Classification For Predicting Multi-level Product Categories

Latha Airodi (Group 45)

April 17, 2022

Abstract

The purpose of this project is to apply various classification algorithms to the Otto Group product dataset and build a predictive classification model that is able to distinguish products between 9 main product categories.

1 Introduction

Classification is the process of recognizing, understanding and grouping of objects and ideas into specific categories or sub-populations. With the help of these pre-categorized training data-sets, classification machine learning programs leverage a wide range of algorithms to classify future data-sets into respective and relevant categories.

This project explores various classification methods and tries to find the best model for classifying a group of products. The dataset used is from the Otto Group Product Classification Challenge on Kaggle. Some of the models that will be explored are decision trees, neural network, gradient boosting model. All these models will then be bench marked against an ElasticNet classifier (Teisseyre, 2017). The ultimate goal is to get a better understanding of the various models in order and apply to a work (real world) scenario (classifying customer preferences for window coverings).

1.1 The Otto Group Classification Challenge

The Otto Group is one of the world's biggest e-commerce companies, with subsidiaries in more than 20 countries, including Crate & Barrel (USA), Otto.de (Germany) and 3 Suisses (France). They sell millions of products worldwide every day, with several thousand products being added to their product line. They sponsored a Kaggle competition seeking a way to more accurately group their products into product lines for further business analysis and decision-making.

The dataset contains over than 200,000 products with 93 features and the objective is to build a predictive model which is able to distinguish between the main product categories. More information about the dataset can be found here - Otto Group Product Classification Challenge

2 Project Objectives

The project aims to answer the following questions,

- Compare the prediction accuracy of the different models. Does a particular model perform better than other models? The models that will be built are Random Forest, Naive Bayes, Neural Network, Logistic regression (multinomial), KNN, SVM, XGBoost, Elastic-net.
- How does each model compare against the benchmark Elastic-net classifier?
- Is there a category (or categories) that are similar enough that all models struggle to classify accurately? Is so, why and how to address it?
- It is better to apply dimension reduction and reduce data to its n principal components or is it better to use the n most important features (Permutation Feature Importance - PFI) (Altmann et al., 2010)?

3 Exploratory Data Analysis

3.1 Data-set Overview

The training data-set contains 61,878 rows and 94 columns. Each row represents one product, with each column (except the last column) representing one feature. The last column specifies the classification for the product (9 classes). The test data-set contains 144,368 rows and 93 columns, the target class has not been provided for the test data-set. Per the competition rules, submissions are evaluated using the multi-class logarithmic loss and for each

product, a set of predicted probabilities (one for every category) must be submitted. But, in this project, we will be evaluating the models based on how well the model predicts the product class for a particular product.

3.2 Summary Statistics

3.2.1 Response variable

The response variable is named "target" and is a factor variable with 9 classes ("Class_1", "Class_2"..,"Class_9"). There are no missing values in this column missing values. The classes are not balanced, with 26% of the data belonging to Class_2 and 23% of the data belonging to Class_6 and Class_1 with the least frequently-observed. This indicates that the data is biased toward certain classes and it would be better to apply some method of sampling when fitting models.

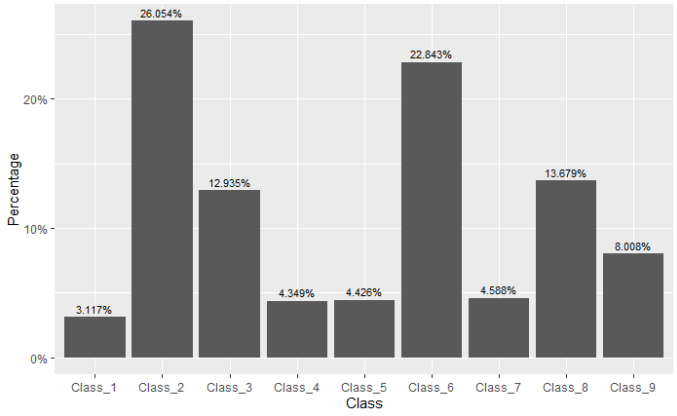


Figure 1: Class Distribution

3.2.2 Predictor variables

There are 93 numerical "feature" columns (with obfuscated values) implicitly describing each product, Since the features and the classes were labeled simply as feat_1, feat_2, class_1, class_2, etc., it is not possible to use any domain knowledge or interpret anything from the real world. The table below shows the summary stats for a few features. Additional statistics like mean of means, median of medias etc. were calculated to get a better understanding of the features. The empirical probabilities of each feature provided an indication on which feature was more likely to have non-zero values - feat_24 had the max probability of 0.64 and feature_6 had the least.

Variable	N	Mean	Std. Dev.	Min	Pctl. 25	Pctl. 75	Max
feat_1	61878	0.387	1.525	0	0	0	61
feat_2	61878	0.263	1.252	0	0	0	51
feat_3	61878	0.901	2.935	0	0	0	64
feat_5	61878	0.071	0.439	0	0	0	19
feat_88	61878	0.875	2.115	0	0	1	30
feat_89	61878	0.458	1.527	0	0	0	61
feat_90	61878	0.812	4.598	0	0	0	130
feat_91	61878	0.265	2.046	0	0	0	52
feat_92	61878	0.38	0.982	0	0	0	19
feat_93	61878	0.126	1.202	0	0	0	87

Figure 2: Summary Stats

STATISTIC	MIN	MEDIAN	MAX
Mean (of means)	0.0257	0.4865	2.8977
Median (of medians)	0	0	1
Standard Deviation	0.2153	1.9013	5.7832
Emp Probabilities	0.0189 (feature 6)	0.1680	0.6432 (feature 24)

Figure 3: Additional Stats

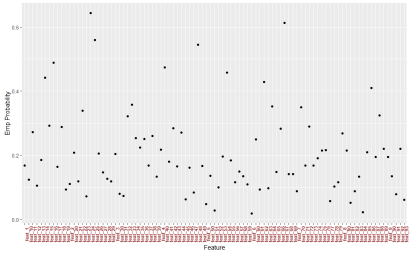


Figure 4: Empirical Probabilities

3.3 Feature Analysis

The data for the features is highly skewed and each feature has a high percentage of zeros. Some of the features have a limited number of values and can be treated as categorical values. A box-plot of the features against the target shows that there are a large number of outliers. The density plots show that there is no clear separation of product classes and hence unable to discriminate between the classes.

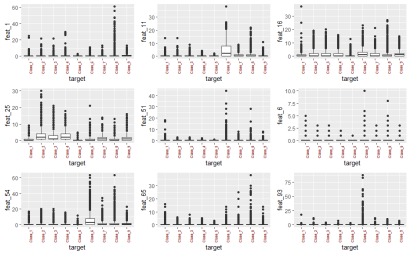


Figure 5: Feature Boxplots

	means	vars	skews
feat_1	3.866802e-01	2.326630e+00	1.289969e+01
feat_2	2.630660e-01	1.567688e+00	1.132324e+01
feat_3	9.014674e-01	8.613156e+00	5.304938e+00
feat_4	7.790814e-01	7.772970e+00	8.634979e+00
feat_5	7.104302e-02	1.926353e-01	1.330883e+01
feat_6	2.569572e-02	4.636850e-02	1.282835e+01
feat_7	1.937037e-01	1.061109e+00	1.275644e+01
feat_8	6.624325e-01	5.088496e+00	1.095535e+01
feat_9	1.011296e+00	1.207439e+01	4.522726e+00
feat_10	2.639064e-01	1.173625e+00	8.778516e+00
feat_11	1.252869e+00	9.255788e+00	3.031503e+00
feat_12	1.408740e-01	3.215896e-01	1.626632e+01
feat_13	4.809787e-01	4.059006e+00	1.119058e+01
feat_14	1.696693e+00	1.000591e+01	2.859573e+00
feat_15	1.284398e+00	1.491687e+01	4.415401e+00
feat_16	1.413459e+00	4.955801e+00	2.484998e+00

Figure 6: Skewness in each feature

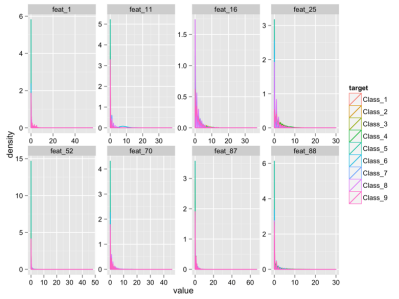


Figure 7: Feature Density Plot

In order to help improve class separability, two synthetic variables, count of all non zero entries per row, and the total of all the non zero entries per row was added to the dataset. Adding these two additional features improved class separability as shown by the below plots.

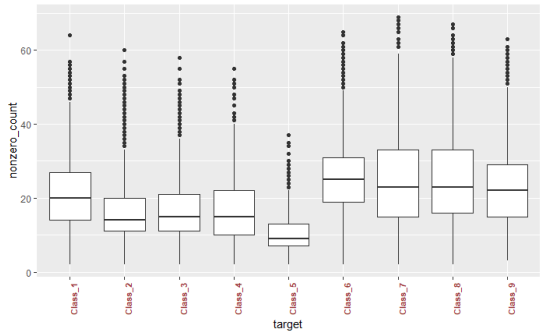


Figure 8: Box plot for non_zero_count

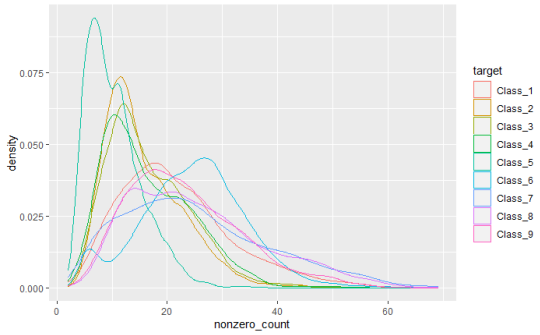


Figure 9: Density plot for non_zero_count

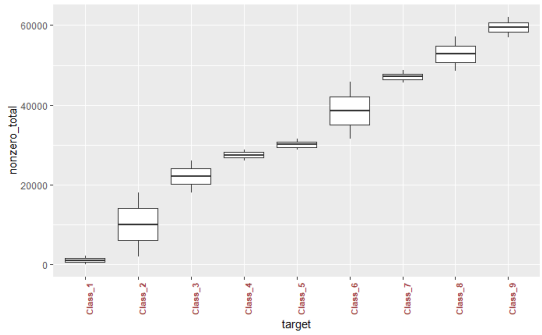


Figure 10: Box plot for non_zero_count

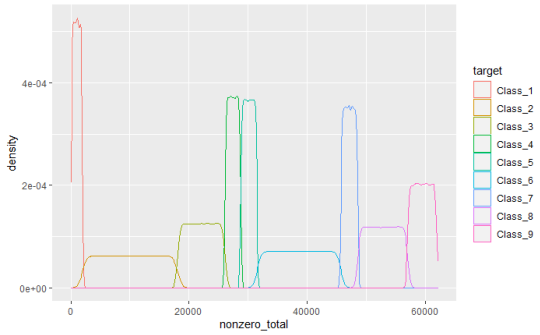


Figure 11: Density plot for non_zero_count

3.3.1 Correlations

Looking at the correlation plot indicates multi-colinearity between the features, so it could be possible to eliminate or combine two features with high correlations.

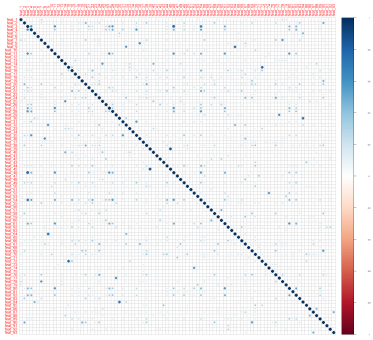


Figure 12: Correlation Plot

Taking a look at the correlation matrix for each class separately, we further see the correlation between the features and that a small section of features have higher parity in explaining the dataset than others.

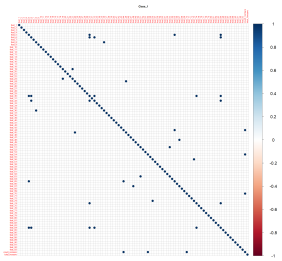


Figure 13: Correlation for Class 1

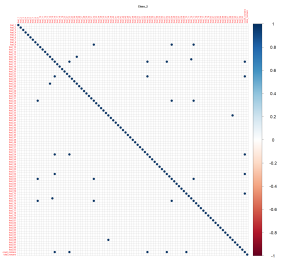


Figure 14: Correlation for Class 2

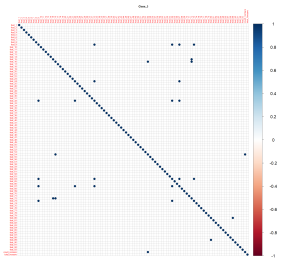


Figure 15: Correlation for Class 3

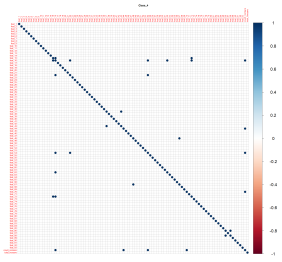


Figure 16: Correlation for Class 4

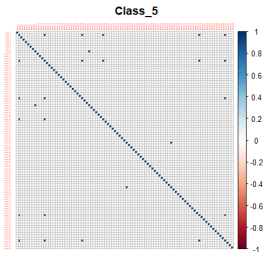


Figure 17: Correlation for Class 5

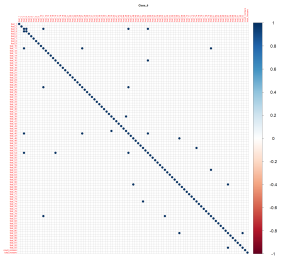


Figure 18: Correlation for Class 6

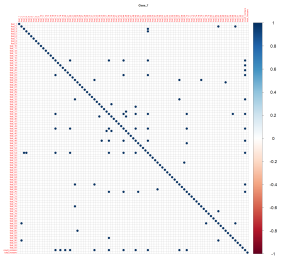


Figure 19: Correlation for Class 7

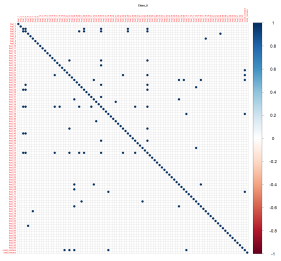


Figure 20: Correlation for Class 8

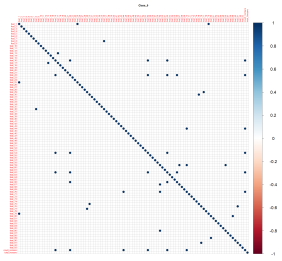


Figure 21: Correlation for Class 9

3.4 Feature Selection

Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model. Feature selection is also related to dimensionally reduction techniques in that both methods seek fewer input variables to a predictive model. The difference is that feature selection select features to keep or remove from the dataset, whereas dimensionality reduction create a projection of the data resulting in entirely new input features. (Brownlee)

For this project, we used the dimension reduction method PCA (principle component analysis) and random forest to analyze the importance of each feature and to determine which features to include in the model.

3.4.1 PCA

Principal component analysis and resulting scree plot revealed a "cutoff point" of around 68 components. This threshold indicates that in attempting to capture the collective variability among all feature variables, a significant portion of the variability can be explained with only 68 principal components rather than the original 93 features. Note that both synthetic variables were picked up by PCA as important features contributing to PC1.

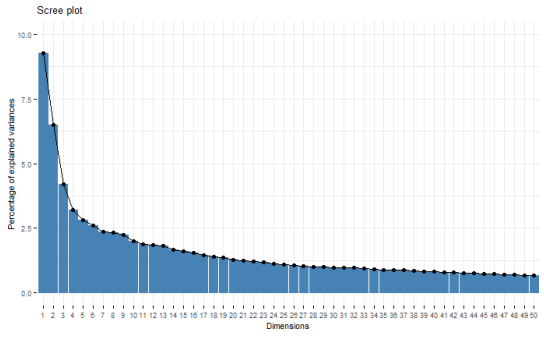


Figure 22: PCA Scree Plot

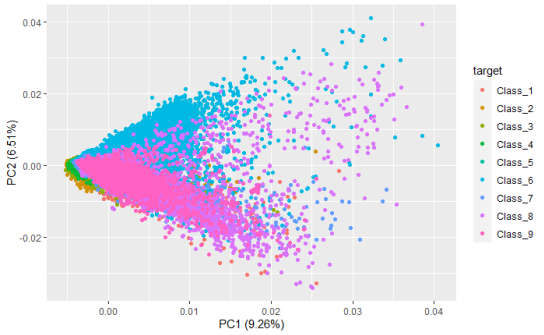


Figure 23: PC1 vs PC2

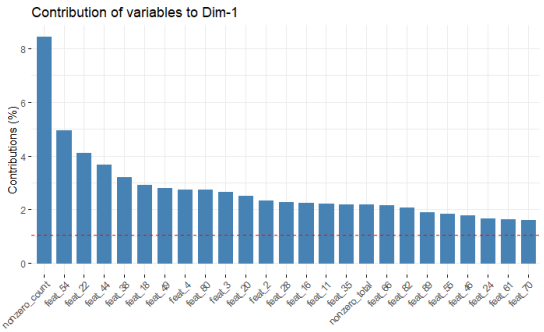


Figure 24: Feature contribution to PC1

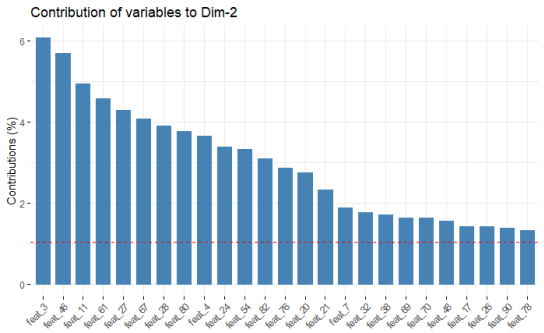


Figure 25: Feature contribution to PC2

3.4.2 Random Forest

Random forest are widely used as a feature selection method because the tree-based strategies used by random forests naturally ranks by how well they improve the purity of the node. Nodes with the greatest decrease in impurity happen at the start of the trees, while notes with the least decrease in impurity occur at the end of trees. Thus, by pruning trees below a particular node, we can create a subset of the most important features.

The below figures show the most important features by Gini Index as well as by accuracy. Here again, we note that the two synthetic variables were among the most important variables, so these two variables will definitely be included in all the models.

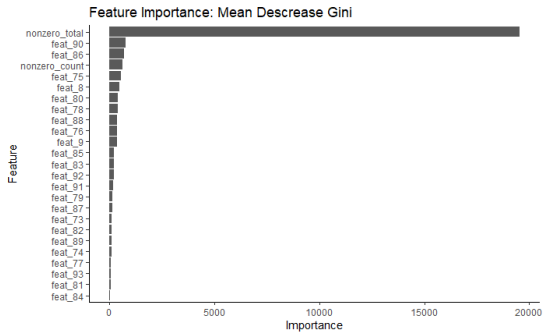


Figure 26: Mean Decrease Gini

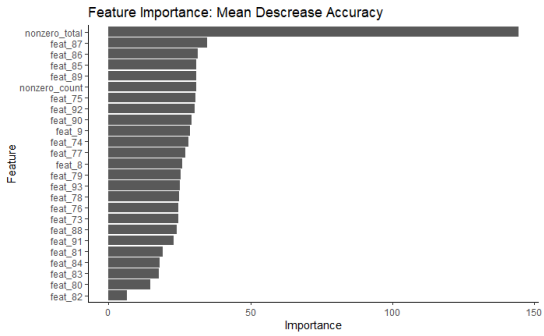


Figure 27: Mean Decrease Accuracy

4 Models

4.1 Data Pre-processing

Before creating various models, the data-set was altered based on the results of the EDA from the previous section.

- The test data-set provided by the competition was missing the target column, hence it was not possible to use that test data-set. Instead, the train data-set was divided into train (70%) and test (30%). The models were trained and evaluated using the split train data.
- To address class separability two new synthetic features were added (nonzero_count – count of the non-zero entries in each row, nonzero_total – total of the non-zero entries in each row)
- Scaled all the features
- Class imbalance was however not addressed, so there could be an issue with model over-fitting.
- Based on PCA and Random Forest only the top 68 important features by included in the models

4.2 Models

4.2.1 Naive Bayes

Naïve Bayes algorithm is a machine learning supervised classification technique based on Bayes theorem with strong independence assumptions between the features. Since a fundamental assumption of Naive Bayes is that the features are independent and contribute equally to the classification, it was no surprise that it performed poorly with this dataset. The accuracy without and with cross validation was around 74%. But, the execution time for building a Naive Bayes model was the fastest compared to all other models (4-10 seconds).

Confusion Matrix and Statistics

Prediction	Class_1	Class_2	Class_3	Class_4	Class_5	Class_6	Class_7	Class_8	Class_9
Class_1	305	5	8	4	4	205	37	477	50
Class_2	114	4400	1145	37	14	126	172	142	5
Class_3	0	285	923	38	6	50	11	0	0
Class_4	0	0	199	677	4	32	0	0	107
Class_5	0	0	49	76	800	90	0	0	21
Class_6	5	11	22	9	3	3547	38	112	22
Class_7	21	118	1	2	0	54	570	121	2
Class_8	1	8	12	3	0	58	28	1404	24
Class_9	131	2	0	0	0	33	0	322	1261

Overall Statistics

Accuracy : 0.7481
95% CI : (0.7418, 0.7543)
No Information Rate : 0.2601
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.6968

Mcnemar's Test P-Value : NA

Figure 28: Confusion Matrix(Naive Bayes)

	Sensitivity	Specificity
Class: Class_1	0.5285962	0.9560769
Class: Class_2	0.9111617	0.8722149
Class: Class_3	0.3912675	0.9759319
Class: Class_4	0.8002364	0.9806965
Class: Class_5	0.9626955	0.9866907
Class: Class_6	0.8455304	0.9845490
Class: Class_7	0.6658879	0.9819845
Class: Class_8	0.5446082	0.9916171
Class: Class_9	0.8451743	0.9714135

Figure 29: Sensitivity and Specificity by Class (Naive Bayes)

4.2.2 KNN

KNN is a model that classifies data points based on the points that are most similar to it. It uses test data to make an "educated guess" on what an unclassified point should be classified as. The KNN model with this dataset was slightly better than the Naive Bayes model with an accuracy score of around 84%. However, building this model was extremely slow, the model with cross validation took around 3 hours to complete. It might be better to use KNN for feature engineering to create meta features and use its output as another feature that xgboost or another more competitive model could use as an input.

Confusion Matrix and Statistics

Prediction	Class_1	Class_2	Class_3	Class_4	Class_5	Class_6	Class_7	Class_8	Class_9
Class_1	432	8	0	0	0	3	2	1	3
Class_2	61	4299	728	158	5	8	12	7	2
Class_3	1	474	1521	320	10	16	56	7	0
Class_4	0	19	77	327	5	11	10	2	0
Class_5	0	10	2	9	809	6	3	1	1
Class_6	25	2	5	22	0	3976	60	89	48
Class_7	7	14	19	6	1	67	620	40	16
Class_8	18	2	4	2	1	69	75	2365	77
Class_9	33	1	3	2	0	39	18	66	1345

Overall Statistics

Accuracy : 0.8454
95% CI : (0.8402, 0.8506)
No Information Rate : 0.2601
P-Value [Acc > NIR] : < 2.2e-16

Kappa : 0.8129

Mcnemar's Test P-Value : NA

Figure 30: Confusion Matrix (KNN)

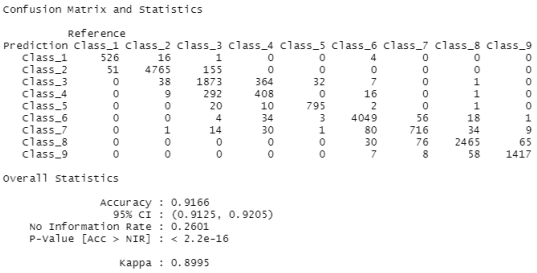
	Sensitivity	Specificity
Class: Class_1	0.7487002	0.9990548
Class: Class_2	0.8902464	0.9285714
Class: Class_3	0.6447647	0.9454456
Class: Class_4	0.3865248	0.9930011
Class: Class_5	0.9735259	0.9981954
Class: Class_6	0.9477950	0.9825306
Class: Class_7	0.7242991	0.9903993
Class: Class_8	0.9173778	0.9844855
Class: Class_9	0.9014745	0.9905102

Figure 31: Sensitivity and Specificity by Class(KNN)

4.2.3 Logistic Regression (multinomial

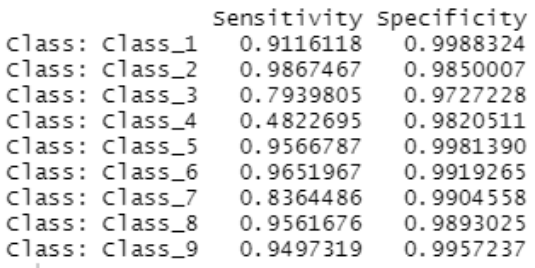
Multinomial logistic regression is an extension of logistic regression that adds native support for multi-class classification problems. The multinomial logistic regression algorithm is an extension to the logistic regression model that

involves changing the loss function to cross-entropy loss and predict probability distribution to a multinomial probability distribution to natively support multi-class classification problems. Multinomial regression with this data-set performed well with an accuracy (cross validation) of 91% and took around 26 minutes to build the model.



```
Confusion Matrix and Statistics

          Reference
Prediction Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_1     526      16       1       0       0       4       0       0       0
   Class_2      51    4765     155       0       0       0       0       0       0
   Class_3       0      38    1873     364      32       7       0       1       0
   Class_4       0       9     292     408       0      16       0       1       0
   Class_5       0       0      20      10     795       2       0       1       0
   Class_6       0       0       4      34       3    4049      56      18       1
   Class_7       0       1      14      30       1      80     716      34       9
   Class_8       0       0       0       0       0      30      76    2465      65
   Class_9       0       0       0       0       0       7       8      58    1417

Overall Statistics

               Accuracy : 0.9166
                 95% CI : (0.9125, 0.9205)
    No Information Rate : 0.2601
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.8995
```

Figure 32: Confusion Matrix (LR- MN)

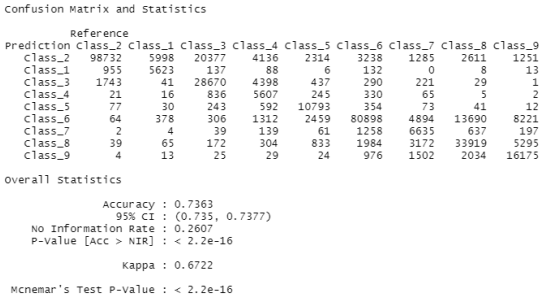


	Sensitivity	Specificity
Class: Class_1	0.9116118	0.9988324
Class: Class_2	0.9867467	0.9850007
Class: Class_3	0.7939805	0.9727228
Class: Class_4	0.4822695	0.9820511
Class: Class_5	0.9566787	0.9981390
Class: Class_6	0.9651967	0.9919265
Class: Class_7	0.8364486	0.9904558
Class: Class_8	0.9561676	0.9893025
Class: Class_9	0.9497319	0.9957237

Figure 33: Sensitivity and Specificity by Class (LR-MN)

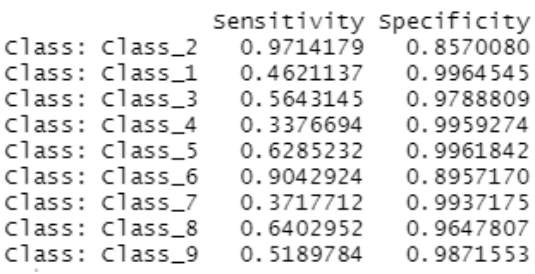
4.2.4 Neural Network

An artificial neural network learning algorithm, or neural network, or just neural net, is a computational learning system that uses a network of functions to understand and translate a data input of one form into a desired output, usually in another form. Interestingly, neural networks performed very poorly with this data-set with an accuracy of only around 73%. It also took a long time to build the model (with cross validation 54 minutes). Many teams in this competition seem to have achieved better accuracy with Neural networks, so we would need to investigate how to tune to network to achieve better results.



```
Confusion Matrix and Statistics

          Reference
Prediction Class_2 Class_1 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_2   98732    5998   20377    4136    2314    3238    1285    2611    1251
   Class_1     955    5623     137      88       6     132       0       8      13
   Class_3    1743      41   28670    4398     437     290     221      29       1
   Class_4      21      16     836    5607     245     330      65       5       2
   Class_5      77      30     243     592   10793     354      73      41      12
   Class_6      64     378     306    1312    2459   80898    4894   13690    8221
   Class_7       2       4      39     139      61    1258    6635     637     197
   Class_8      39      65     172     304     833    1984    3172   33919    5295
   Class_9       4      13      25      29      24     976    1502    2034   16175

Overall Statistics

               Accuracy : 0.7363
                 95% CI : (0.735, 0.7377)
    No Information Rate : 0.2607
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.6722

 Mcnemar's Test P-Value : < 2.2e-16
```

Figure 34: Confusion Matrix (Neural Network)

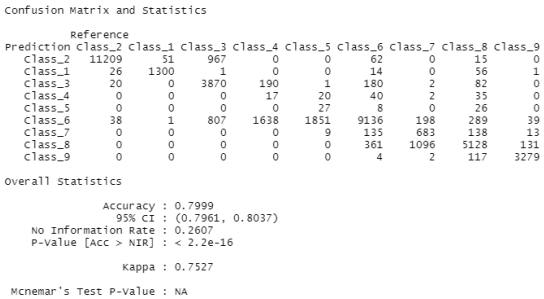


	Sensitivity	Specificity
Class: Class_2	0.9714179	0.8570080
Class: Class_1	0.4621137	0.9964545
Class: Class_3	0.5643145	0.9788809
Class: Class_4	0.3376694	0.9959274
Class: Class_5	0.6285232	0.9961842
Class: Class_6	0.9042924	0.8957170
Class: Class_7	0.3717712	0.9937175
Class: Class_8	0.6402952	0.9647807
Class: Class_9	0.5189784	0.9871553

Figure 35: Sensitivity and Specificity by Class (Neural Network)

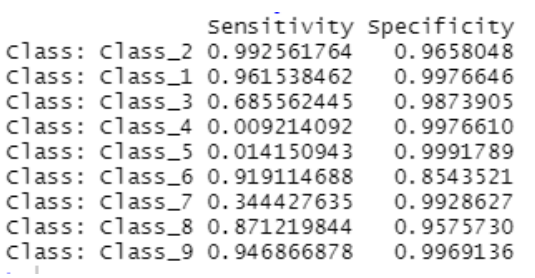
4.2.5 SVM (with PCA)

A Support Vector Machine (SVM) is a very powerful and versatile Machine Learning model, capable of performing linear or nonliner classification, regression, and even outlier detection. For the next model, the data was pre-processed to build a PCA model and then used the preprocessed PCA model to build an SVM model. The accuracy with cross validation was around 80% but model build was a bit faster taking around 16 minutes to build the model (with cross validation).



```
Confusion Matrix and Statistics

          Reference
Prediction Class_2 Class_1 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_2   11209      51     967       0       0      62       0      15       0
   Class_1      26    1300       1       0       0      14       0      56       1
   Class_3      20       0    3870     190       1     180       2      82       0
   Class_4       0       0       0      17      20      40       2      35       0
   Class_5       0       0       0       0      27       8       0      26       0
   Class_6      38       1     807    1638    1851    9136     198     289      39
   Class_7       0       0       0       0       9     135     683     138      13
   Class_8       0       0       0       0       0     361    1096    5128     131
   Class_9       0       0       0       0       0       4       2     117    3279

Overall Statistics

               Accuracy : 0.7999
                 95% CI : (0.7961, 0.8037)
    No Information Rate : 0.2607
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.7527

 Mcnemar's Test P-Value : NA
```

Figure 36: Confusion Matrix (PCA+SVM)



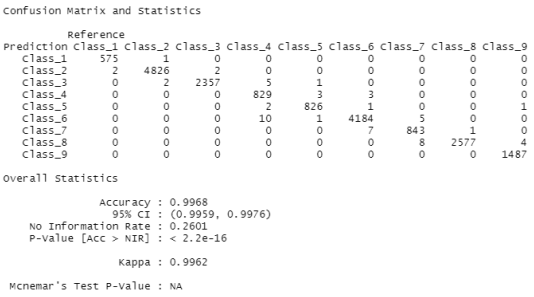
	Sensitivity	Specificity
Class: Class_2	0.992561764	0.9658048
Class: Class_1	0.961538462	0.9976646
Class: Class_3	0.685562445	0.9873905
Class: Class_4	0.009214092	0.9976610
Class: Class_5	0.014150943	0.9991789
Class: Class_6	0.919114688	0.8543521
Class: Class_7	0.344427635	0.9928627
Class: Class_8	0.871219844	0.9575730
Class: Class_9	0.946866878	0.9969136

Figure 37: Sensitivity and Specificity by Class (PCA+SVM)

4.2.6 Random Forest

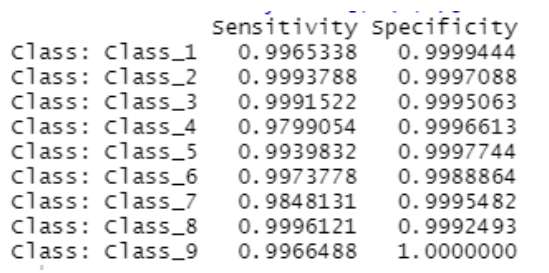
Random forests or random decision forests is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. The random forest algorithm worked very well with this dataset with an accuracy of 99%. The execution time was also fairly acceptable taking around 16 minutes to build the model (with

cross validation). We have to be cautious in using this model as there could be an issue with over-fitting. Further data pre-processing and analysis is required to address the class imbalance and feature separability.



```
Confusion Matrix and Statistics

          Reference
Prediction Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_1     575       1       0       0       0       0       0       0       0
   Class_2       2    4826       2       0       0       0       0       0       0
   Class_3       0       2    2357       5       1       0       0       0       0
   Class_4       0       0       0     829       3       3       0       0       0
   Class_5       0       0       0       2     826       1       0       0       1
   Class_6       0       0       0      10       1    4184       5       0       0
   Class_7       0       0       0       0       0       7     843       1       0
   Class_8       0       0       0       0       0       0       8    2577       4
   Class_9       0       0       0       0       0       0       0       0    1487

Overall Statistics

               Accuracy : 0.9968
                 95% CI : (0.9959, 0.9976)
    No Information Rate : 0.2601
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9962

 Mcnemar's Test P-Value : NA
```

Figure 38: Confusion Matrix (Random Forest)

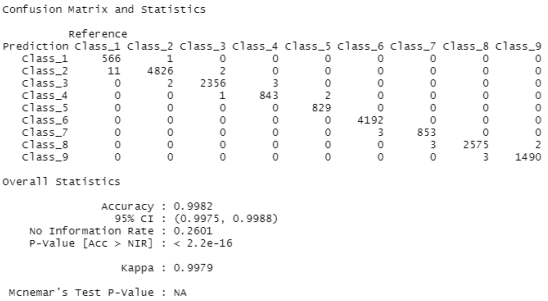


	Sensitivity	Specificity
Class: Class_1	0.9965338	0.9999444
Class: Class_2	0.9993788	0.9997088
Class: Class_3	0.9991522	0.9995063
Class: Class_4	0.9799054	0.9996613
Class: Class_5	0.9939832	0.9997744
Class: Class_6	0.9973778	0.9988864
Class: Class_7	0.9848131	0.9995482
Class: Class_8	0.9996121	0.9992493
Class: Class_9	0.9966488	1.0000000

Figure 39: Sensitivity and Specificity by Class (Random Forest)

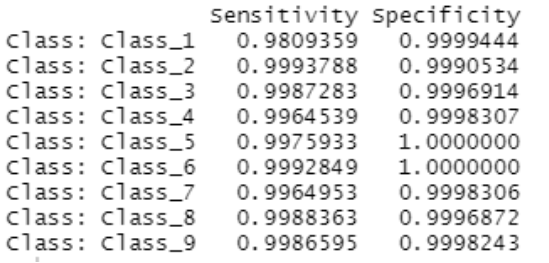
4.2.7 XGBoost

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient, flexible and portable. It implements machine learning algorithms under the Gradient Boosting framework. XGBoost also performed very well with this dataset with an accuracy (cross validation) of 99%. It was also very fast taking only 3 minutes to build the model. As with random forest, we need to be careful in recommending this model before addressing the class imbalance issue.



```
Confusion Matrix and Statistics

          Reference
Prediction Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_1     566       1       0       0       0       0       0       0       0
   Class_2      11    4826       2       0       0       0       0       0       0
   Class_3       0       2    2356       3       0       0       0       0       0
   Class_4       0       0       1     843       2       0       0       0       0
   Class_5       0       0       0       0     829       0       0       0       0
   Class_6       0       0       0       0       0    4192       0       0       0
   Class_7       0       0       0       0       0       3     853       0       0
   Class_8       0       0       0       0       0       0       3    2575       2
   Class_9       0       0       0       0       0       0       0       3    1490

Overall Statistics

               Accuracy : 0.9982
                 95% CI : (0.9975, 0.9988)
    No Information Rate : 0.2601
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9979

 Mcnemar's Test P-Value : NA
```

Figure 40: Confusion Matrix (XGBoost)

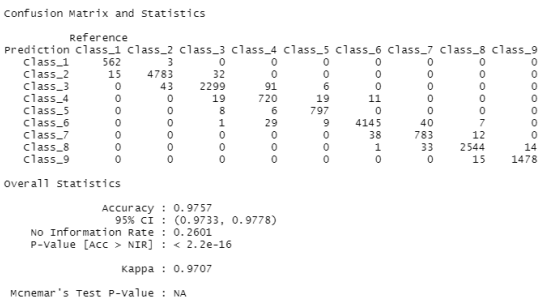


	Sensitivity	Specificity
Class: Class_1	0.9809359	0.9999444
Class: Class_2	0.9993788	0.9990534
Class: Class_3	0.9987283	0.9996914
Class: Class_4	0.9964539	0.9998307
Class: Class_5	0.9975933	1.0000000
Class: Class_6	0.9992849	1.0000000
Class: Class_7	0.9964953	0.9998306
Class: Class_8	0.9988363	0.9996872
Class: Class_9	0.9986595	0.9998243

Figure 41: Sensitivity and Specificity by Class (XGBoost)

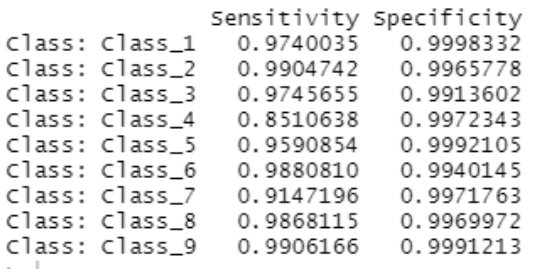
4.2.8 Benchmark - Elastic Net Classifier

Elastic Net regression is a classification algorithm that overcomes the limitations of the lasso(least absolute shrinkage and selection operator) method which uses a penalty function in its L1 regularization. Elastic Net regression is a hybrid approach that blends both penalizations of the L2 and L1 regularization of lasso and ridge methods. We decided to use this model as a benchmark for comparing model performance. The model performed well with an accuracy of 97%, but the computation time was very slow, taking over 3 hours to create the model. Still, it was a good benchmark to compare it with other models.



```
Confusion Matrix and Statistics

          Reference
Prediction Class_1 Class_2 Class_3 Class_4 Class_5 Class_6 Class_7 Class_8 Class_9
   Class_1     562       3       0       0       0       0       0       0       0
   Class_2      15    4783      32       0       0       0       0       0       0
   Class_3       0      43    2299      91       6       0       0       0       0
   Class_4       0       0      19     720      19      11       0       0       0
   Class_5       0       0       8       6     797       0       0       0       0
   Class_6       0       0       1      29       9    4145      40       7       0
   Class_7       0       0       0       0       0      38     783      12       0
   Class_8       0       0       0       0       0       1      33    2544      14
   Class_9       0       0       0       0       0       0       0      15    1478

Overall Statistics

               Accuracy : 0.9757
                 95% CI : (0.9733, 0.9778)
    No Information Rate : 0.2601
    P-Value [Acc > NIR] : < 2.2e-16

                  Kappa : 0.9707

 Mcnemar's Test P-Value : NA
```

Figure 42: Confusion Matrix (Elastic Net)



	Sensitivity	Specificity
Class: Class_1	0.9740035	0.9998332
Class: Class_2	0.9904742	0.9965778
Class: Class_3	0.9745655	0.9913602
Class: Class_4	0.8510638	0.9972343
Class: Class_5	0.9590854	0.9992105
Class: Class_6	0.9880810	0.9940145
Class: Class_7	0.9147196	0.9971763
Class: Class_8	0.9868115	0.9969972
Class: Class_9	0.9906166	0.9991213

Figure 43: Sensitivity and Specificity by Class (Elastic Net)

4.3 Model Performance

The top 3 models that performed the best (with cross validation) were XGBoost, Random Forest and Elastic Net. XGBoost overall was the most desirable model in terms of accuracy and execution time. The table below summarizes the model performance for each model.

MODEL	EXECUTION TIME	ACCURACY	KAPPA
XGBoost	3.42 mins	99.82%	99.79%
RandomForest	16.36 mins	99.68%	99.62%
Elastic Net	2.38 hours	97.57%	97.07%
LR - Multinomial	26.81 mins	91.66%	89.95%
KNN	3.28 hours	84.21%	80.85%
PCA + SVM	16.81 min	79.99%	75.27%
Naive Bayes	26.60 secs	74.81%	69.68%
NeuralNetwork	53.66 mins	73.63%	67.22%

Figure 44: Model Performance Summary

4.4 Model Summary

- Scaling the data and the addition of the two synthetic variables helped in improving model accuracy.
- Without cross validation
  - All models, except for Naive Bayes, performed well without cross validation.
  - XGBoost and Random Forest had the best performance.
- With cross validation
  - Here again, XGBoost and Random Forest had the best performance.
  - Neural network and Naive Bayes did not perform well with cross validation.
- Computationally, XGBoost and Random forest were very fast. Even though Naïve Bayes had the fastest execution time, it had very poor accuracy.
- The benchmark Elastic Net classifier performed better than KNN, SVM, Logistic Regression and Naive Bayes. XGBoost and Random forest were better than Elastic net in terms of accuracy and speed.

5 Conclusion

This was an excellent project to apply all the learnings from the ISYE-7406 course. From feature analysis to model selection, I were able to apply techniques learned in the course to improve the model.

5.1 Question we hoped to answer

- How does each model perform?
  All models except for Naive Bayes performed fairly well even with cross-validation.
- Is there a category (or categories) that are similar enough that all models struggle to classify accurately? If so, why and how to address it?
  Even though it appeared that class separability could be a problem, adding the two variables seem to address that issue. Looking at the confusion matrix for each model, it appears that not a lot of rows were incorrectly classified by each model. However a caveat here is that class imbalance was not addressed and that might affect the classification accuracy.
- It is better to dimensionally reduce data to its n principal components or is it better to use the n most important features (Permutation Feature Importance - PFI)?
  PCA performed moderately well, but all features were used for XGBoost and Random Forest and both these models performed very well.Additional dimension reducing methods (t-SNE -t-distributed Stochastic Neighbor Embedding) can be investigated to improve model performance.

5.2 Key Takeaways

- Since the data was obfuscated, no real-world interpretation of the data was possible.
- Most of the models built performed well, and cross validation was valuable in identifying the best model.

- The classes were imbalanced, so overfitting is an issue. The models created in this project did not address the class imbalance.
- Next steps, would be to address class imbalance by applying techniques like SMOTE, up-sampling, etc.
- The performance evaluation method for the actual Kaggle competition was different from the evaluation done for this project. It would be good to evaluate the models developed in this project against the Kaggle evaluation method.

References

P. Teisseyre, "Ccnet: Joint multi-label classification and feature selection using classifier chains and elastic net regularization," *Neurocomputing*, vol. 235, pp. 98–111, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231217300073

A. Altmann, L. Toloşi, O. Sander, and T. Lengauer, "Permutation importance: a corrected feature importance measure," *Bioinformatics*, vol. 26, no. 10, pp. 1340–1347, 04 2010. [Online]. Available: https://doi.org/10.1093/bioinformatics/btq134

J. Brownlee, "How to choose a feature selection method for machine learning." [Online]. Available: https://machinelearningmastery.com/feature-selection-with-real-and-categorical-data/

C. Molnar, G. König, B. Bischl, and G. Casalicchio, "Model-agnostic feature importance and effects with dependent features–a conditional subgroup approach," *arXiv preprint arXiv:2006.04628*, 2020.

G. König, C. Molnar, B. Bischl, and M. Grosse-Wentrup, "Relative feature importance," in *2020 25th International Conference on Pattern Recognition (ICPR)*, 2021, pp. 9318–9325.

H. Zou and T. Hastie, "Regularization and variable selection via the elastic net," *Journal of the royal statistical society: series B (statistical methodology)*, vol. 67, no. 2, pp. 301–320, 2005.

7 Appendix

7.1 R code

See attached R files