt how to run the k-fold split on a model written from scratch.
- We learned about various pruning techniques and analysed our pruned data according to the F1 scores we got.

Contribution:

- Equal Contribution was done by both the team members.
- The analysis of the data was done by both the team members together.
- After the analysis, separate parts of code were handled individually and they were cross-checked by the other member.