REPORT

1) We have removed the number of rows from the image.csv and splitted the data into columns in the Edgehistogram.csv.

2) Load the “EdgeHistogram.csv” into a data frame “df_Histrogram” then rename the columns to be readable as previously in the csv the columns did not have any names. So, the image id column has been renamed to "Image_ID” and Features have been renamed as Feature_1, Feature_2 …. till the last feature. Then I have dropped the “Image_ID” column since all images are in an order top to bottom.

3) EdgeHistogram.csv is being loaded into the data frame "df Histrogram." The columns were previously unnamed in the csv, but I renamed them so they could be read. Therefore, "Image ID" has been added as a column name for the picture id column, and till the final feature, been renamed as Feature 1, Feature 2, etc. Since all of the images are arranged top to bottom, I have removed the "Image ID" column because it is useless.

4) I then combined the two separate data frames into a single data frame. " Concate DF" is the name of this data frame. The dataset was then examined for NULLs and NANs. To guarantee optimal training and performance, any missing values or NANs must be removed from the dataset. The dataset had no null values.

5) The dataset was described in order to provide a statistical summary of the dataset, including the mean, standard deviation, maximum, minimum, etc. for each numerical column. Determine whether normalization is effective or necessary with its assistance.

6) We have used preprocessing function `DF.iloc[:,1:] = Preprocess(DF.iloc[:,1:])` to see if the data is normalised or not. It will return TRUE if 1 is selected and FALSE if 2 is selected.

7) Since the dataset is not sufficiently sampled for training. Reducing the number of instances of a class label to match all class labels is known as Undersampling. As directed in the PDF text, this is carried out. 3,5,10, and 15 training photos per label are used to test each model. The total number of photos in the dataset will increase to 10 * 3 = 30 if there are 10 different labels, each of which will contain 3 samples.

8) Using random search and cv, I've implemented hyperparameter tuning in the third-to-last cell. The dataset was divided into train and test for this. The hyperparameter tuner receives this information. The model's corresponding parameter distribution dictionary is used to do a random search within the distribution for the optimal parameters. Cross validation allows me to identify the optimum parameters, which I then use to train my actual model.

9) It is then put to the test, and ratings like accuracy, f1 score, precision, and recall are computed. All of the outcomes are added to a data frame called "Global Performance DF," which is then sorted by f1 values.

10) The performance data frame is saved at the end in csv format for further analysis one using pre-processing and other without pre-processing.

Q1. How were the training / test image sets selected?

1) The ratio for images of each class label is constant in train and test, i.e.

For example, if images from class 'A' constitute 10% of total data in train, they will also constitute 10% of test data.

2) We kept the test size at 0.2 because we want the test data to account for 20% of the total data.

Q2. How were the parameters for the classification method optimized?

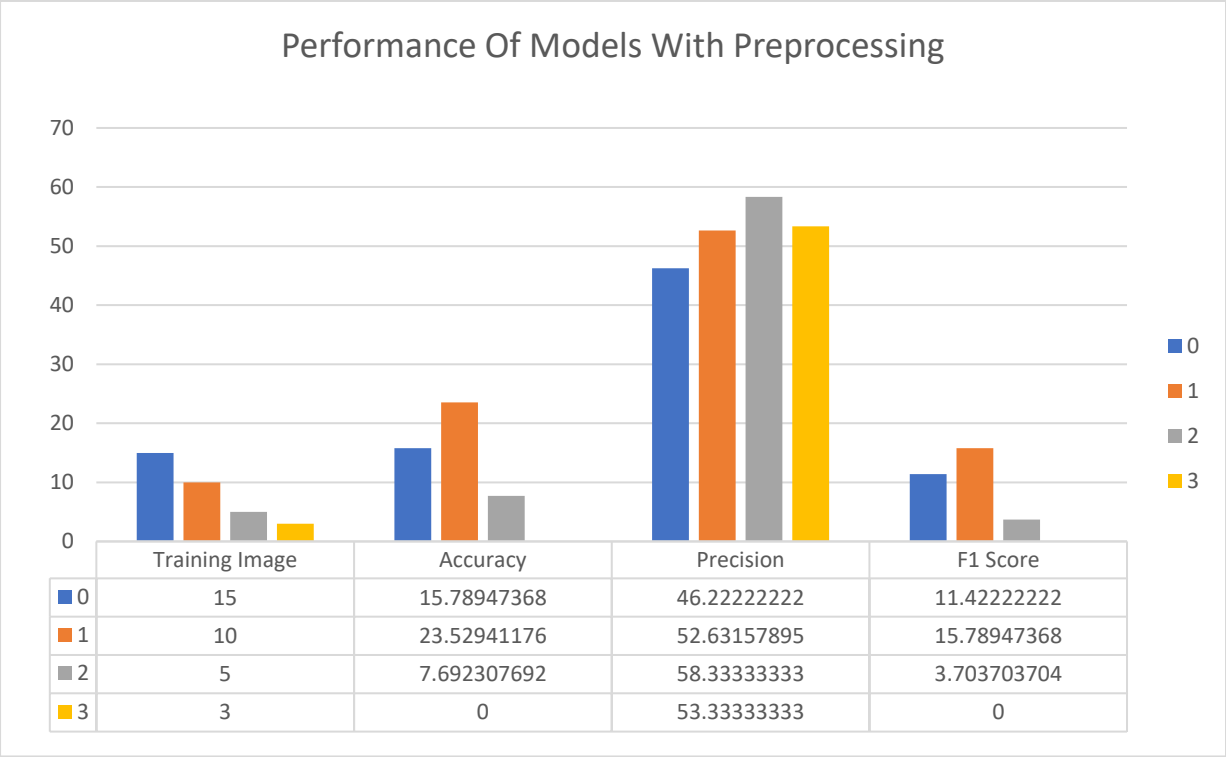
The parameters for the classification method optimized using GridSearchCv (hyperparameter tuning type).

Q3. How many different runs were executed for each amount of training images?

Only a single run will be executed for each amount of training images. (Either with or without pre-processing)

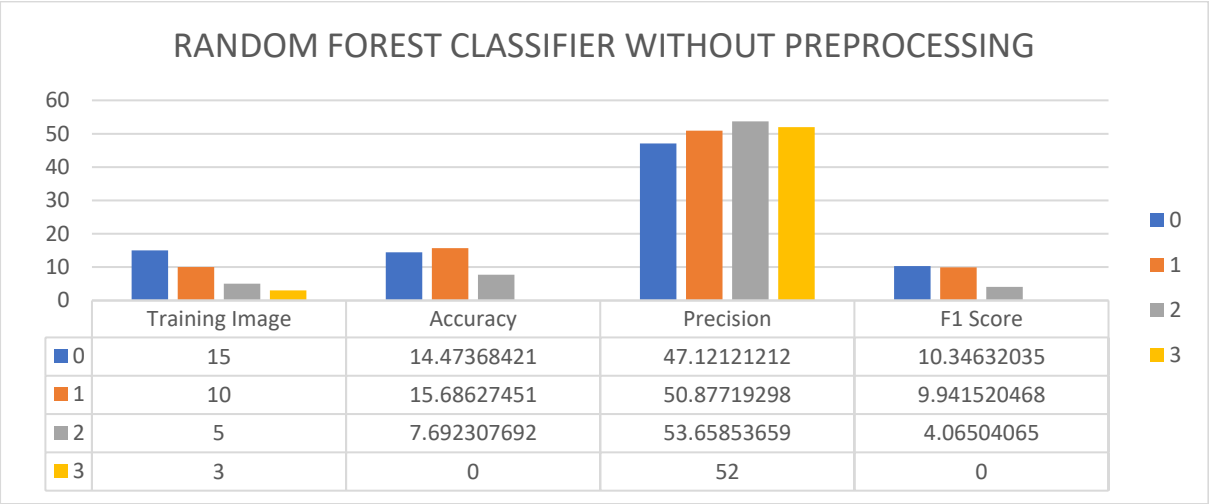
RESULT WITHOUT PREPROCESSING

	Training Image	Model	Parameters	Accuracy	Precision	F1 Score
0	15	RandomForestClassifier(max_depth=70, n_estimators=12, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 12, 'max_depth': 70}	15.78947368	46.22222222	11.42222222
1	10	RandomForestClassifier(max_depth=80, n_estimators=12, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 12, 'max_depth': 80}	23.52941176	52.63157895	15.78947368
2	5	RandomForestClassifier(max_depth=80, n_estimators=12, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 12, 'max_depth': 80}	7.692307692	58.33333333	3.703703704
3	3	RandomForestClassifier(max_depth=80, n_estimators=12, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 12, 'max_depth': 80}	0	53.33333333	0



RESULT WITHOUT PREPROCESSING

	Training Image	Model	Parameters	Accuracy	Precision	F1 Score
0	15	RandomForestClassifier(max_depth=80, n_estimators=12, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 12, 'max_depth': 80}	14.47368421	47.12121212	10.34632035
1	10	RandomForestClassifier(max_depth=100, n_estimators=10, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 10, 'max_depth': 100}	15.68627451	50.87719298	9.941520468
2	5	RandomForestClassifier(max_depth=70, n_estimators=7, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 7, 'max_depth': 70}	7.692307692	53.65853659	4.06504065
3	3	RandomForestClassifier(max_depth=70, n_estimators=7, n_jobs=-1)	{'n_jobs': -1, 'n_estimators': 7, 'max_depth': 70}	0	52	0



	Training Image	Accuracy	Precision	F1 Score
0	15	14.47368421	47.12121212	10.34632035
1	10	15.68627451	50.87719298	9.941520468
2	5	7.692307692	53.65853659	4.06504065
3	3	0	52	0