CS4011 : INTRODUCTION TO MACHINE LEARNING

REPORT

November 13, 2017

Student ID: EE15B123 & EE15B025
Namida M
Ganga Meghanath

Contents

1 Introduction . . . 3
2 Folder Structure . . . 3
3 Data Analysis . . . 4
3.1 General observations . . . 4
3.2 Missing Value Analysis . . . 6
3.3 Imputation and Removal . . . 9
3.4 Scaling and Normalization . . . 9
3.5 Linear Discriminant Analysis (LDA) . . . 10
3.6 Principle Component Analysis (PCA) . . . 10
3.7 Clustering . . . 12
4 Implemented Algorithms . . . 16
4.1 Neural Networks . . . 16
4.1.1 Procedure followed . . . 17
4.1.2 (a) Hyperparameters . . . 18
4.1.3 (b) Observation and Intuition . . . 21
4.2 Random Forest . . . 21
4.2.1 Reduced feature space . . . 22
4.2.2 Imputed original feature space . . . 25
4.3 Adaboost . . . 27
4.4 Gradient Boost . . . 29
4.5 Naive Bayes . . . 34
4.5.1 Gaussian . . . 34
4.6 SVM . . . 36
4.6.1 Sigmoid Kernel . . . 36
4.6.2 Gaussian Kernel . . . 36
4.7 XGBoost . . . 36
4.8 Voting Classifier . . . 40

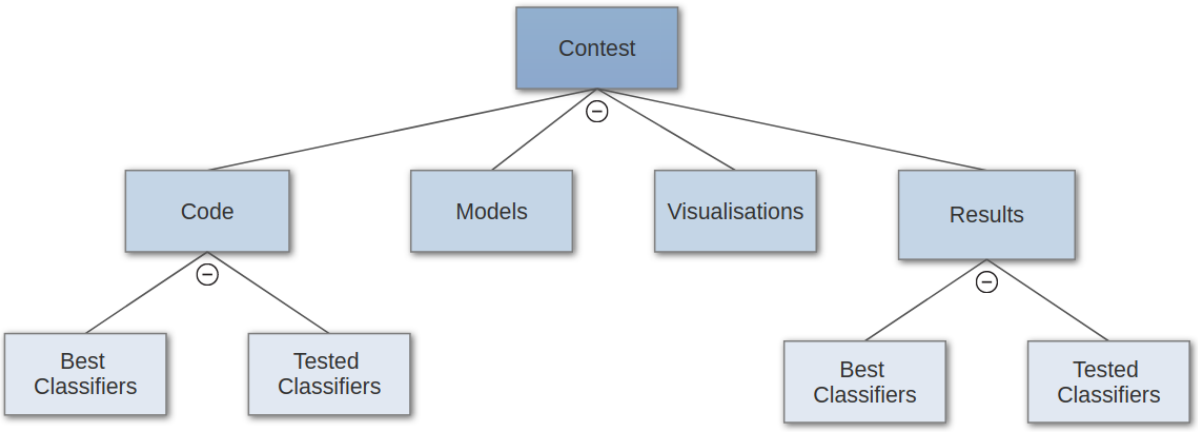
5 Our Insights about the Dataset 42
6 Final Model 43
References 44

1 INTRODUCTION

Machine learning is a type of artificial intelligence (AI) that allows software applications to become more accurate in predicting outcomes without being explicitly programmed. The basic premise of machine learning is to build algorithms that can receive input data and use statistical analysis to predict an output value within an acceptable range.

Machine learning algorithms are often categorized as being supervised or unsupervised. Supervised algorithms require humans to provide both input and desired output, in addition to furnishing feedback about the accuracy of predictions during training. Once training is complete, the algorithm will apply what was learned to new data. Unsupervised algorithms do not need to be trained with desired outcome data. Instead, they use an iterative approach called deep learning to review data and arrive at conclusions.

2 FOLDER STRUCTURE



3 DATA ANALYSIS

3.1 General observations

We have been given a dataset consisting of 29 classes with 9501 training instances, of dimension 2600. The class based distribution has been shown below.

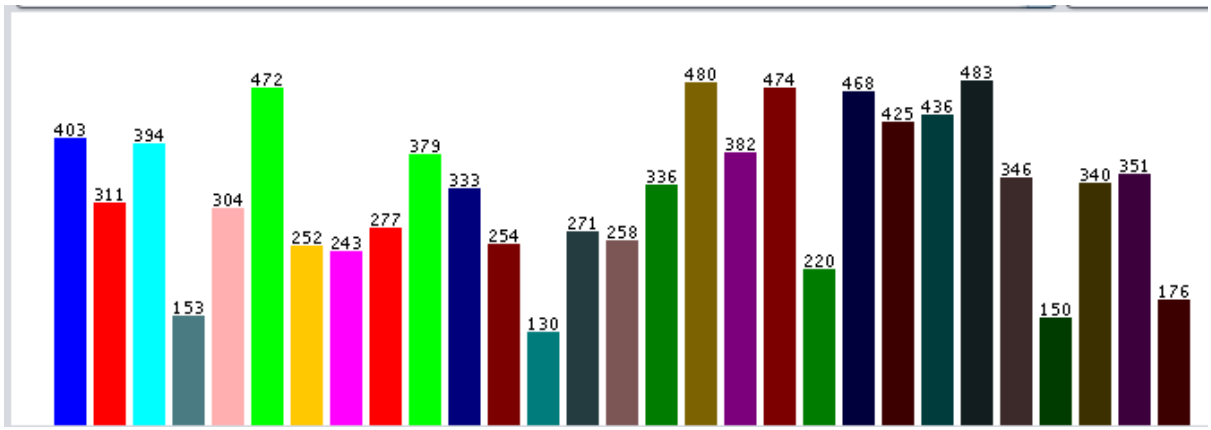
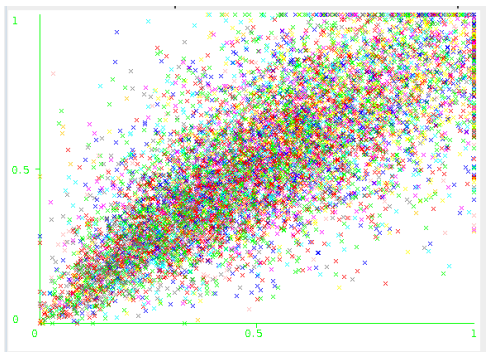
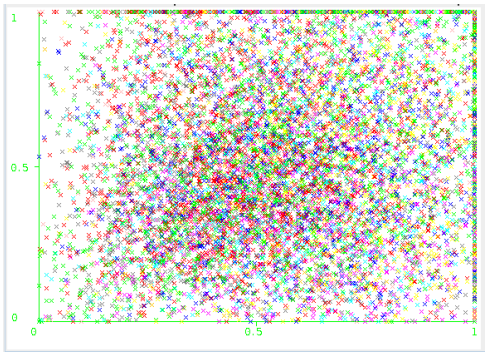


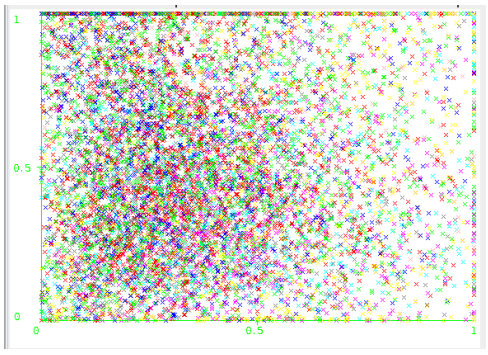
Figure 1: Class Distribution



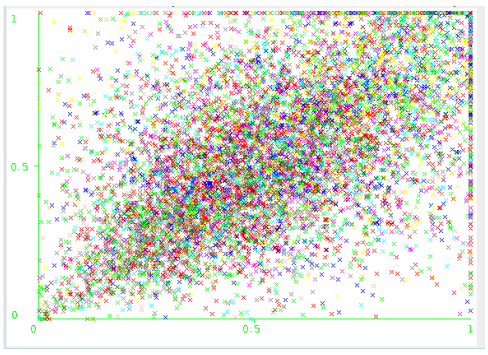
(a) f1405 vs f2368



(b) f1371 vs f2433



(c) f681 vs f2564



(d) f697 vs f2571

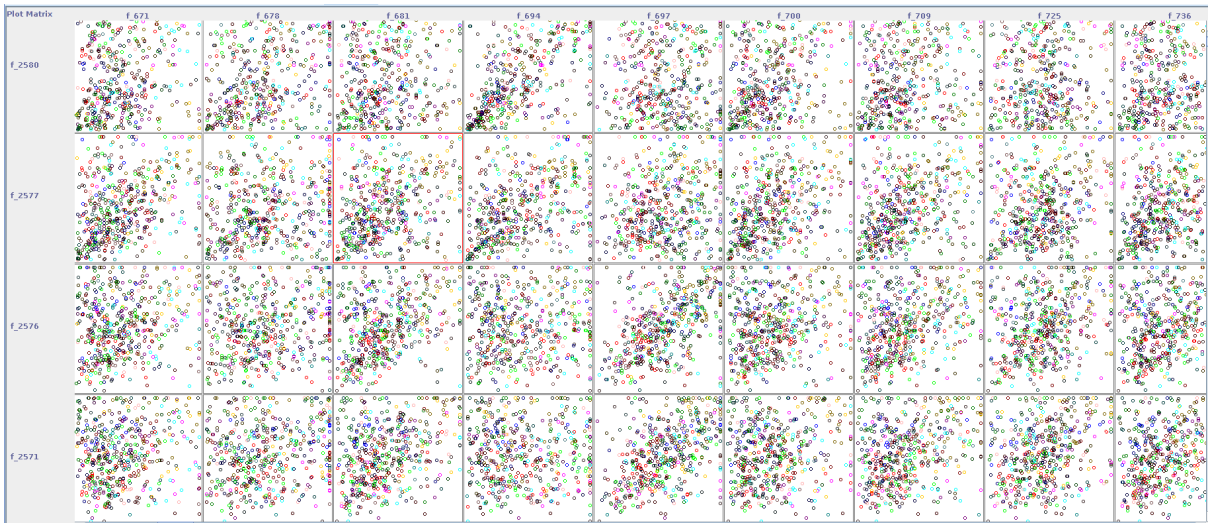


Figure 2: Feature Analysis

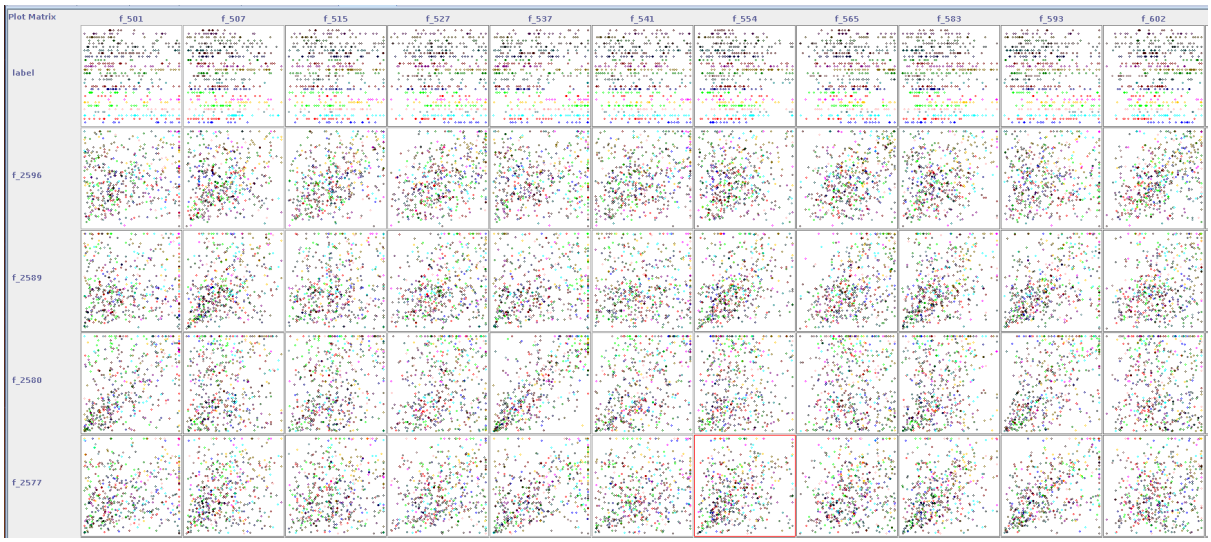


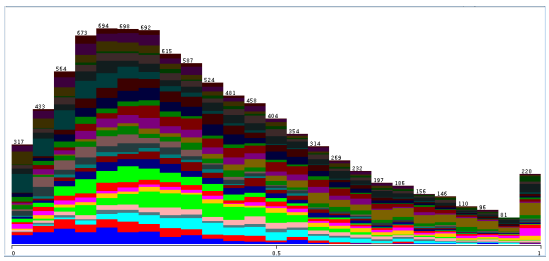
Figure 3: Feature Analysis

We can observe from the figures that there are correlated features in the data. Hence, we can try out different dimensionality techniques on the data to reduce the feature space. This will help reduce the computation time, as well as remove unwanted features from the dataset.

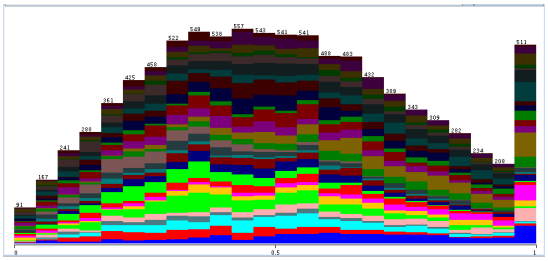
Also, it is likely that there is a lot of overlap between the different classes in the original 2600 dimensional space. Hence, the classification task is probably difficult. This can be infered from the above figures upon seeing the scattered data.

The initial figure also gives us understanding about the class imbalance that exists in the provided dataset that might make recognition of minority class instances difficult.

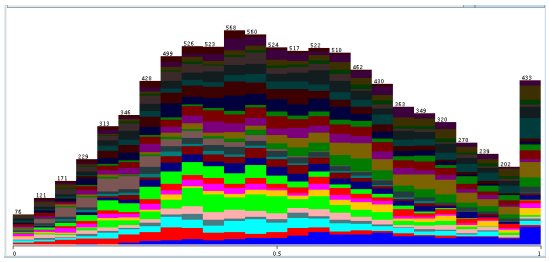
The class-based distribution of data along some features have been depicted below :



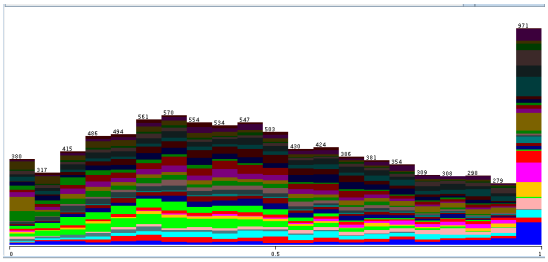
(a) f997



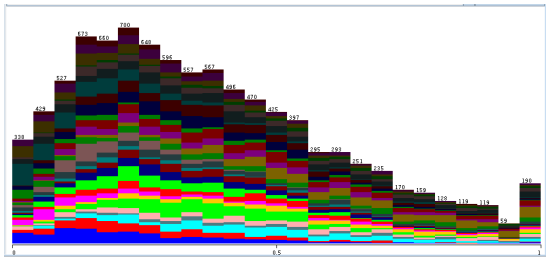
(b) f1663



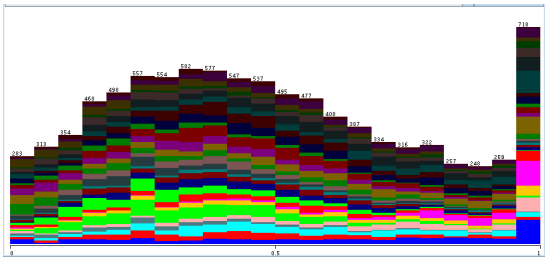
(c) f2033



(d) f2546



(e) f507



(f) f996

3.2 Missing Value Analysis

In real world data, many a times, one is not able to get values for all features of a data point. Depending on the number of missing values, and the pattern in which they are missing, the imputation strategy becomes really important.

The statistics of the missing values in the given dataset are as follows :

View	Estimates of regions missing values
Datapoints	Average of 200 in each row
Features	Around 3750 in each of the first 500 columns

The frequency plots are as follows :

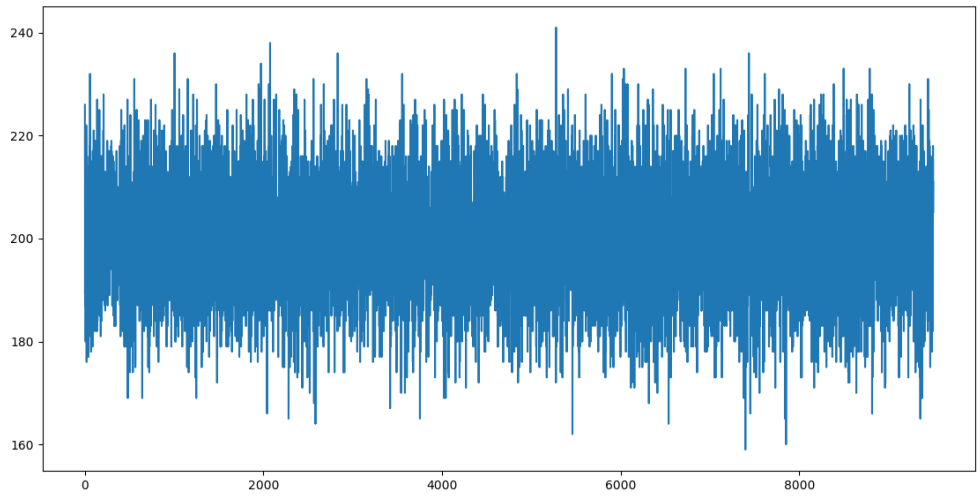


Figure 4: Train data: Plot of no. of missing attributes vs each datapoint

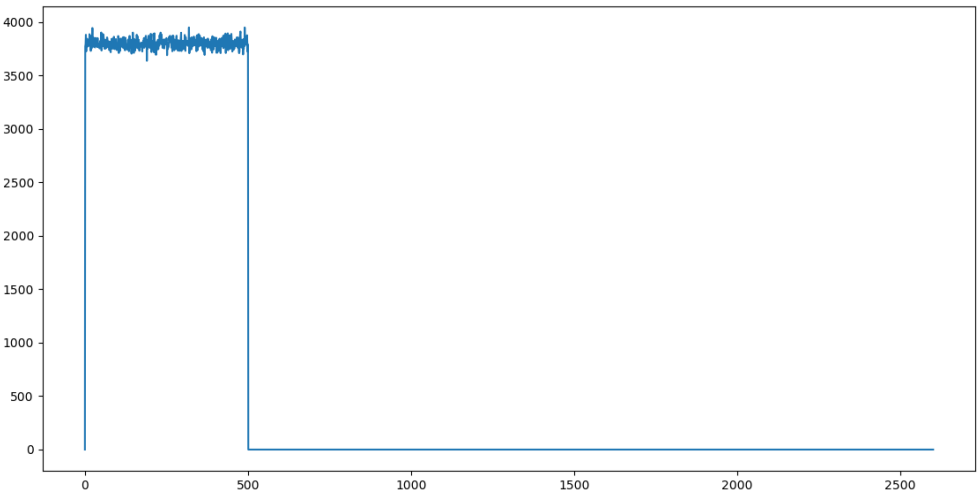


Figure 5: Train-data: Plot of no. of missing values vs attribute index

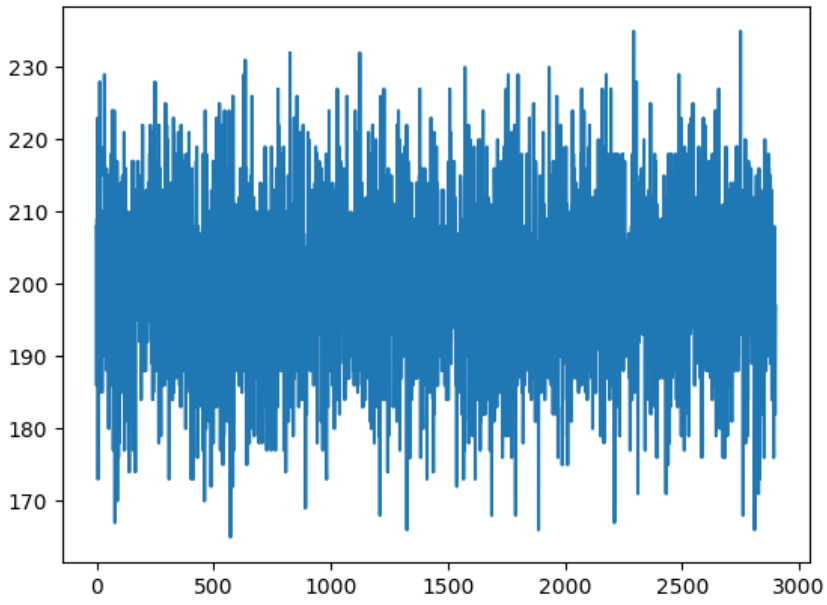


Figure 6: Test-data: Plot of no. of missing values vs attribute index

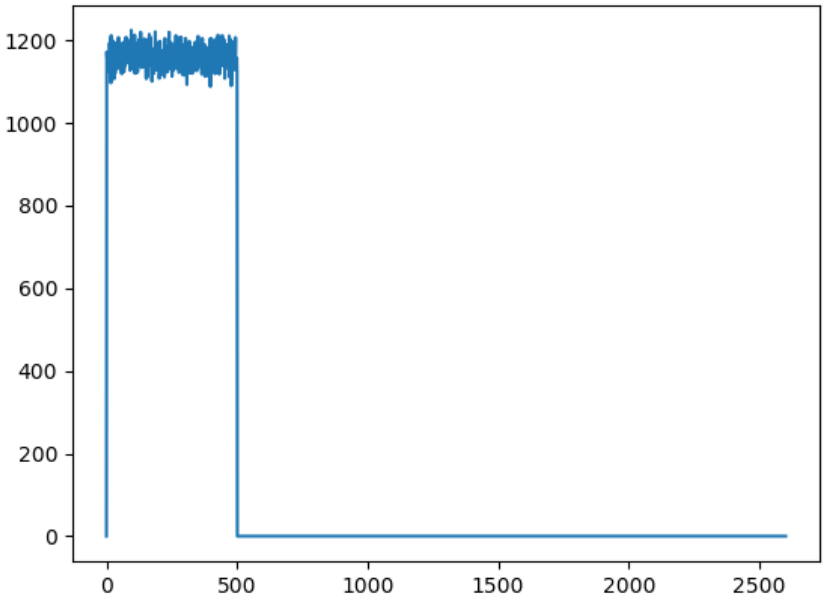


Figure 7: Test-Data: Plot of no. of missing values vs data point

3.3 Imputation and Removal

There are four simple options we can use in such cases:

- Mean Imputation
  Advantage - mean of each feature is preserved.
  Disadvantage - same value inserted in all vacancies in a column. Hence when the statistics of missing values per column is high, the contribution of the column towards class prediction decreases significantly (especially if the missing values have roughly equal in number for each class (or weighted by no. of instances in each class)
- Median Imputation: Advantage:The median value is mostly unaffected by outliers.
- Mode Imputation
- Remove rows/columns: In cases where there are too many values missing to make any proper inferences about the data, it may be best to not use the data at all. Even if we use regression to fill in values , the feature itself may become redundant.

Other possible ways :

- Linear Regression : Initial 500 attributes in the given dataset have missing values. Hence linear regression models are built using the rest of the 1500 attributes to predict the missing values in the first 500 attributes. This is by assuming that the attributes consisting of missing can be roughly represented as some linear combination of the last 1500 attributes.
- Correlation : Replacing the missing values using another attribute that is highly correlated with the attribute consisting of missing values.
- Class-conditioned Regression : Consider the class as an attribute during linear regression or do class-wise linear regression, ie, linear regression for each class separately. But this cannot be implemented on the test data since class labels are not available. Hence, full information regression is undertaken for the test data.

3.4 Scaling and Normalization

On analysing the data, a few more interesting observations can be made:

- The minimum of each feature is zero

- The maximum of each feature is 1 (implying that the data has probably already been scaled)
- The are also a large number of '1's in the data this can be observed from the distribution of one of the features.

3.5 Linear Discriminant Analysis (LDA)

Linear Discriminant Analysis (LDA) is a commonly used dimensionality reduction technique in the pre-processing step for pattern-classification and machine learning applications. The goal is to project a dataset onto a lower-dimensional space with good class-separability in order avoid overfitting (âĂIJcurse of dimensionalityâĂİ) and also reduce computational costs.

But on performing LDA with the given features, it was observed that only 28 features were returned upon specifying any number of dimensions greater that 28. This could mean that there are correlated features and dependent features in the given dataset. The correlation can also be clearly seen in visualisations that were depicted above.

On running the classifiers on LDA reduced dataset, the Precision, Recall and F-scores obtained came only upto 0.15. Hence this method of dimensionality reduction wasn't used in further computations.
Possible explanation: Maybe the distribution of the classes was skewed/not separable in a simple linear manner. This probably made the features it returned worse than their inverse transforms.

3.6 Principle Component Analysis (PCA)

Principal component analysis is a method of extracting important variables from a large set of variables available in a data set. It extracts low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible. With fewer variables, visualization also becomes much more meaningful.

PCA is mathematically defined as an orthogonal linear transformation that transforms the data to a new coordinate system.It can be thought of as fitting an n-dimensional ellipsoid

to the data, where each axis of the ellipsoid represents a principal component. If some axis of the ellipsoid is small, then the variance along that axis is also small, and by omitting that axis and its corresponding principal component from our representation of the dataset, we lose only a small amount of information.

We worked with a PCA reduced data consisting of 500 features in most of the classifiers. This value was based upon the obtained variance score which reduces steeply and then remains almost constant. This can be observed from the plot given below :

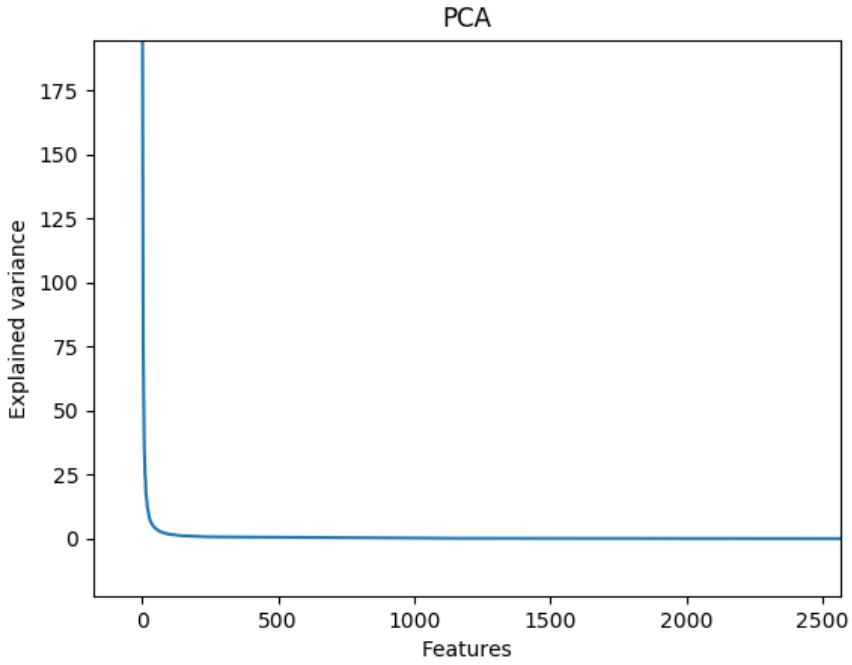


Figure 8: Plot of explained variances of features

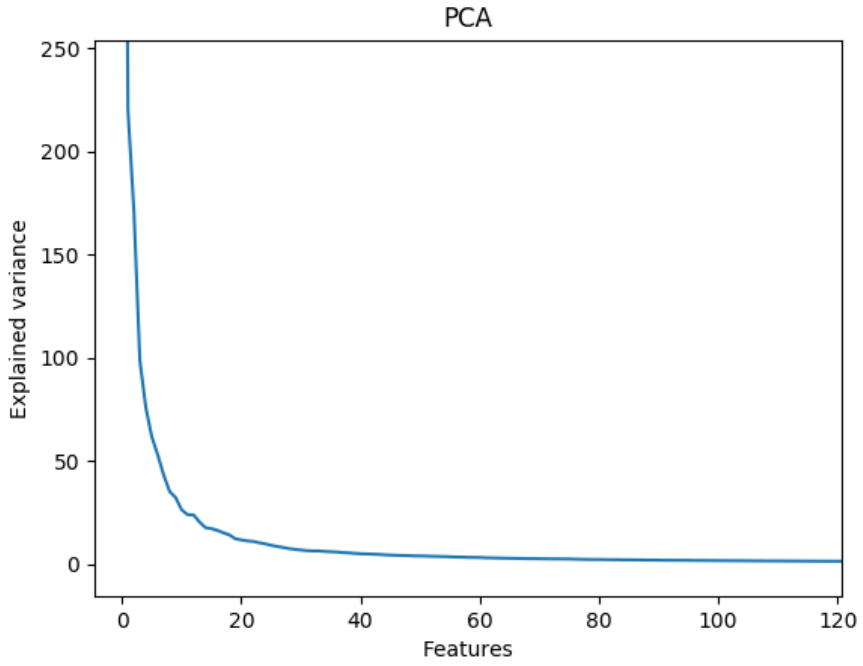


Figure 9: Plot of explained variances of features(a closer look)

3.7 Clustering

Clustering is the task of dividing the population or data points into a number of groups such that data points in the same groups are more similar to other data points in the same group than those in other groups. A cluster is therefore a collection of objects which are âĂIJsimilarâĂİ between them and are âĂIJdissimilarâĂİ to the objects belonging to other clusters. Hence, clustering is a way of studying the underlying distribution for classification.

DBSCAN Density-based spatial clustering of applications with noise (DBSCAN) is a data clustering algorithm which is density based: given a set of points in some space, it groups together points that are closely packed together (points with many nearby neighbors), marking as outliers points that lie alone in low density regions (whose nearest neighbors are too far away).

We tried Density Based Clustering on the dataset reduced to a feature space of 500 dimensions. Due to the large number of dimensions and instances, it was difficult loading the dataset into Weka. But the results obtained were poor as can be seen below :

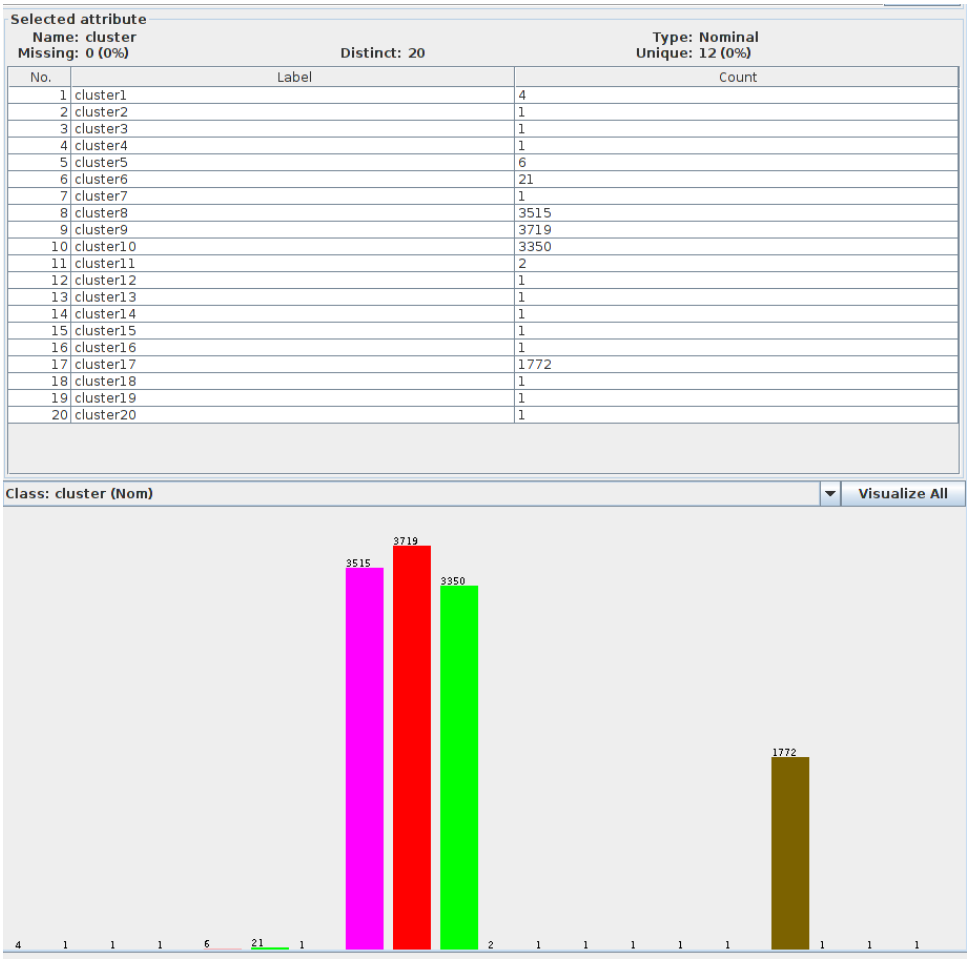


Figure 10: Density Based Cluster : 500 features

Kmeans: It is a type of unsupervised learning, which aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean, serving as a prototype of the cluster.

We first tried k-means on the reduced feature space of 500 dimensions with k=10. The results obtained were as follows :

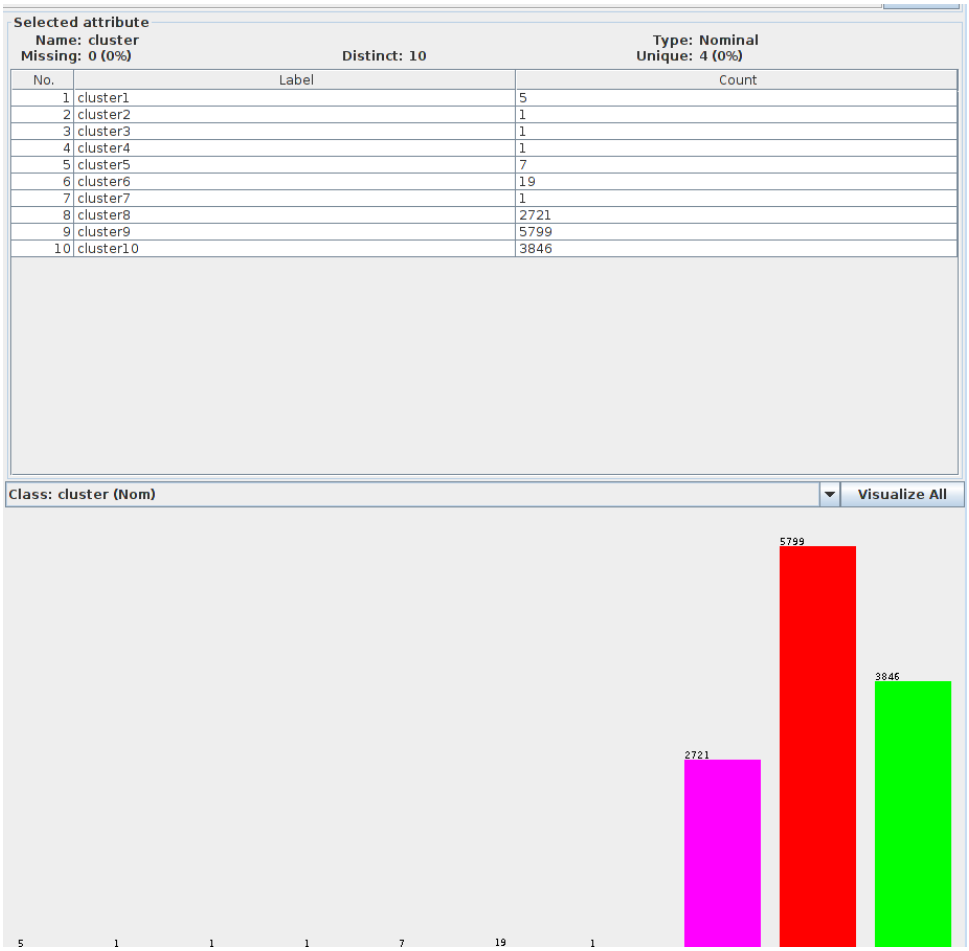


Figure 11: K-Means (k=10) : 500 features

Hence, clustering wasn't considered in all the initial stages.

But towards the end, we tested K-means clustering on the original training dataset consisting of 2600 dimensions and 9501 instances. Loading the dataset onto Weka was very difficult and it failed multiple times. We were only able to load the full dataset once, and the results obtained have been depicted below :

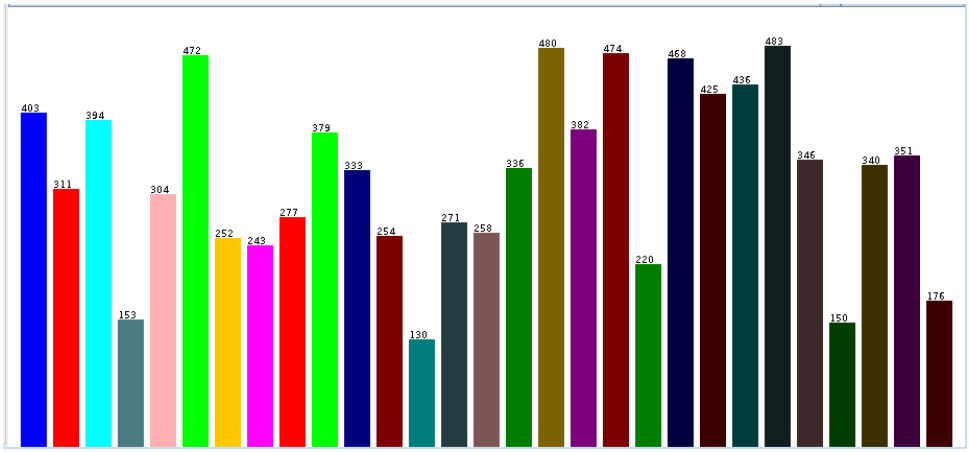


Figure 12: Class Labels : 2600 features

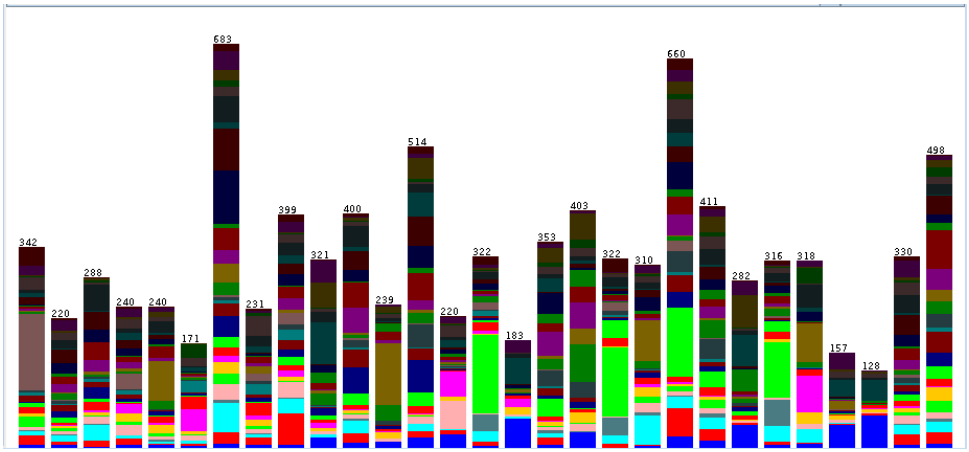


Figure 13: K-Means (k=29) : 2600 features

The above image shows the fraction of cluster instances in each of the 29 clusters. It has been color-coded by the class label. Hence we can see that there are clusters that contain majority of a particular class instances. This gives us the inference that such classes are more easily identifiable when we run our classification algorithms. We can also see the class imbalances. Hence, we have to keep in mind that during classification, identification of the instances belonging to the minority class can be difficult.

4 IMPLEMENTED ALGORITHMS

4.1 Neural Networks

A multilayer perceptron (MLP) is a class of feedforward artificial neural network. An MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called backpropagation for training. Its multiple layers and non-linear activation distinguish it from a linear perceptron. It can be utilised to distinguish data that is not linearly separable.

An MLP (or Artificial Neural Network - ANN) with a single hidden layer can be represented graphically as follows:

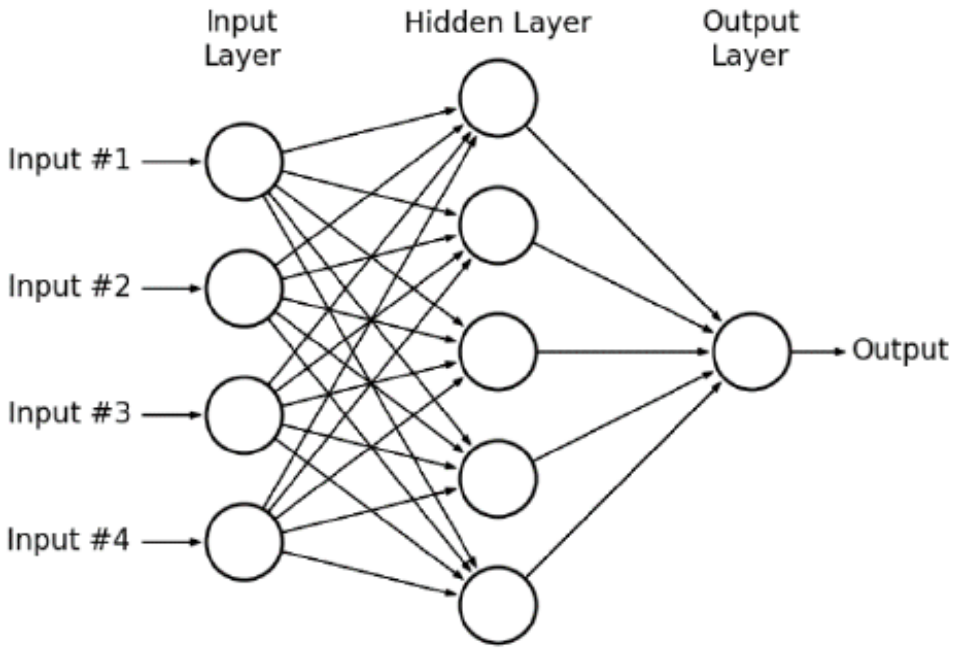


Figure 14: Multilayer Layer Perceptron Network

The advantages of using Neutal Networs are as follows :

- Adaptive learning: An ability to learn how to do tasks based on the data given for training or initial experience
- Self-Organisation: An ANN can create its own organisation or representation of the information it receives during learning time

- It has the ability to derive meaning from complicated or imprecise data, and can be used to extract patterns and detect trends that are too complex to be noticed by either humans or other computer techniques

4.1.1 Procedure followed

In addition to the hyperparameters of the neural net itself,there are a few other aspects that can be varied for this part.

- Out of all mentioned possibilities, what strategy was used to clean/impute the data.
  - Median Imputation: did not work well(variance in these features drop significantly)
  - Feature removal: The first five hundred features were thrown away. This is what gave the best results.
- The are a large number of features in the data. What feature reduction/selection technique was used on the data.
  - LDA :This does not work well at all.(possibly because the data is skewed)
  - Raw data (after imputation/removal)
  - PCA Helped reduce the features, and enabled us to be able to apply more complex and computationally intensive models, while providing us with results in a reasonably short time interval.
- What if multiple neural nets were sensitive to different classes. How do we combine the models?
  - Mode: We analysed the classification reports for models that had high f-scores, preferably with a different trend in fscores over different classes. Finally, the mode of the predictions was calculated(and submitted). This is the equivalent of a hard margin voting classifier with equal class weights.
  - Weka J48: In order to combine the models in a more complex manner , we need an algo that is able work with data that has only nominal features. The J48 seems to do this. The validation fscore was observed to be 0.37, but on submission, we were only able to achieve a score of 0.30.

4.1.2 (a) Hyperparameters

The following values are a small subset of the set of values we tried(these worked relatively better).

PCA features	**MLP layers**			**Avg Fscore**
n_features	**Layer1**	**Layer 2**	**Layer 3**	**F-measure**
1400	800	800	1400	0.31
800	800	800	400	0.32
400	900	500	300	0.34
100	800	800	400	0.38
100	800	1200	300	0.41
60	800	1200	300	0.42
30	800	1200	300	0.35

(NOTE: The results are obtained after doing a 5 fold cross validation and choosing the best performing model for each set of parameters)

The rest of the hyperparameters of the MLP classifier were kept constant. Some of the other important hyperparameters are as follows:

Activation='relu', Solver='adam', alpha=0.0005, batch-size='auto',learning-rate='adaptive' learning-rate-init=0.0005, power-t=0.5, max-iter=15000, shufle=True, random-state=5, tol=0.0001, early-stopping=False,validation-fraction=0.1

The classification report for the best 2 models have been shown below :

Cross_val2, PCA_comp:60MLP layers: 800,1200,300

Class	Precision	Recall	f1-score	support
1	0.51	0.53	0.52	40
2	0.34	0.39	0.36	31
3	0.42	0.33	0.37	40
4	0.38	0.40	0.39	15
5	0.46	0.43	0.45	30
6	0.60	0.87	0.71	47
7	0.39	0.28	0.33	25
8	0.64	0.67	0.65	24
9	0.29	0.25	0.27	28
10	0.38	0.42	0.40	38
11	0.32	0.39	0.35	33
12	0.16	0.12	0.14	25
13	0.22	0.15	0.18	13
14	0.26	0.37	0.31	27
15	0.73	0.85	0.79	26
16	0.42	0.41	0.42	34
17	0.59	0.71	0.64	48
18	0.26	0.32	0.28	38
19	0.33	0.25	0.29	48
20	0.23	0.14	0.17	22
21	0.42	0.47	0.44	47
22	0.19	0.14	0.16	43
23	0.59	0.52	0.55	44
24	0.43	0.33	0.38	48
25	0.48	0.46	0.47	35
26	0.50	0.47	0.48	15
27	0.36	0.41	0.38	34
28	0.53	0.57	0.55	35
29	0.27	0.22	0.24	18
avg / total	0.42	0.43	0.42	951

Figure 15: Classification Report

Cross_val1, PCA_comp:80MLP layers: 1200,300,300

Class	Precision	Recall	f1-score	support
1	0.61	0.57	0.59	40
2	0.21	0.19	0.20	31
3	0.30	0.30	0.30	40
4	0.43	0.40	0.41	15
5	0.33	0.27	0.30	30
6	0.55	0.68	0.61	47
7	0.33	0.24	0.28	25
8	0.66	0.79	0.72	24
9	0.39	0.25	0.30	28
10	0.35	0.32	0.33	38
11	0.45	0.30	0.36	33
12	0.12	0.16	0.14	25
13	0.57	0.31	0.40	13
14	0.31	0.37	0.34	27
15	0.62	0.77	0.69	26
16	0.56	0.53	0.55	34
17	0.60	0.77	0.67	48
18	0.24	0.26	0.25	38
19	0.28	0.29	0.29	48
20	0.17	0.18	0.18	22
21	0.34	0.38	0.36	47
22	0.14	0.14	0.14	43
23	0.40	0.32	0.35	44
24	0.36	0.40	0.38	48
25	0.54	0.40	0.46	35
26	0.50	0.33	0.40	15
27	0.35	0.38	0.37	34
28	0.53	0.66	0.59	35
29	0.44	0.39	0.41	18
avg / total	0.40	0.40	0.40	951

Figure 16: Classification Report

4.1.3 (b) Observation and Intuition

From the table we notice a few things:

- Increasing the number of features mostly does NOT increase the performance of the model.
- The The number of perceptrons in the first and second layer of the good models are much higher than the number of input. One possible explanation is that the first layer transforms the input into a "useful" set of features. Continuing on this logic, we tried increasing the number of inputs in the second layer as compared to the first. An f-score of 0.42 was achieved on the validation set.
- Decreasing the number of features to 30 does not work. So the knee point for optimal number of features must lie between 60 and 30.

4.2 Random Forest

Random forests is an ensemble learning method for classification, regression and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees. It also reduces the chances of overfitting on the training data. The classifier creates the set of decision trees from randomly selected subset of training set (bagging).

The test labels were predicted by using two methods :

- Taking the mode of the result of 5-fold cross-validation, ie, assigning the class predicted by majority of the folds.
- Summing up the class probabilities in each of the 5 folds for the predicted test labels and assigning it to the class having highest probability

This classifier was chosen over a single decision tree to counter overfitting and also get a more confident result as it is based on bagging and the result is given by an estimation from the predictions of all the estimators.

The classifier was tested using different sets of parameters. Two approaches were done :

4.2.1 Reduced feature space

Random forest was tested on the reduced feature space consisting of PCA reduced 500 features. Some of the best results obtained are as follows :

n_estimators=2000, max_depth=25, criterion='gini', min_samples_split=2, min_samples_leaf=1, n_jobs=-1, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.35	0.68	0.46	60
1.0	0.58	0.15	0.24	47
2.0	0.54	0.22	0.31	59
3.0	0.00	0.00	0.00	23
4.0	0.88	0.15	0.26	46
5.0	0.33	0.94	0.49	71
6.0	0.00	0.00	0.00	38
7.0	0.71	0.67	0.69	36
8.0	0.00	0.00	0.00	42
9.0	0.27	0.40	0.33	57
10.0	0.43	0.06	0.11	50
11.0	0.00	0.00	0.00	38
12.0	0.00	0.00	0.00	20
13.0	0.00	0.00	0.00	41
14.0	0.57	0.64	0.60	39
15.0	0.45	0.20	0.28	50
16.0	0.38	0.81	0.51	72
17.0	0.20	0.12	0.15	57
18.0	0.15	0.32	0.20	71
19.0	0.00	0.00	0.00	33
20.0	0.23	0.56	0.32	70
21.0	0.24	0.25	0.25	64
22.0	0.47	0.52	0.50	65
23.0	0.28	0.28	0.28	72
24.0	0.30	0.27	0.29	52
25.0	1.00	0.13	0.23	23
26.0	0.42	0.25	0.32	51
27.0	0.37	0.43	0.40	53
28.0	0.00	0.00	0.00	26
avg / total	0.32	0.33	0.28	1426

Figure 17: Random Forest : Classificatiion Report

n_estimators=1750, max_depth=25, criterion='gini', min_samples_split=2, min_samples_leaf=1, n_jobs=-1, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.34	0.65	0.45	60
1.0	0.40	0.09	0.14	47
2.0	0.65	0.25	0.37	59
3.0	0.00	0.00	0.00	23
4.0	0.75	0.13	0.22	46
5.0	0.30	0.92	0.45	71
6.0	0.00	0.00	0.00	38
7.0	0.71	0.61	0.66	36
8.0	0.00	0.00	0.00	42
9.0	0.32	0.47	0.38	57
10.0	0.50	0.06	0.11	50
11.0	0.00	0.00	0.00	38
12.0	0.00	0.00	0.00	20
13.0	1.00	0.02	0.05	41
14.0	0.56	0.62	0.59	39
15.0	0.54	0.26	0.35	50
16.0	0.37	0.83	0.51	72
17.0	0.27	0.14	0.18	57
18.0	0.17	0.34	0.22	71
19.0	0.50	0.03	0.06	33
20.0	0.22	0.57	0.32	70
21.0	0.18	0.20	0.19	64
22.0	0.40	0.52	0.46	65
23.0	0.23	0.22	0.23	72
24.0	0.37	0.27	0.31	52
25.0	1.00	0.09	0.16	23
26.0	0.61	0.22	0.32	51
27.0	0.34	0.40	0.37	53
28.0	0.00	0.00	0.00	26
avg / total	0.37	0.32	0.27	1426

Figure 18: Random Forest : Classificatiion Report

Some of the other good results obtained are as follows :

Main Parameters			**Average Estimates**		
n_estimators	**max_depth**	**criterion**	**Precision**	**Recall**	**F-measure**
3000	25	Entropy	0.28	0.29	0.24
3000	None	Entropy	0.33	0.29	0.25
Balanced Subsample					
3000	None	Gini	0.31	0.30	0.25
6000	None	Gini	0.35	0.30	0.25
6000	15	Gini	0.30	0.28	0.25
3000	None	Entropy	0.30	0.29	0.24
6000	None	Entropy	0.40	0.29	0.24
6000	60	Entropy	0.37	0.32	0.27

The parameters other than the ones specified that have been used are :

min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, random_state=None, warm_start=False

4.2.2 Imputed original feature space

As the computation time for the algorithm was lesser compared to the other tested classifiers, we decided tot test Random forest with 2600 features per instance. The best 2 results obtained are as follows :

n_estimators=2000, max_depth=25, criterion='gini', min_samples_split=2, min_samples_leaf=1, n_jobs=-1, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.54	0.47	0.50	60
1.0	0.34	0.36	0.35	47
2.0	0.42	0.17	0.24	59
3.0	1.00	0.04	0.08	23
4.0	0.64	0.20	0.30	46
5.0	0.34	0.85	0.49	71
6.0	0.00	0.00	0.00	38
7.0	0.74	0.78	0.76	36
8.0	0.33	0.07	0.12	42
9.0	0.27	0.51	0.36	57
10.0	0.35	0.14	0.20	50
11.0	1.00	0.03	0.05	38
12.0	1.00	0.05	0.10	20
13.0	0.39	0.17	0.24	41
14.0	0.52	0.85	0.65	39
15.0	0.34	0.32	0.33	50
16.0	0.42	0.74	0.53	72
17.0	0.26	0.33	0.29	57
18.0	0.17	0.30	0.22	71
19.0	0.00	0.00	0.00	33
20.0	0.27	0.37	0.32	70
21.0	0.18	0.31	0.22	64
22.0	0.46	0.57	0.51	65
23.0	0.33	0.22	0.26	72
24.0	0.41	0.27	0.33	52
25.0	0.83	0.22	0.34	23
26.0	0.38	0.41	0.39	51
27.0	0.39	0.38	0.38	53
28.0	0.00	0.00	0.00	26
avg / total	0.39	0.35	0.32	1426

Figure 19: Random Forest : Classificatiion Report

n_estimators=1750, max_depth=25, criterion='gini', min_samples_split=2, min_samples_leaf=1, n_jobs=-1, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.53	0.50	0.51	60
1.0	0.22	0.21	0.22	47
2.0	0.44	0.19	0.26	59
3.0	0.00	0.00	0.00	23
4.0	0.65	0.28	0.39	46
5.0	0.37	0.90	0.52	71
6.0	0.80	0.11	0.19	38
7.0	0.36	0.56	0.43	36
8.0	0.33	0.05	0.08	42
9.0	0.27	0.47	0.35	57
10.0	0.32	0.20	0.25	50
11.0	1.00	0.03	0.05	38
12.0	0.00	0.00	0.00	20
13.0	0.20	0.07	0.11	41
14.0	0.52	0.82	0.64	39
15.0	0.35	0.36	0.36	50
16.0	0.38	0.65	0.48	72
17.0	0.21	0.28	0.24	57
18.0	0.20	0.32	0.25	71
19.0	1.00	0.06	0.11	33
20.0	0.19	0.29	0.23	70
21.0	0.16	0.22	0.19	64
22.0	0.42	0.43	0.43	65
23.0	0.29	0.17	0.21	72
24.0	0.35	0.23	0.28	52
25.0	0.83	0.22	0.34	23
26.0	0.23	0.31	0.26	51
27.0	0.38	0.36	0.37	53
28.0	0.00	0.00	0.00	26
avg / total	0.37	0.32	0.29	1426

Figure 20: Random Forest : Classificatiion Report

The parameters other than the ones specified that have been used are :

min_weight_fraction_leaf=0.0, max_features='auto', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None, bootstrap=True, oob_score=False, random_state=None, warm_start=False

4.3 Adaboost

An AdaBoost classifier begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases. We tested the classifier using DecisionTreeClassifier. But the results obtained were poor.

The computation time of the algorithm was also very high, on a dataset with 500 features and 9501 instances and 5-fold cross-validation. The code, depending on the set of parameters took upto one full day to run. The classifier was evaluated for various values of parameters :

DecisionTreeClassifier(max_depth=3), n_estimators=700, learning_rate=0.1

Class	Precision	Recall	f1-score	support
0.0	0.42	0.28	0.34	60
1.0	0.08	0.11	0.09	47
2.0	0.15	0.15	0.15	59
3.0	1.00	0.13	0.23	23
4.0	0.18	0.13	0.15	46
5.0	0.53	0.54	0.53	71
6.0	0.00	0.00	0.00	38
7.0	0.80	0.11	0.20	36
8.0	0.16	0.14	0.15	42
9.0	0.24	0.21	0.22	57
10.0	0.14	0.12	0.13	50
11.0	0.05	0.03	0.03	38
12.0	0.00	0.00	0.00	20
13.0	0.00	0.00	0.00	41
14.0	0.59	0.41	0.48	39
15.0	0.29	0.26	0.27	50
16.0	0.38	0.40	0.39	72
17.0	0.17	0.23	0.19	57
18.0	0.12	0.20	0.15	71
19.0	0.00	0.00	0.00	33
20.0	0.20	0.34	0.26	70
21.0	0.14	0.28	0.19	64
22.0	0.42	0.17	0.24	65
23.0	0.11	0.31	0.16	72
24.0	0.27	0.29	0.28	52
25.0	0.00	0.00	0.00	23
26.0	0.13	0.04	0.06	51
27.0	0.31	0.32	0.31	53
28.0	0.29	0.23	0.26	26
avg / total	0.25	0.22	0.21	1426

Figure 21: Adaboost Classificatiion Report

The test labels were predicted by using two methods :

- Taking the mode of the result of 5-fold cross-validation, ie, assigning the class predicted by majority of the folds.
- Summing up the class probabilities in each of the 5 folds for the predicted test labels and assigning it to the class having highest probability

Using the probabilities and DecisionTreeClassifier(max_depth=3), n_estimators=800, learning_rate=0.2

Class	precision	recall	f1-score	support
0.0	0.54	0.33	0.41	60
1.0	0.09	0.13	0.11	47
2.0	0.18	0.22	0.20	59
3.0	1.00	0.09	0.16	23
4.0	0.11	0.07	0.08	46
5.0	0.50	0.52	0.51	71
6.0	0.17	0.03	0.05	38
7.0	0.71	0.14	0.23	36
8.0	0.16	0.12	0.14	42
9.0	0.20	0.19	0.20	57
10.0	0.13	0.12	0.12	50
11.0	0.13	0.05	0.08	38
12.0	1.00	0.05	0.10	20
13.0	0.00	0.00	0.00	41
14.0	0.57	0.41	0.48	39
15.0	0.26	0.18	0.21	50
16.0	0.36	0.43	0.39	72
17.0	0.16	0.23	0.19	57
18.0	0.11	0.21	0.15	71
19.0	0.00	0.00	0.00	33
20.0	0.16	0.26	0.19	70
21.0	0.12	0.25	0.17	64
22.0	0.46	0.17	0.25	65
23.0	0.11	0.32	0.16	72
24.0	0.28	0.21	0.24	52
25.0	0.00	0.00	0.00	23
26.0	0.18	0.06	0.09	51
27.0	0.33	0.30	0.32	53
28.0	0.25	0.19	0.22	26
avg / total	0.26	0.21	0.21	1426

Figure 22: Adaboost Classificatiion Report

These are two of the best results obtained using the above classifier with a variety of parameters. Hence, this classifier was avoided during ensembling.

Reasons why Decision tree was chosen as the classifier for boosting :

- Theys are non-linear (Boosting with linear models doesn't usually work well)
- They are reasonably fast to train and classify
- By changing the depth, we have a simple and easy control over the bias/variance trade off

Reasons as to why Adaboost might have failed :

- The noise level in the data: AdaBoost is particularly prone to overfitting on noisy datasets
- The dimensionality of the data: We know that generally, we experience overfitting more in high dimensional spaces, and AdaBoost can also suffer in that respect, as it is simply a linear combination of classifiers which themselves suffer from the problem
- Multi-label Classification: It is not an inherently multiclass classifier in sklearn

4.4 Gradient Boost

Gradient boosting is a machine learning technique for regression and classification problems, which produces a prediction model in the form of an ensemble of weak prediction models, typically decision trees. It builds the model in a stage-wise fashion like other boosting methods do, and it generalizes them by allowing optimization of an arbitrary differentiable loss function.

We worked a lot using gradient boosting with differnt sets of parameters. It was giving us better results than Adaboost and the performance on the cross-validation sets were upto what was obtained using Random Forests. The dataset used was the feature reduced dataset (PCA) comprising of 500 dimensions

The test labels were predicted by using two methods :

- Taking the mode of the result of 5-fold cross-validation, ie, assigning the class predicted by majority of the folds.
- Summing up the class probabilities in each of the 5 folds for the predicted test labels and assigning it to the class having highest probability

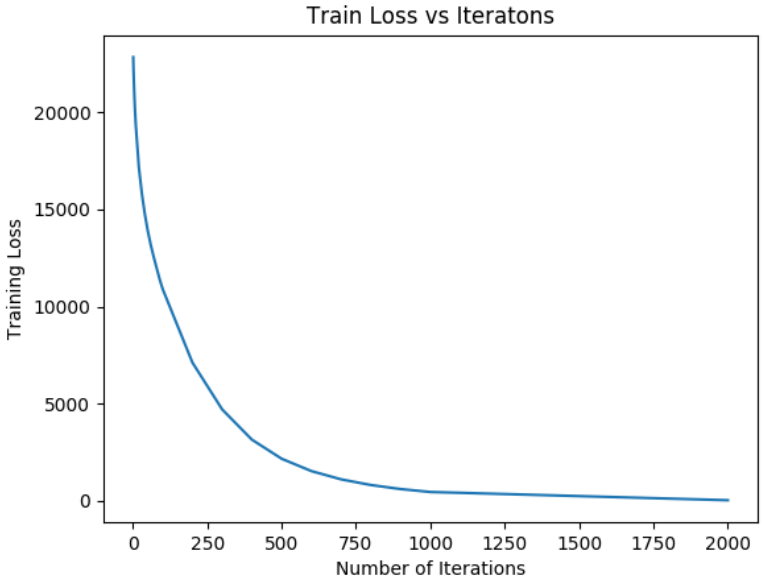


Figure 23: Training Loss

The three best classifiers obtained among the lot :

n_estimators=1500, learning_rate=0.1, max_depth=2, random_state=0, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.48	0.58	0.53	60
1.0	0.17	0.13	0.15	47
2.0	0.37	0.31	0.33	59
3.0	0.22	0.09	0.12	23
4.0	0.23	0.15	0.18	46
5.0	0.47	0.76	0.58	71
6.0	0.12	0.08	0.10	38
7.0	0.71	0.61	0.66	36
8.0	0.15	0.12	0.13	42
9.0	0.34	0.32	0.33	57
10.0	0.28	0.24	0.26	50
11.0	0.11	0.05	0.07	38
12.0	0.00	0.00	0.00	20
13.0	0.22	0.12	0.16	41
14.0	0.76	0.64	0.69	39
15.0	0.23	0.18	0.20	50
16.0	0.49	0.68	0.57	72
17.0	0.21	0.25	0.22	57
18.0	0.14	0.23	0.17	71
19.0	0.19	0.09	0.12	33
20.0	0.27	0.47	0.35	70
21.0	0.16	0.19	0.17	64
22.0	0.43	0.37	0.40	65
23.0	0.16	0.24	0.19	72
24.0	0.18	0.19	0.19	52
25.0	0.29	0.09	0.13	23
26.0	0.16	0.14	0.15	51
27.0	0.55	0.40	0.46	53
28.0	0.70	0.27	0.39	26
avg / total	0.30	0.31	0.30	1426

Figure 24: Gradient Boosting: Classification Report

n_estimators=2000, learning_rate=0.15, max_depth=2, random_state=0, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.44	0.52	0.48	60
1.0	0.21	0.15	0.17	47
2.0	0.34	0.32	0.33	59
3.0	0.22	0.09	0.12	23
4.0	0.22	0.15	0.18	46
5.0	0.47	0.77	0.59	71
6.0	0.16	0.11	0.13	38
7.0	0.63	0.61	0.62	36
8.0	0.16	0.12	0.14	42
9.0	0.35	0.33	0.34	57
10.0	0.29	0.24	0.26	50
11.0	0.09	0.05	0.07	38
12.0	0.00	0.00	0.00	20
13.0	0.28	0.17	0.21	41
14.0	0.74	0.64	0.68	39
15.0	0.24	0.20	0.22	50
16.0	0.51	0.71	0.59	72
17.0	0.21	0.28	0.24	57
18.0	0.14	0.20	0.16	71
19.0	0.12	0.06	0.08	33
20.0	0.28	0.46	0.34	70
21.0	0.19	0.23	0.21	64
22.0	0.45	0.37	0.41	65
23.0	0.19	0.25	0.22	72
24.0	0.23	0.23	0.23	52
25.0	0.22	0.09	0.12	23
26.0	0.17	0.14	0.15	51
27.0	0.51	0.43	0.47	53
28.0	0.78	0.27	0.40	26
avg / total	0.31	0.32	0.30	1426

Figure 25: Gradient Boosting: Classification Report

n_estimators=2000, learning_rate=0.05, max_depth=2, random_state=0, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.55	0.55	0.55	60
1.0	0.19	0.15	0.17	47
2.0	0.31	0.27	0.29	59
3.0	0.33	0.13	0.19	23
4.0	0.18	0.15	0.16	46
5.0	0.50	0.76	0.61	71
6.0	0.12	0.05	0.07	38
7.0	0.50	0.44	0.47	36
8.0	0.07	0.05	0.06	42
9.0	0.34	0.40	0.37	57
10.0	0.22	0.22	0.22	50
11.0	0.05	0.03	0.03	38
12.0	0.33	0.05	0.09	20
13.0	0.21	0.10	0.13	41
14.0	0.67	0.74	0.71	39
15.0	0.40	0.34	0.37	50
16.0	0.43	0.61	0.51	72
17.0	0.23	0.25	0.24	57
18.0	0.14	0.27	0.19	71
19.0	0.15	0.09	0.11	33
20.0	0.27	0.34	0.30	70
21.0	0.18	0.28	0.22	64
22.0	0.49	0.45	0.47	65
23.0	0.15	0.18	0.16	72
24.0	0.27	0.21	0.24	52
25.0	0.40	0.09	0.14	23
26.0	0.24	0.22	0.23	51
27.0	0.44	0.43	0.44	53
28.0	0.30	0.12	0.17	26
avg / total	0.30	0.31	0.30	1426

Figure 26: Gradient Boosting: Classification Report

Some of the other results obtained are as follows :

Main Parameters			Average Estimates		
n_estimators	**max_depth**	**learning rate**	**Precision**	**Recall**	**F-measure**
1000	1	0.5	0.21	0.23	0.21
2000	1	0.5	0.21	0.23	0.21
3000	1	0.25	0.26	0.26	0.25
4000	1	0.15	0.27	0.27	0.26
6000	1	0.15	0.26	0.27	0.26
6000	3	0.5	0.19	0.20	0.19
9000	1	0.2	0.26	0.26	0.25
2000	3	0.1	0.27	0.30	0.28
4000	3	0.09	0.28	0.30	0.29
2000	3	0.05	0.30	0.31	0.29

Observed Disadvantages :

- The code took over a day to run, probably because of the fact that trees are built sequentially and the data has high dimensionality (500 dimensions - PCA reduced)
- There are 3 main parameters to train : learning rate, depth of tree and number of trees. Now each of these parameters should be tuned to get a good fit. This was difficult to achieve keeping in mind the time constraint and the available resources and the fact that the classifier is prone to overfitting

4.5 Naive Bayes

The Naive Bayes Classifier technique is based on the Bayesian theorem and is suitable when the dimensionality of the inputs is high. The different naive Bayes classifiers differ mainly by the assumptions they make regarding the distribution of $P(x_i \mid y)$.

4.5.1 Gaussian

Here we are dealing with continuous data, so we assume that the continuous values associated with each class are distributed according to a Gaussian distribution. For the training data containing a continuous attribute x, we first segment the data by the class, and then

compute the mean and variance of x in each class.

The sklearn Naive Bayes Packages was tested on the features for the reduced feature space (500 dimensions) (by using PCA) and the following results were observed :

Class	Precision	Recall	f1-score	support
0.0	0.45	0.34	0.39	101
1.0	0.26	0.14	0.18	78
2.0	0.41	0.11	0.17	99
3.0	0.18	0.05	0.08	38
4.0	0.22	0.13	0.17	76
5.0	0.48	0.31	0.38	118
6.0	0.23	0.11	0.15	63
7.0	0.51	0.34	0.41	61
8.0	0.11	0.07	0.09	69
9.0	0.11	0.48	0.19	95
10.0	0.09	0.06	0.07	83
11.0	0.07	0.05	0.06	63
12.0	0.17	0.31	0.22	32
13.0	0.13	0.15	0.14	68
14.0	0.70	0.22	0.33	64
15.0	0.28	0.26	0.27	84
16.0	0.44	0.28	0.34	120
17.0	0.13	0.12	0.13	96
18.0	0.07	0.05	0.06	119
19.0	0.11	0.07	0.09	55
20.0	0.16	0.15	0.16	117
21.0	0.11	0.23	0.15	106
22.0	0.48	0.32	0.38	109
23.0	0.12	0.07	0.09	121
24.0	0.21	0.23	0.22	87
25.0	1.00	0.27	0.43	37
26.0	0.29	0.18	0.22	85
27.0	0.28	0.27	0.28	88
28.0	0.06	0.32	0.10	44
avg / total	0.26	0.20	0.21	2376

Figure 27: Naive Bayes: Classification Report

Due to the poor results obtained, this further work was not done on the classifier.
One possible reason why naive bayes does'nt work well: It's basic assumption on strong feature independence must have been violated. Possibly there is still some dependencies among the features that we were not able to eliminate.

4.6 SVM

Support Vector Machines (SVMs) are supervised learning models with associated learning algorithms that analyze data used for classification and regression. In addition to performing linear classification, SVMs can efficiently perform a non-linear classification using what is called the kernel trick, implicitly mapping their inputs into high-dimensional feature spaces.

4.6.1 Sigmoid Kernel

The sigmoid kernel function is of the form :

$$tanh(\gamma < x, x' > +r)$$

But our input dimension is 2600 with 9501 instances and hence, even after running the algorithm for more than 3 days, convergence was not obtained and hence the algorithm was discarded and not worked on further.

4.6.2 Gaussian Kernel

The Gaussian kernel function is of the form :

$$\exp(-\gamma||x - x'||^2)$$

The code ran for a long time, and did'nt seem to be close to finishing any time soon.

Probable reason:SVM when used for multiclass classification uses either one of 'ovr'-one-vs-rest or 'ovo'- one-vs-one. When running for a problem with 29 classes, the number of combinations required in ovo is too much, and the class imbalance in case of ovr is too high. This lead us to rule out SVM as a feasible method of solving the given problem.

4.7 XGBoost

XGBoost is short for âĂIJExtreme Gradient BoostingâĂİ, as it is an optimized distributed gradient boosting library. It is an open-source software library which provides the gradient boosting framework.

Advantages of XGBoost over gradient boosted trees :

- Parallel Computing: It is enabled with parallel processing (using OpenMP); i.e., when we run xgboost, by default, it would use all the cores of our laptop.

- Regularization: GBM has no provision for regularization. Regularization is a technique used to avoid overfitting in linear and tree-based models.
- Tree Pruning: Unlike GBM, where tree pruning stops once a negative loss is encountered, XGBoost grows the tree upto max_depth and then prune backward until the improvement in loss function is below a threshold.

XGBoost was implemented on the feature reduced dataset comprising of 500 PCA reduced features. Some of the best results obtained are as follows :

n_estimators=1000

Class	Precision	Recall	f1-score	support
0.0	0.49	0.57	0.52	60
1.0	0.14	0.11	0.12	47
2.0	0.31	0.29	0.30	59
3.0	0.60	0.13	0.21	23
4.0	0.36	0.17	0.24	46
5.0	0.50	0.82	0.62	71
6.0	0.40	0.11	0.17	38
7.0	0.42	0.47	0.45	36
8.0	0.10	0.05	0.06	42
9.0	0.38	0.51	0.43	57
10.0	0.26	0.20	0.22	50
11.0	0.25	0.05	0.09	38
12.0	1.00	0.05	0.10	20
13.0	0.33	0.22	0.26	41
14.0	0.70	0.67	0.68	39
15.0	0.45	0.26	0.33	50
16.0	0.53	0.75	0.62	72
17.0	0.12	0.12	0.12	57
18.0	0.19	0.37	0.25	71
19.0	0.40	0.18	0.25	33
20.0	0.30	0.43	0.35	70
21.0	0.10	0.16	0.12	64
22.0	0.54	0.46	0.50	65
23.0	0.22	0.29	0.25	72
24.0	0.33	0.33	0.33	52
25.0	0.62	0.22	0.32	23
26.0	0.33	0.29	0.31	51
27.0	0.50	0.51	0.50	53
28.0	0.30	0.12	0.17	26
avg / total	0.36	0.34	0.33	1426

Figure 28: XGBoost: Classification Report

n_estimators=1500

Class	Precision	Recall	f1-score	support
0.0	0.47	0.65	0.55	60
1.0	0.27	0.21	0.24	47
2.0	0.42	0.34	0.37	59
3.0	0.20	0.04	0.07	23
4.0	0.42	0.24	0.31	46
5.0	0.47	0.86	0.61	71
6.0	0.15	0.05	0.08	38
7.0	0.69	0.67	0.68	36
8.0	0.14	0.05	0.07	42
9.0	0.31	0.32	0.31	57
10.0	0.29	0.20	0.24	50
11.0	0.00	0.00	0.00	38
12.0	1.00	0.10	0.18	20
13.0	0.38	0.24	0.30	41
14.0	0.71	0.74	0.72	39
15.0	0.30	0.26	0.28	50
16.0	0.55	0.75	0.63	72
17.0	0.23	0.25	0.24	57
18.0	0.14	0.21	0.16	71
19.0	0.36	0.15	0.21	33
20.0	0.24	0.40	0.30	70
21.0	0.15	0.23	0.19	64
22.0	0.53	0.51	0.52	65
23.0	0.25	0.29	0.27	72
24.0	0.28	0.33	0.30	52
25.0	0.67	0.17	0.28	23
26.0	0.35	0.31	0.33	51
27.0	0.50	0.43	0.46	53
28.0	0.46	0.23	0.31	26
avg / total	0.36	0.35	0.34	1426

Figure 29: XGBoost: Classification Report

n_estimators=2000

Class	Precision	Recall	f1-score	support
0.0	0.47	0.60	0.53	60
1.0	0.20	0.19	0.20	47
2.0	0.31	0.25	0.28	59
3.0	0.33	0.13	0.19	23
4.0	0.30	0.17	0.22	46
5.0	0.47	0.79	0.59	71
6.0	0.25	0.11	0.15	38
7.0	0.40	0.53	0.46	36
8.0	0.09	0.05	0.06	42
9.0	0.38	0.47	0.42	57
10.0	0.26	0.20	0.23	50
11.0	0.13	0.05	0.08	38
12.0	0.67	0.10	0.17	20
13.0	0.29	0.24	0.27	41
14.0	0.62	0.72	0.67	39
15.0	0.38	0.24	0.29	50
16.0	0.55	0.74	0.63	72
17.0	0.13	0.14	0.14	57
18.0	0.19	0.30	0.23	71
19.0	0.45	0.27	0.34	33
20.0	0.26	0.34	0.30	70
21.0	0.09	0.11	0.10	64
22.0	0.56	0.48	0.52	65
23.0	0.27	0.29	0.28	72
24.0	0.33	0.33	0.33	52
25.0	0.55	0.26	0.35	23
26.0	0.34	0.27	0.30	51
27.0	0.46	0.55	0.50	53
28.0	0.22	0.15	0.18	26
avg / total	0.34	0.34	0.33	1426

Figure 30: XGBoost: Classification Report

Among the classifiers, other than MLP that was used, this gave the highest results and hence, it was considered during ensembling.

4.8 Voting Classifier

The idea behind the VotingClassifier is to combine conceptually different machine learning classifiers and use a majority vote or the average predicted probabilities (soft vote) to predict the class labels. Such a classifier can be useful for a set of equally well performing model in order to balance out their individual weaknesses.

We have used soft voting, as it returns the class label as argmax of the sum of predicted probabilities. The saved best models (using pickle) of Random Forest, Gradient Boosting and XGB Boosting were used for building the Voting classifier. The feature reduced dataset comprising of 500 dimensions were used during the computation. The computation proved to be very expensive and took more than 2 days to reach convergence when implemented using 5-fold cross-validation.

The results obtained are as follows :

- XGBoost : n_estimators=1500
- Gradient Boosting : n_estimators=1500, learning_rate=0.1, max_depth=2, random_state=0, verbose=1
- Random Forest : n_estimators=2000, max_depth=25, criterion='gini', min_samples_split=2, min_samples_leaf=1, n_jobs=-1, verbose=1

Class	Precision	Recall	f1-score	support
0.0	0.48	0.60	0.53	60
1.0	0.23	0.19	0.21	47
2.0	0.46	0.36	0.40	59
3.0	0.40	0.09	0.14	23
4.0	0.37	0.22	0.27	46
5.0	0.47	0.87	0.61	71
6.0	0.19	0.11	0.14	38
7.0	0.69	0.67	0.68	36
8.0	0.22	0.12	0.15	42
9.0	0.31	0.32	0.31	57
10.0	0.24	0.16	0.19	50
11.0	0.07	0.03	0.04	38
12.0	0.25	0.05	0.08	20
13.0	0.22	0.12	0.16	41
14.0	0.76	0.67	0.71	39
15.0	0.33	0.24	0.28	50
16.0	0.52	0.71	0.60	72
17.0	0.23	0.25	0.24	57
18.0	0.13	0.21	0.16	71
19.0	0.08	0.03	0.04	33
20.0	0.27	0.47	0.35	70
21.0	0.16	0.22	0.19	64
22.0	0.45	0.40	0.42	65
23.0	0.17	0.22	0.19	72
24.0	0.27	0.31	0.29	52
25.0	0.33	0.09	0.14	23
26.0	0.30	0.27	0.29	51
27.0	0.51	0.42	0.46	53
28.0	0.55	0.23	0.32	26
avg / total	0.33	0.33	0.32	1426

Figure 31: Voting Classifier: Classification Report

5 OUR INSIGHTS ABOUT THE DATASET

Here are some points we have concluded about the data (both major and minor)

- From our experience with the data, based on observed patterns of missing features and variance patterns, it looks like the data was artificially generated.
- The data has already been scaled and normalised. But even then, the number of ones varies from 3-9 percent of each feature . Class 7 seems to have the highest number of features per data point taking the value 1. This could been caused by some kind of overflow/threshold.
- We also feel that the number of important features in the dataset are relatively low in comparison to the the number of features given. This conclusion is supported by our observations on PCA variances and trends in neural net result.
- The data generation could have been achieved by generating a couple of features, and performing some complex transformations on them and appending the results to the dataset as new features.
- Also, when visualising the features, one can directly conclude that linear models are not feasible, as histogram plots of features show varying percentages of each classes data points in every range. Also, the data distribution within most of the features is skewed.

6 FINAL MODEL

Till now, we have presented and explained all the models that we tried out. Here is a description of the final model that we have submitted:

- First the data was cleaned. That is we threw away the first 500 features.
- Next we performed PCA and extracted some features. Mostly in the set of 50,60,80,100,200
- Then we passed the extracted features to a neural net. We made sure that the first layer of the neural net had a much larger number of features than the number of inputs.
- Finally, we analysed the classification reports of multiple models and put together models that collectively had a decent fscore in all the classes. The mode of the models at each data point was computed.
- In order to make our model more robust, we combined many models with fscores ranging from 0.38 to 0.42, where some of their individual class fscores were higher than those in other models
- We then submitted our result :)

REFERENCES

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.VotingClassifier.html

http://scikit-learn.org/stable/modules/ensemble.html#voting-classifier

https://cran.r-project.org/web/packages/xgboost/xgboost.pdf

http://scikit-learn.org/stable/modules/svm.html

https://www.youtube.com/watch?v=1SxPbhaK_uc&list=PL1xHD4vteKYVpaIiy295pg6_SY5qznc77&index=68

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html

http://scikit-learn.org/stable/modules/multiclass.html

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html#sklearn.ensemble.ExtraTreesClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html#sklearn.naive_bayes.GaussianNB

http://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html#sklearn.neighbors.KNeighborsClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html#sklearn.discriminant_analysis.LinearDiscriminantAnalysis

http://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#sklearn.neural_network.MLPClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html#sklearn.ensemble.RandomForestClassifier

http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

http://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html#sklearn.ensemble.GradientBoostingClassifier