Table of Contents

1 Project Definition

1.1 Project Overview

Advent of mobile devices and affordable internet has redefined the way millennials shop. E-commerce has transformed the shopping experience from physical stores and standing in queues to one of 'websites/apps' and online payment. In 2017 global ecommerce sales totalled US$1.9 trillions and is projected to increase to US$4.5 trillion by 2021 [1]. With increasing competition, multi-national operation and rapid addition of new products – product categorization has become the backbone of every online retailer.

Accurate product categorization is key to ensure that the customers are able to find what they are look for in an easy and intuitive manner. Product categorization often forms the first impression among the users and goes a long way for increasing customer retention. In addition, accurate and consistent product categorization is essential for the ecommerce retailers to be able to generate meaningful business insights from their data. This is especially so for retailers with diverse global operations and millions of products where many identical products can get classified differently.

Traditionally, product categorization has been done manually by humans. This poses two main challenges – (i) it is a time intensive task especially in the light of millions of products handled by retailers and (ii) inconsistencies as different people may categorize the same product differently [1].

To overcome these challenges, several research studies have been devoted towards developing an automated mechanism for accurate and consistent product categorization. Various supervised learning algorithms such as Naïve Bayes, KNN, SVM, tree classifiers and neural networks have been used to classify products with reasonable accuracy into different categories [1] [2] [3] [4] [5]. Features used in these studies for classification include – product images, product title, description, SKUs and technical details. The choice of classification algorithm used in the studies depends on the dataset - with deep learning being used for image data sets and NLP algorithms for textual features. The methodologies and results in the surveyed literature demonstrates the feasibility of using supervised learning for product classification.

This project extends the automated product categorization techniques to the online retailer – Otto Group. Otto group is one of the largest global ecommerce companies with operations in more that 20 countries. Their catalogue spans millions of products. The company relies on their ability to accurately classify products into categories for doing product and business analysis. However, given their diverse global operations and large product catalogue, several similar products get classified into different categories.

1.2 Problem Statement

This project aims to train a machine learning algorithm that is able to classify a product into different categories based on the product characteristics. The project is based on Kaggle competition - Otto Group Product Classification Challenge [6]. The data set provided by the Otto group consists of product features and their categories for around 60,000 products. This project aims to use the labelled data and supervised learning techniques to develop an algorithm for predicting the product categories given the product features.

1.3 Solution Overview

With the given problem statement and dataset, the project will use supervised probabilistic classification algorithms to classify the products into different categories. The given dataset will be split into training and testing sets. Supervised learning will be used to train the algorithms to predict the probability distribution of the product belonging to the different categories given its input features. The proposed solution will use an ensemble of classification algorithms. The developed model will be evaluated on the test set using log-loss function which is discussed in the Evaluation Metrics section.

The project will be approached in 4 stages –

1. Data Exploration and Pre-Processing

The first step of the project would be to analyse and pre-process the dataset. Dataset will be cleaned to remove unwanted values (such as product ids) and encode the target labels. Feature scaling and normalizing techniques will be used to scale skewed features.

2. Dimensionality Reduction

The given dataset consists of 93 features. This is a large feature set and this section will explore dimensionality reduction and feature selection for reducing the input feature set. Principal component analysis will be used to explore the viability of dimensionality reduction for the dataset.

3. Supervised Learning

Next step would to use the reduced feature set and target labels to train supervised classification models. Different classification algorithms that will be explored are - random forests, K-Nearest Neighbours and boosting. These algorithms have been chosen as potential candidates because of their ability to work with multiclass classification problems and output a probability distribution over target classes. In addition to looking at the individual algorithms' performance, ensemble learning approach will also be explored.

4. Model Testing

The trained models will be evaluated on the test data set using the log loss function defined in the Evaluation Metrics section. Models' performance on both test and training dataset will be evaluated for signs of overfitting. The goal of the training process would be to minimize the log loss function on the test data set.

1.4 Evaluation Metric

The solution for this project will be evaluated using the multi-class logarithmic loss (log loss) function. Multi class log loss is the pre-defined evaluation metric for the Kaggle competition in Otto product classification challenge. The metric is suitable for the problem as the classification algorithms will be trained to output a set of predicted probabilities (one for each of the 9 product categories). Log loss is a soft evaluation metric that takes into account the uncertainty in predicted values. It measures the performance of a classification model where

the prediction output is a probability value between 0 and 1. The log loss function is defined as

$$logloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{j=1}^{M} y_{ij}\log(p_{ij})$$

where –

N is the number of data points in the test set

M is the number of product categories

y_{ij} is 1 if observation i belongs to class j and 0 otherwise

p_{ij} is the predicted probability that observation i belongs to class j

log is natural logarithm

The machine learning models will be trained to minimize the log loss function. The perfect model will have a log loss of 0. The value of the log loss function increases as the predicted probability diverges from the actual label.

1.5 Framework and Libraries

Following framework and libraries will be used for the project

- Python 3.x
- Jupyter
- Pandas, Numpy, Scikit-Learn

2. Analysis

2.1 Data Exploration

The dataset for this project comes from the Kaggle competition organized by Otto Group [6]. The dataset contains information about 60,000 products and their target categories. Each row in the dataset corresponds to a single product. Each product has a unique anonymous id, features and a target category.

- Input Features
  Each data point is described by an id and 93 numerical features. As the 'id' is a unique identifier, it will be excluded from the feature set. Each of the 93 features represent counts of different events. The exact description of the features has been obfuscated implying that we would have to rely more on algorithmic techniques rather than intuition to build the pre-processing and classification models.

- Target Labels

  There are 9 target categories (Class_1, Class_2 ……. Class_9) that the products need to be classified into. Each target category represents one of the major product categories (such as fashion, electronics etc.). Any Further description of the categories is obfuscated by the competition hosts.

The competition hosts have provided two data files – train.csv and test.csv. Train.csv has labelled data. The data in test.csv is unlabelled and is provided for assessing the performance of competitors models on the public and private leaderboard on Kaggle. As the data in test.csv is unlabelled, it will not be used for the purpose of this project.

The dataset is relatively clean with no missing values. All the features are numerical with positive integer values greater than equal to 0.

Table 1 below shows the features of three data points from Class 1, Class 2 and Class 8.

Table 1: Three data points from the raw Otto product dataset

	Sample Point 1	Data Point 2	Data Point 3
Label	**Class 1**	**Class 2**	**Class 8**
id	1	2361	49384
feat_1	1	0	0
feat_2	0	0	2
feat_3	0	0	0
feat_4	0	0	0
feat_5	0	0	0
feat_6	0	0	0
feat_7	0	0	1
feat_8	0	0	0
feat_9	0	0	0
feat_10	0	0	0
feat_11	1	0	0
feat_12	0	0	1
feat_13	0	0	11
feat_14	0	17	2
feat_15	0	0	0
feat_16	0	0	0
feat_17	2	0	0
feat_18	0	0	0
feat_19	0	0	0
feat_20	0	0	7
feat_21	0	0	0
feat_22	1	0	0
feat_23	0	0	0
feat_24	4	1	15
feat_25	1	11	2
feat_26	1	0	0
feat_27	0	0	0
feat_28	0	0	0
feat_29	2	0	0
feat_30	0	0	1
feat_31	0	0	0
feat_32	0	0	4
feat_33	0	3	0
feat_34	0	0	2
feat_35	1	0	1
feat_36	0	1	2
feat_37	0	0	0
feat_38	0	0	10
feat_39	0	0	0
feat_40	1	6	0
feat_41	0	0	1
feat_42	5	2	1
feat_43	0	0	0
feat_44	0	0	1
feat_45	0	0	2
feat_46	0	0	0
feat_47	0	0	0
feat_48	2	5	4
feat_49	0	0	0

feat_50	0	0	2
feat_51	0	0	0
feat_52	0	0	0
feat_53	0	0	0
feat_54	1	0	3
feat_55	0	1	0
feat_56	0	0	0
feat_57	2	1	0
feat_58	0	0	1
feat_59	0	0	0
feat_60	11	0	1
feat_61	0	0	0
feat_62	1	0	3
feat_63	1	0	1
feat_64	0	0	4
feat_65	1	0	1
feat_66	0	0	0
feat_67	7	3	11
feat_68	0	1	0
feat_69	0	0	11
feat_70	0	0	5
feat_71	1	1	0
feat_72	0	0	0
feat_73	0	0	0
feat_74	0	0	0
feat_75	0	0	0
feat_76	0	0	25
feat_77	0	0	0
feat_78	0	0	0
feat_79	2	0	0
feat_80	1	0	0
feat_81	0	0	0
feat_82	0	0	0
feat_83	0	0	2
feat_84	0	0	0
feat_85	1	0	1
feat_86	0	0	2
feat_87	0	0	1
feat_88	0	8	1
feat_89	0	0	0
feat_90	0	0	3
feat_91	0	0	0
feat_92	0	0	0
feat_93	0	0	0

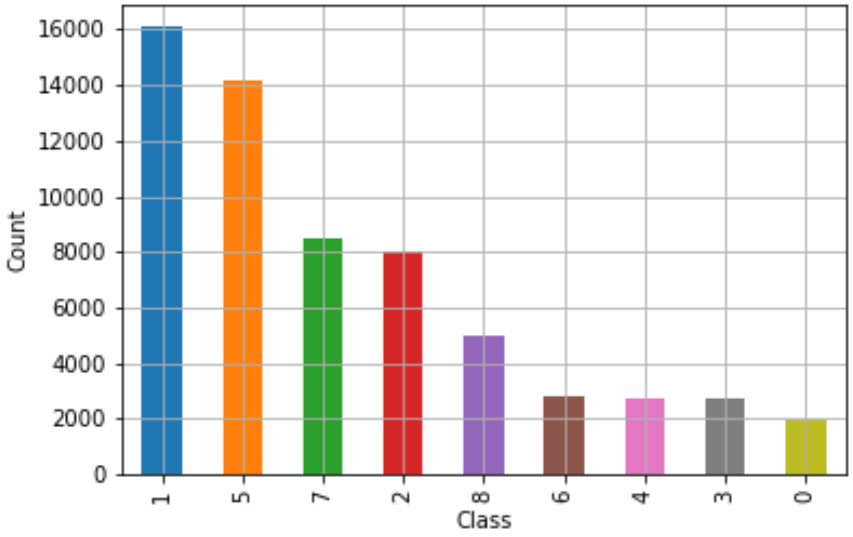

Fig 1: Class wise distribution of the data samples

Fig 1. shows the distribution of the data points in different classes. It is observed that the dataset is imbalanced with the Class 1 having more than 7 times the number of samples in Class 0. The imbalanced dataset can potentially bias the classifier algorithms as the algorithms see more samples from a particular class. One of the ways to work around this imbalance will be to ensure that the training and testing sets retain the class distribution of the original dataset. Additionally, confusion matrix would be used to analyse if the algorithm is biased towards the majority class (falsely predicting minority class to be in majority class).

Table 2: Descriptive statistics for the first 10 features

Feature	Count	Mean	Std	Min	25%	50%	75%	max
feat_1	61878	0.38668	1.52533	0	0	0	0	61
feat_2	61878	0.263066	1.252073	0	0	0	0	51
feat_3	61878	0.901467	2.934818	0	0	0	0	64
feat_4	61878	0.779081	2.788005	0	0	0	0	70
feat_5	61878	0.071043	0.438902	0	0	0	0	19
feat_6	61878	0.025696	0.215333	0	0	0	0	10
feat_7	61878	0.193704	1.030102	0	0	0	0	38
feat_8	61878	0.662433	2.25577	0	0	0	1	76
feat_9	61878	1.011296	3.474822	0	0	0	0	43
feat_10	61878	0.263906	1.08334	0	0	0	0	30

Table 2 shows descriptive statistics (count, mean, standard deviation and percentiles) for the first 10 features in the dataset. It can be observed that the data distribution on each feature is very skewed – with very low min and even 75percentile value and a high max value. The data distribution along each feature has a long right sided tail with most of the values lying close

to 0. This is a trend that is observed across all features in the dataset. This skewed distribution can be overcome through feature scaling during the data pre-processing step.

2.2 Algorithms and Techniques

The given dataset has a high-dimensionality with 93 features. After data pre-processing, the first step will be to reduce the dimensionality. A low dimensional data would be faster to train and also help to minimize noise. Principal Component Analysis (PCA) will be explored for dimensionality reduction. PCA helps to reduce the dimensionality of the data by linearly mapping correlated variables into uncorrelated orthogonal components while maintaining the variance in the data. The cumulative variance explained by different components will be studied and the possibility of using a smaller number of principal components as features will be explored.

The evaluation metric for this problem is log-loss. Therefore, we will be looking at supervised classification algorithms that are able to predict the class wise probability distribution for each observation. Given the large number of features and a relatively medium sized dataset, an important consideration is the computational resources required for training the classification algorithms. In this light, four supervised classification algorithms will be explored – Random Forest, K-Nearest Neighbors, Gradient Boosting and AdaBoost. These four algorithms will be trained using grid search cross-validation. Grid search will help in tuning the hyper parameters for each algorithm. Cross-validation will help to minimise overfitting in the data.

In the first run, each of the four algorithms are trained individually and their log-loss score will be benchmarked. In the next step, ensemble combination of two or more of these algorithms will be explored. Soft voting ensemble will be used combine the results from multiple algorithms. This will involve iteratively determining the optimal weights for different algorithms in the ensemble.

2.3 Benchmark Models

Three benchmarks will be considered for this project –

1. Naïve model – The Naïve model randomly assigns the products into categories. This is the lowest benchmark for the model. The log-loss metric for Naïve model is **30.52**. The developed model should be able to outperform the naïve model.
2. Simple model – Each of the four individual model's performance will be considered as a benchmark for the final ensemble model
3. Top score on the Kaggle Competition – The top score on public leaderboard for the Otto Product Classification challenge is **0.38055** [7]. This will be the best-case benchmark.

3. Methodology

3.1 Data Pre-Processing

The raw dataset for Otto product classification is relatively clean with no missing values. To make the data ready for analysis, following pre-processing steps were applied:

(i) Dropping the 'id' column from the dataset: 'id' is a unique identifier for each sample in the row and therefore it is excluded from the dataset

(ii) Separating the dataset into features and labels: All the 93 features are included in the dataset for analysis. The product category is the target label for the dataset

(iii) Label encoding: In the raw dataset, the target labels are defined as [Class_1, Class_2 … Class_9]. Using sklearn label encoder, the default labels are encoding into integers ranging from 0 to 8 such that '0' indicates 'Class_1' and '8' indicates 'Class_9'. Label encoding is required to make the target labels suitable for input to classifiers

(iv) Feature normalization: In the data exploration, it is observed that the data distribution on each feature is skewed (Table 2) . The data needs to be normalized to prevent the classification algorithms from being biased towards the samples with large feature values. Log(x+1) normalization is used due to the frequent occurrence of 0 as feature value.

Table 3 shows the first three datapoints from the transformed dataset after all the pre-processing steps are applied.

Table 3: First three data points after applying pre-processing steps

	Sample Point 1	Data Point 2	Data Point 3
Label	0	1	2
feat_1	0.693147	0	0
feat_2	0	0	0
feat_3	0	0	0
feat_4	0	0	0
feat_5	0	0	0
feat_6	0	0	0
feat_7	0	0	0
feat_8	0	0.693147	0.693147
feat_9	0	0	0
feat_10	0	0	0
feat_11	0.693147	0	0
feat_12	0	0	0
feat_13	0	0	0
feat_14	0	0	0
feat_15	0	0	0
feat_16	0	0	0
feat_17	1.098612	0	0.693147
feat_18	0	1.098612	0
feat_19	0	0	0
feat_20	0	0	0
feat_21	0	0	0
feat_22	0.693147	0	0
feat_23	0	0	0
feat_24	1.609438	0	0
feat_25	0.693147	0	0
feat_26	0.693147	0	0
feat_27	0	0	0
feat_28	0	0	0
feat_29	1.098612	0	0
feat_30	0	0	0
feat_31	0	0	0
feat_32	0	0	0
feat_33	0	0	0.693147
feat_34	0	0	0
feat_35	0.693147	0	0
feat_36	0	0	0
feat_37	0	0.693147	0
feat_38	0	0	0
feat_39	0	0	0
feat_40	0.693147	0	0
feat_41	0	0	0
feat_42	1.791759	0	0
feat_43	0	0	0
feat_44	0	0	0
feat_45	0	0	0
feat_46	0	0	0
feat_47	0	0	0
feat_48	1.098612	0	0.693147
feat_49	0	0	0
feat_50	0	0	0

feat_51	0	0	0
feat_52	0	0	0
feat_53	0	0	0
feat_54	0.693147	0	0
feat_55	0	0	0.693147
feat_56	0	0	0
feat_57	1.098612	0	0
feat_58	0	0.693147	0
feat_59	0	0	0
feat_60	2.484907	0	0
feat_61	0	0	0
feat_62	0.693147	0	0
feat_63	0.693147	0	0
feat_64	0	0.693147	0
feat_65	0.693147	0	0
feat_66	0	0	0
feat_67	2.079442	0.693147	1.94591
feat_68	0	0	0
feat_69	0	0	0
feat_70	0	0	1.098612
feat_71	0.693147	0	0
feat_72	0	0	0
feat_73	0	1.098612	0
feat_74	0	0.693147	0
feat_75	0	0	0
feat_76	0	0.693147	0
feat_77	0	0	0
feat_78	0	0.693147	0
feat_79	1.098612	0	0
feat_80	0.693147	0	0
feat_81	0	0	0.693147
feat_82	0	0	0
feat_83	0	0	0
feat_84	0	0	0
feat_85	0.693147	0	0
feat_86	0	0	0
feat_87	0	0	0
feat_88	0	0	0
feat_89	0	0	0
feat_90	0	0	0
feat_91	0	0	0
feat_92	0	0	0
feat_93	0	0	0

Table 4 shows the descriptive stats of the transformed dataset

	count	mean	std	min	25%	50%	75%	max
feat_1	61878	0.173119	0.43391	0	0	0	0	4.127134
feat_2	61878	0.114539	0.367244	0	0	0	0	3.951244
feat_3	61878	0.278861	0.647542	0	0	0	0	4.174387
feat_4	61878	0.269545	0.596573	0	0	0	0	4.26268
feat_5	61878	0.040018	0.192083	0	0	0	0	2.995732
feat_6	61878	0.015508	0.116978	0	0	0	0	2.397895
feat_7	61878	0.087794	0.317947	0	0	0	0	3.663562
feat_8	61878	0.278311	0.535108	0	0	0	0.693147	4.343805
feat_9	61878	0.273583	0.672973	0	0	0	0	3.78419
feat_10	61878	0.123938	0.36726	0	0	0	0	3.433987

From Table 4, we see that after pre-processing, the distribution of data samples on each features is less skewed – lower standard deviation and lower difference between lowest and highest value for each feature.

3.2 Dimensionality Reduction

After doing data pre-processing, the next question is whether we need all 93 features that is can we use a smaller number of features while retaining the maximal variance in the data. Principal Component Analysis (PCA) is used draw observations about the underlying structure of the product data. Applying PCA on a dataset, calculates the dimensions that explain the maximum variance in the data. The PCA dimensions are linear combinations of original features. With PCA, we can discover the dimensions that best explain the variance in data thereby helping in dimensionality reduction.

Sklearn.PCA library is used to apply PCA on the pre-processed dataset. The number of components is set to 93 which is the number of features in the dataset. Fig 2 shows the variance explained by each principal component:

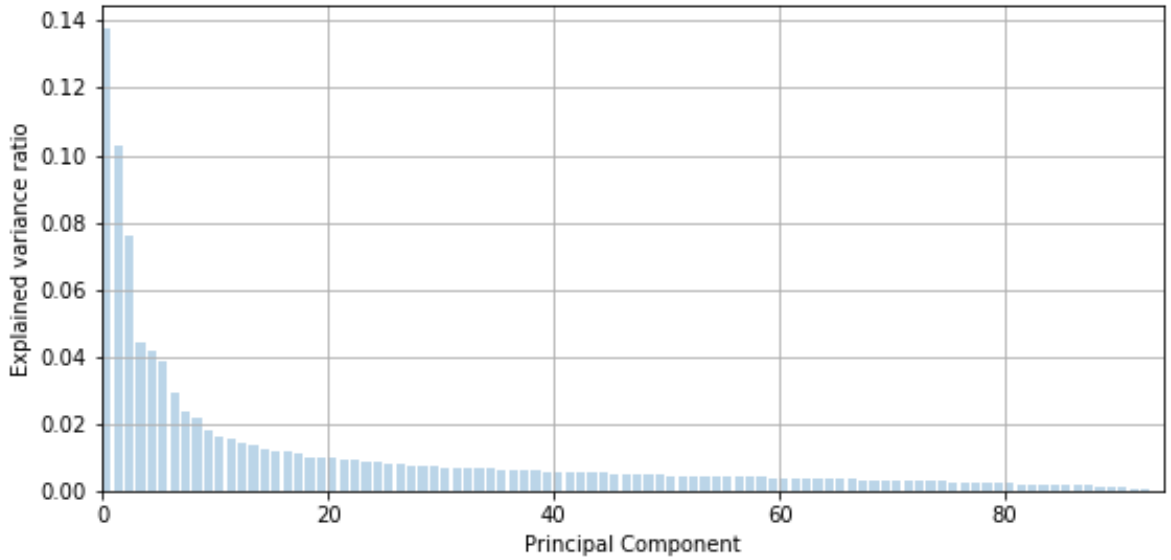

Fig 2: Proportion of variance in data explained by each principal component

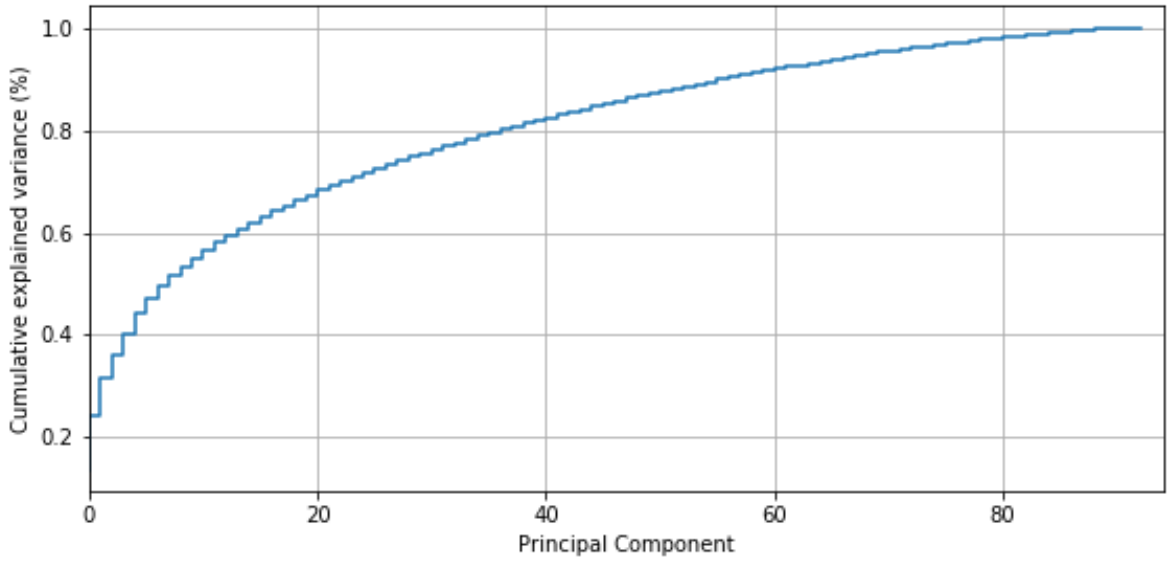

Fig 3: Cumulative variance explained by each successive principal component

Figure 2 and 3 show the variance explained by each principal component (PC) and the cumulative variance explained by each successive PC. The first PC explains 14% variance (highest among all PCs) in the data. It can be seen that no single PC is able to explain significant variance in the data. At least 60 PCs are required to explain 90% of variance in the product dataset. Due to the lack of a smaller number of PCs to explain maximal variance in the dataset, PCA is not suitable for dimensionality reduction with this dataset.

3.3 Implementation and Refinement

3.3.1 Test/Train data split

Before training the classifiers, the first step is to split the data into training and testing sets. As we observed earlier, the product data is an imbalanced dataset in terms of the number of samples available for each category. To prevent the classifiers from becoming biased towards the majority class, **stratified shuffle split** is used to preserve the original class frequency distribution in the training and testing data sets.

3.3.2 Cross-Validation and Parameter Tuning

Cross-validation is a validation technique that helps to minimize overfitting while avoiding the need to carve out a separate validation set from the training dataset. In k-fold cross validation, the training data is split into k smaller subsets. The models are trained on k-1 subsets and the results are validated on the remaining 1 subset. This process is repeated k times and the results are averaged across the k-iterations. Cross validation can be computationally expensive but it is found to be quite useful in getting a better estimate of the accuracy of the model while being able to train on maximal number of samples.

For this project, **GridSearch Cross Validation** (GridSearch CV) is used. GridSearch CV takes hyperparameter space as an input parameter. During the training process, GridSearch CV searches the hyperparameter for the best cross validation score thereby helping out in parameter tuning. The hyperparameters tuned for each classifier are discussed in the following section.

3.3.3 Model Training and Iteration

Naïve Predictor

As the worst case benchmark, a Naïve predictor is developed to randomly assign a category to each product. The log-loss score of the Naïve predictor is 30.52.

Random Forests

Random forests is a supervised ensemble learning algorithm that builds multiple decision trees on the dataset. Each decision tree is trained on a subset of the dataset (samples are drawn with replacement). The final result is averaged over the prediction of the individual decision trees. Random Forests algorithm chosen for the product dataset due to the following reasons –

- By averaging the results over multiple trees, Random Forests algorithm is able to prevent overfitting.
- It is computationally inexpensive to run over large datasets
- It is able to predict class wise probability distribution which is required to compute the log loss metric

Two hyper parameters are tuned using GridSearch CV:

- **N_estimators** : which specifies the number of trees in the forests. The general rule of thumb being more trees leads to better accuracy
- **Max_depth** : which specifies the maximum depth of a tree

Min_samples required for a split is set to 20 and min_samples in each leaf is set to 10.

Fig 4 shows the random forest model performance with different combinations of (num of trees, max depth).

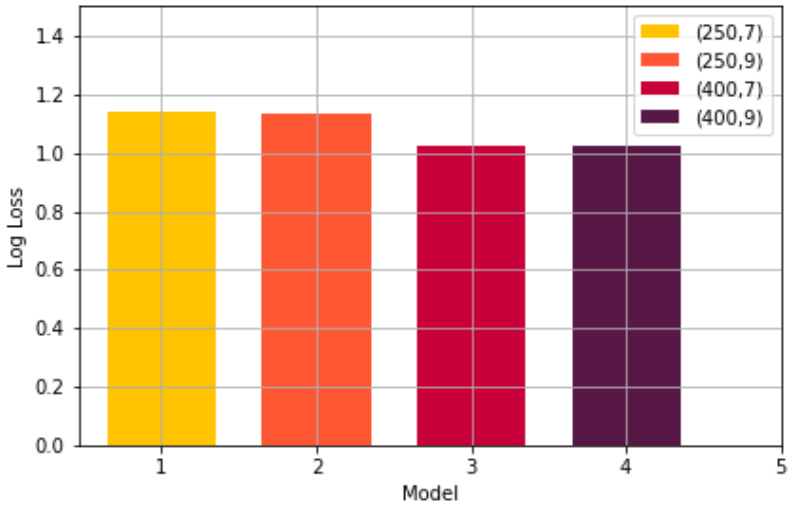

Fig 4: Log loss score for four random forest models with different combinations of (num of trees and max_depth). The number of trees and max_depth for each model is indicated in the legend.

From Figure 4, we can observe that having more number of trees helps to enhance the model performance. Models 1 and 2 with 250 trees have a log loss score of approx. 1.15 while Models 3 and 4 with 400 trees have a log score of approx. 1.10. Variations in max_depth does not have a significant impact on model performance.

The best performing Random Forest model is detailed below:

Log-Loss Score	1.0087
Number of trees	400
Max Depth	9
Min sample per leaf	10
Min samples to split	20

K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a supervised instance based learning model. KNN stores all the instances of training data. Predictions are then made on the basis of a similarity measure.

To make a prediction, KNN takes a majority vote from the K nearest neighbours of the test sample. The similarity index (distance measure) is usually chosen based on domain knowledge. KNN algorithm is chosen for this project because of the following reasons:

- Ability to work with multiclass classification problems
- Computationally inexpensive to train the model, ability to work well with large datasets
- Generally works well in classification tasks

To train the KNN model, the weights parameter is set to 'distance' such that data points 'closer' (more similarity) to the test sample have a higher weight in the prediction.

Parameter 'K' – number of neighbours to use - is tuned using GridSearch. Fig 5. shows the model performance as a function of the number of neighbours parameter:

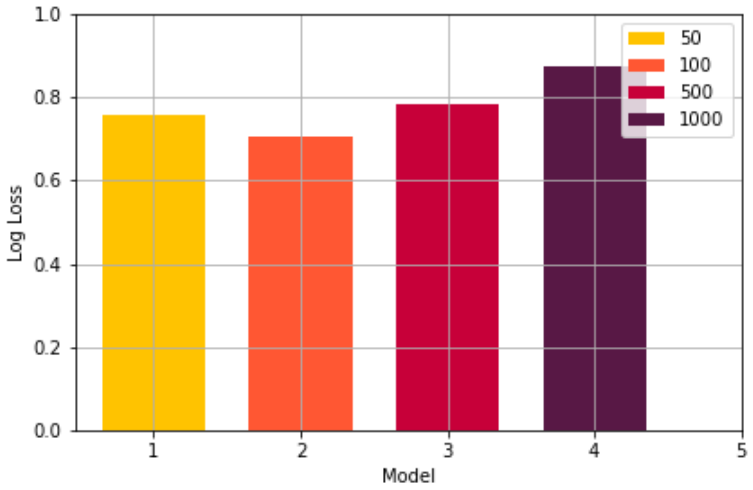

Fig 5: Log loss score of four KNN models with different K parameters. The K parameter for each model is listed in the legend.

The model with lowest log-loss score has K value of 100. The details of the best performing KNN model is listed below:

Log-Loss Score	0.6546
K value	100
Weights	Distance

Boosting

Two Boosting algorithms are trained : AdaBoost and Gradient Boosting.

Adaptive Boosting (Adaboost)

Adaboost is supervised ensemble learning algorithm that uses multiple weak learners to build a strong classifier. Adaboost first fits a classifier on the original dataset. It then trains multiple

copies of the classifier on same dataset wherein each subsequent classifier given more weight to the samples incorrectly classified by the previous classifiers. Adaboost is chosen for this problem for the following reasons –

- Its ability to work with classifiers (that computationally faster to train) and improve their performance through ensembling

For the product dataset, '**Decision Tree**' is used as the base classifier for Adaboost. GridSearch is used to optimize the n_estimators and learning rate parameters.

Fig 6. shows the performance of the Adaboost classifier with different hyperparameters:

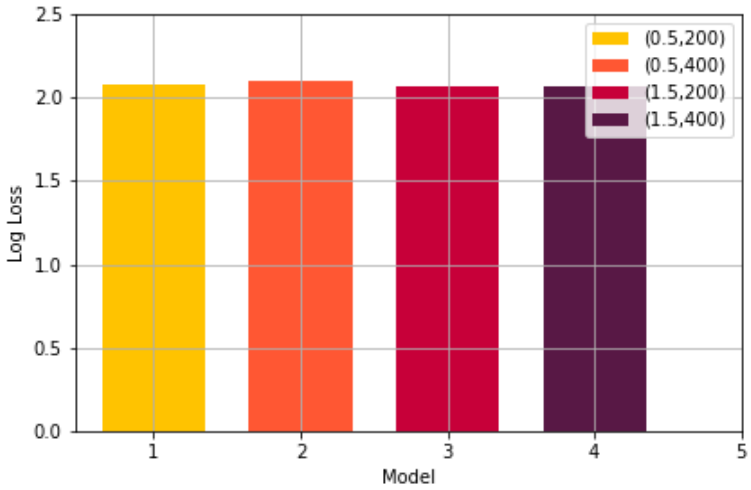

Fig 6: Log loss score of four adaboost models with different learning rates and number of tree parameters. The learning rate and number of trees for each model are listed in the legend.

From Fig 6., it can be observed that there is no significant variation in model performance with the variation in number of trees and learning rate parameter. All the models have a log-loss score of approx. 2.10 .

The best performing Adaboost model is summarised below:

Log-Loss Score	2.0637
Number of trees	400
Learning Rate	1.5
Base estimator	Decision Tree

Gradient Boosting

Similar to Adaboost, Gradient Boosting uses multiple weak learners to develop a strong classifier. The difference between Adaboost and Gradient Boosting is that while Adaboost alters the sample distribution (by giving more weigh to the incorrectly classified samples), Gradient Boosting trains successive classifiers on the residual errors of the previous classifiers. Gradient Boosting is chosen for the following advantages:

- Low generalization error
- Ability to detect non-linear feature interactions

The learning rate parameter of the Gradient Boosting algorithm is tuned using Grid Search.

N_trees is set to 300, min_samples_leaf is set to 10 and max_depth is set to 9.

Fig 7. shows the performance of Gradient Boosting algorithm as a function of learning rate:

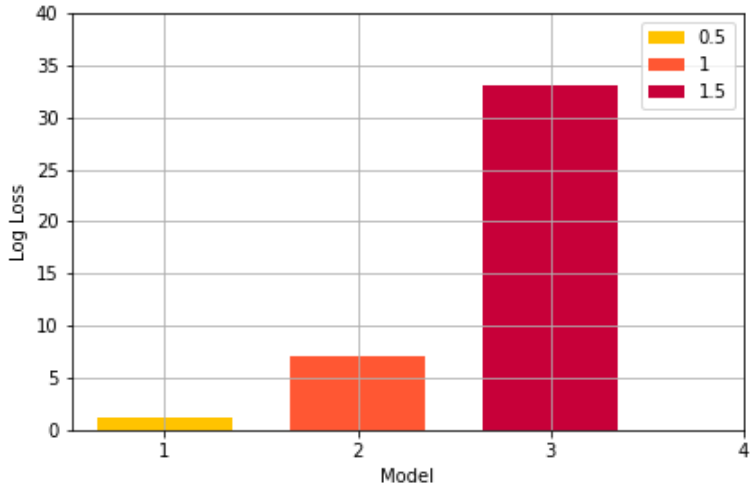

Fig 7: Log loss score of three gradient boosting models with different learning rates. The learning rate for each model is listed in the legend.

From Fig. 7, it can be observed that model 1 has the lowest log loss score. Parameters of Model 1 are described below:

Log-Loss Score	1.2239
Number of trees	300
Learning Rate	0.5
Max Depth	9
Minimum samples in a leaf	10

Ensemble with Voting Classifier

After training the four models above, a voting classifier is used to combine their predictions. A Voting Classifier combines multiple machine learning classifiers and averages their predicted probabilities to predict the target class. A Voting classifier can help to balance out the individual weakness of each classifier to create a better performing model.

Specific weights can be assigned to each classifier defining their contribution to the final prediction. Different combination of classifiers and weights were experimented with to enhance the overall prediction accuracy. The final results are presented in the next section.

4. Results

To enhance the performance of the individual models, we used voting ensemble classifier. Six ensemble classifiers were constructed using different weight combinations of the best performing KNN, Random Forest, AdaBoost and Gradient Boosting models. Table 5 shows the different combinations of weights used. Model 3,4,5 assign weights according to the log loss scores of the individual models such that KNN which is the performing model is given the highest score.

Table 5: Weights for individual algorithms for the 6 ensemble models

	KNN	Random Forest	AdaBoost	Gradient Boosting
Model 1	1	1	1	1
Model 2	1	1	0	1
Model 3	2	1	0	1
Model 4	2	1	0.5	1
Model 5	4	2	0	1
Model 6	1	1	0	0

Fig 8 shows the log loss scores of the six ensemble models.

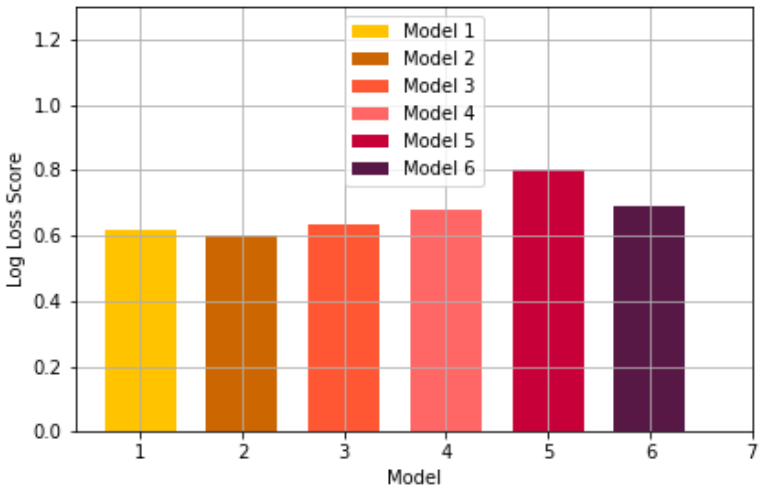

Figure 8: Log loss score of the ensemble models

Table 6 ranks the different classifiers built for the task of product classification and compares them to the benchmark scores.

Table 6

MODEL	LOG LOSS SCORE
Kaggle Top Score	0.3805
Ensemble 2	0.6023
Ensemble 1	0.6183
Ensemble 3	0.6332
KNN	0.6546
Ensemble 4	0.6818
Ensemble 5	0.8010
Ensemble 6	0.8010
Random Forest	1.0087
Gradient Boosting	1.0728
Adaboost	2.0637
Naïve Predictor	30.52

From Table 6, we can observe that all the developed models perform significantly better than the Naïve Predictor (worst case benchmark) which has no intelligence built in. Among the found individual classifiers, KNN has the lowest log loss score. With a Voting classifier, we are able to further improve the score. Ensemble models 1, 2 and 3 outperform KNN model. Amongst all the models developed, **Ensemble Model 2 (final model)** has the lowest log loss score at 0.6023. Ensemble Model 2 is a combination of KNN, Random Forest and Gradient Boosting model. Ensemble Model 2's performance with different target classes is visualized using a confusion matrix in the next section.

While the Ensemble Model 2 is significantly better than Naïve Predictor, it performs worse than the Kaggle top performing model. The first prize winner uses an ensemble of more than 30 models segregated in multiple layers.

5. Conclusion

We use a confusion matrix to visualise the classification accuracy of Ensemble model 6 (Fig 9).

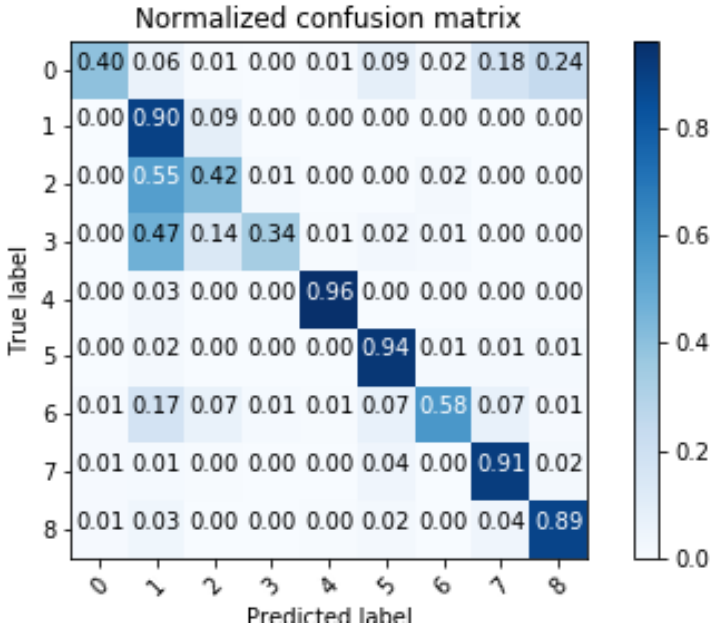

Fig 9: Confusion matrix showing the accuracy of the Ensemble model 6 in predicting different categories

The final model had more than 90% accuracy in predicting the correct class label for products in Categories 1, 4, 5, 7. Categories 1, 5 and 7 have the highest number of samples in the dataset (Figure 1). Categories 2, 3 and 0 have the lowest accuracy. It should be noted that these categories have a smaller proportion of samples available in the dataset. For categories 2 and 3, their samples are highly likely to misclassified as belonging to Class 1 which is the majority class. This indicates that while the model performs well with samples belonging to majority class (approx. 90% accuracy) but misclassifies samples belonging to the minority class. A potential way to remove the bias would be to include more number of classifiers in the ensemble, as discussed in the Improvements section below.

5.1 Reflections

This is my first machine learning project that I have worked on independently from end-to-end. It has been great opportunity to apply what I have learned in the Udacity MLND. When I was starting my capstone, I already had a domain in mind – logistics and assets management. I was already working in this domain at work and found this Kaggle competition to be relevant to the work I was doing.

The steps below summarise my workflow for the capstone project –

1) Literature study on the selected problem domain. I found several interesting articles that utilise machine learning for product classification. Additionally, I read through the Kaggle discussion forums to learn about different ways of approaching the problem and learning from them.

2) Setting up the development environment with Python and importing the dataset

3) Setting up benchmarks and evaluation metric

4) Data cleaning and exploratory analysis to identify the necessary pre-processing steps

5) Data pre-processing and dimensionality reduction

6) Model training and parameter tuning

7) Testing the model and evaluating results against the benchmark

The most difficult part of the process for me was dimensionality reduction. With 93 features, the feature set is quite large. Additionally, as there is no feature description, the approach had to be more algorithmic instead of intuitive. The large feature set also required greater computational resources for model training. I focussed on algorithms that are less computationally intensive.

The most interesting part of the project for me was using voting based ensemble learning. Being new to this technique, I used online resources and trial-error to learn this approach. It was interesting to see how multiple algorithms can be combined to enhance the overall performance.

5.2 Improvements

Based on the survey of research studies in product classification domain and top scoring scripts in Kaggle product classification challenge, it is observed that there is significant improvement in the model performance with more number of classification algorithms in the final ensemble. Including algorithms such as support vector classifier, neural networks and XGBoost can help to further discover hidden patterns in the data. Given the high dimensionality of data, training these algorithms requires significant computational resources.

Additionally, we can explore non-linear dimensionality reduction techniques such as t-SNE. t-Distributed Stochastic Neighbor Embedding (t-SNE) is useful for visualizing high dimensional datasets and dimensionality reduction.

6. References

[1] https://www.shopify.com/enterprise/global-ecommerce-statistics

[2] Everybody Likes Shopping! Multi-class Product Categorization for e-Commerce, Kozareva Z., *Human Language Technologies: The 2015 Annual Conference of the North Americal Chapter of the ACL*, 2015.

[3] E-Commerce Product Categorization, Gottipati S and Vauhkonen M.
URL: http://cs229.stanford.edu/proj2012/GottipatiVauhkonen-ProductCategorization.pdf

[4] Applying Machine Learning to Product Categorization, Shankar S and Lin I.
http://cs229.stanford.edu/proj2011/LinShankar-Applying%20Machine%20Learning%20to%20Product%20Categorization.pdf

[5] Cross-Domain Product Classification with Deep Learning, de Oliveira L, Rodrigo A. L. and Abu A.
http://cs229.stanford.edu/proj2014/Luke%20de%20Oliveira,%20Alfredo%20Lainez,%20Akua%20Abu,%20Cross-Domain%20Product%20Classification%20with%20Deep%20Learning.pdf

[6] Otto Kaggle Competition: https://www.kaggle.com/c/otto-group-product-classification-challenge

[7] Otto Leaderboard: https://www.kaggle.com/c/otto-group-product-classification-challenge/leaderboard