HECTO

HP Engineering Classifier Tool

Sasquatch Engineering team:

Reed Ballesteros, Nan Li, Manojkumar Damodaran Pillai, Shawn Tay

November 30th, 2023

Contents

Table of Figures

List of Tables

Abstract

Our project aims to automate UNSPSC Code assignments to products by developing a machine learning model using product # and product descriptions. We employ various NLP techniques and diverse algorithms, highlighting logistic regression, neural networks, and five more methods. Through extensive experiments, our TF-IDF-based 128-node Keras Sequential Neural Network emerges as the most effective, attaining 93.6% accuracy on a defined dataset. Despite limitations, including restricted code scope and generalization concerns, this model showcases promise in supplementing engineers' tasks, leading to cost reduction and operational efficiency improvements at HP.

1. Introduction

At HP, the manual assignment of UNSPSC classifications to products has been a fundamental process. However, this approach demands significant skilled human resources and involves a manual process susceptible to inefficiencies and errors. With an expanding product catalog, the urgent need for an automated and streamlined classification process has become increasingly evident.

The United Nations Standard Products and Services Code (UNSPSC) stands as a pivotal framework employed globally to categorize products and services within intricate supply chains. Its implementation fosters enhanced visibility and facilitates cost-effective procurement strategies (UNSPSC 2022).

Leveraging the capabilities of natural language processing and classification algorithms, our project aims to automate the assignment of UNSPSC Code classifications to products in HP. We explore to develop a machine learning model capable of accurately categorizing products based on their product # and descriptions. This innovative approach is set to significantly decrease manual labor, enhance precision, and simplify the product categorization procedure. The resulting automation carries the potential to expand data operations and enhance overall operational efficiency, ultimately benefiting diverse aspects of HP's supply chain and procurement workflows.

In the field of Natural Language Processing (NLP), short text classification is one of the important tasks. Unlike paragraphs or documents, short texts are more ambiguous since they do not have enough contextual information, which poses a great challenge for classification (Wang et al. 2020). In this project, we have a dataset of around 25,000 products with short text entries consisting of concise text snippets of product names and descriptions. This unique format requires specialized approaches for accurate categorization and information extraction. Our approach integrates both theoretical design and practical performance analysis obtaining the ideal result through extensive experiments. To the best of our knowledge, it is a novel classification solution tailored to fit with HP's specific operational needs. Instead of adopting a broad approach, we will craft a system that can acutely capture the distinct characteristics of the product information. The breakthrough of our project is that solutions are experimentally compared in the same dataset.

2. Technical Framework

The Technical Framework of this project is outlined through a visual representation Figure 1, illustrating the flow of methodologies and processes employed.

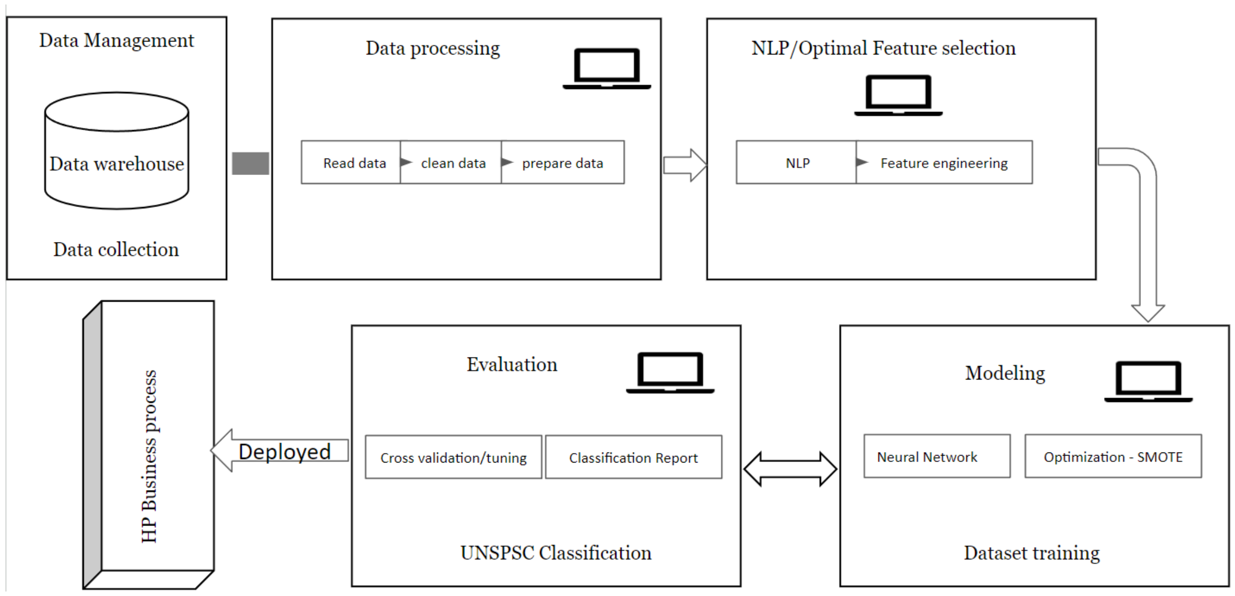


Figure 1 Technical Framework

2.1 Data Collection and Cleaning

The initial phase involves gathering a comprehensive dataset of over 25,000 product components from HP. Each product has a textual description of 3 fields: product #, UNSPSC code that is the class label explained below, and product description. The dataset covers a wide range of products. Each product is labeled by an 8-digit code as belonging to a leaf class in a large taxonomy. The taxonomy is called the United Nations Standard Product and Service Code (UNSPSC). It is the multi-sector standard in US industry for hierarchical classification of general products and services. It has four levels, representing segment, family, class, and commodity, respectively. Sometimes, there is a 4th level called business function, which is usually performed by an organization in support of the commodity. Each node in a level is specified by a 2-digit code and a text description. We identify a leaf class by an 8-digit code, concatenating the 2-digit codes along the path from the first level to the fourth level. The whole UNSPSC taxonomy has more than 17,000 leaf categories and is still increasing. Our dataset covers products in about 200 classes. The statistics of the preprocessed dataset used in our experiments are shown as Table 1. The class distribution is highly skewed as shown in Figure 2.

The EDA data cleaning process focuses on removing missing values and duplicated product records in the dataset. Assumptions we make during EDA are:

- Scope of Representation: the dataset may not cover the entire framework of UNSPSC categories, we assume that it comprehensively represents all types of products that HP handles. Each product category within HP's operations is assumed to be present in the data.
- Label Accuracy: The UNSPSC codes associated with each product description in our dataset are accurate.
- Consistency and Relevance: The descriptions used in the dataset are consistent and relevant within the company's context. They maintain a standard that is uniform across HP’s operations.

Number of Products	21476
Segments (1st level)	26
Family (2nd level)	52
Class (3rd level)	100
Commodity (4th level)	197
Business Function (5th level)	1
Average ± std of description length	31.1±12.2

Table 1 Statistics of the dataset

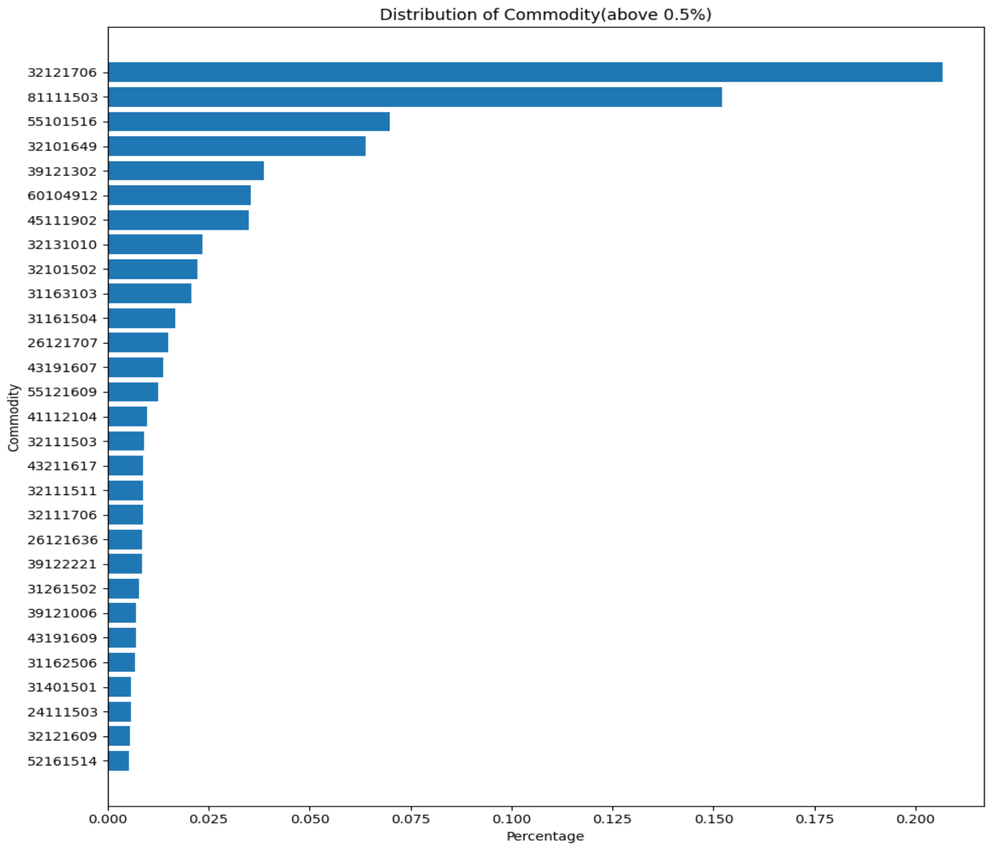


Figure 2 Commodity Distribution in the Dataset

2.2 Text Exploration and Feature Extraction

NLP (Natural Language Processing) is a branch of artificial intelligence focused on enabling machines to understand, process, and generate human language. In this context, NLP techniques are employed to work with the short text entries in our dataset.

2.2.1 Tokenization

This process involves breaking down a text into individual units. These tokens serve as building blocks for further analysis. This involved breaking down the text using common delimiters like spaces and punctuation. Additionally, common words without significant meaning (stopwords) were removed (Mielke et al. 2021). Understanding that the presence of digits within product descriptions often refers to size or dimensions rather than contributing significantly to product classification, we hypothesized that these numeric values might offer limited informational value in this context. As part of our approach, we implemented a customized tokenization process specifically tailored to remove standalone numbers, fractions, and percentages. This step was taken to minimize the noise within the data, focusing instead on more distinctive textual elements.

2.2.2 Feature Extraction

When working with text data, machine learning models require numerical input. This involves converting words or tokens into numerical values that the model can process. These numerical representations capture the underlying relationships between words, enabling the model to make predictions.

Techniques for vectorization include:

- Bag of Words (BoW): It is a robust starting point for text analysis. This technique represents text as a collection of individual words, disregarding grammar, and word order. Each unique word is assigned a numerical value based on its frequency in a document or across a corpus. BoW is effective for capturing the presence or absence of specific words, but it doesn't consider the context in which they appear (Machine Learning Mastery 2019).
- Word Embeddings (Word2Vec): Word embeddings provide a more nuanced representation of words, capturing semantic similarities. They map words to a continuous vector space, where similar words are positioned closer together. Word2Vec learns distributed representations of words based on their co-occurrence patterns (Suri 2022).
- TF-IDF (Term Frequency-Inverse Document Frequency): TF-IDF assigns a weight to each word in a document based on how often it appears in that document relative to its frequency across a corpus. In the context of product descriptions, it can help identify key characteristics that differentiate products.

- GloVe embeddings: GloVe stands for Global Vectors for Word Representation. It's an unsupervised learning technique used to represent words as vectors in high-dimensional space. These vectors capture semantic meaning and relationships between words based on their co-occurrence probabilities in a corpus of text (Ganegedara 2019).

Given the unique nature of our product descriptions, comprising succinct text snippets with inherent ambiguity and technical abbreviations, we have realized that traditional text processing techniques may not seamlessly align with our dataset's characteristics. To address this, all four NLP techniques have been experimented in our research. By testing various approaches, the goal is to pinpoint the one that effectively extracts the key information, making product classification precise.

2.3 Machine Learning Model Development

In a classification task, the goal is to assign a category or label to each input. Here, the classification model aims to predict the appropriate UNSPSC code for a given product description and product #. The following algorithms are experimented in our research:

- Logistic Regression: Logistic regression models the probability that a given input belongs to a particular category. It is relatively simple and interpretable. It can handle both linear and non-linear relationships between features and the target variable. It serves as a baseline model for our classification tasks.
- Multi-Layer Perceptron (MLP) Classifier: Multilayer Perceptron (MLP) is the most fundamental type of neural network architecture when compared to other major types such as Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Autoencoder (AE) and Generative Adversarial Network (GAN). With their ability to model non-linear relationships, flexibility in handling different data types, and feature learning capabilities, MLP classifiers have become a popular choice for various applications (Singh 2023).
- Neural Networks: Neural networks consist of interconnected nodes (neurons) organized in layers. In our project, we employ the ReLU activation function within each dense hidden layer, whereas the final output classification layer utilizes the softmax activation

function. The model compiler is configured using the Adam optimizer. It computes categorical cross-entropy loss through the SparseCategoricalCrossentropy class. To mitigate overfitting, a dropout layer with a rate of 0.5 is included after the dense layer. Furthermore, the most effective neural networks from each fitting session are saved based on improved validation cross-entropy loss.

- Random Forest: Random Forest operates by building multiple decision trees during the training phase. Each tree is trained on a random subset of data and a random subset of features (words or word vectors in our case). For classification, the individual decision trees make predictions, and the final prediction from the Random Forest is determined by aggregating the predictions from each tree (e.g., majority voting for classification). Random Forest can handle high-dimensional data effectively, making it suitable for text data with a large number of features (words or word embeddings). It's robust to overfitting and generally requires less hyperparameter tuning compared to other complex models (Jalal et al 2022).
- Multinomial Naive Bayes classifier: It is a variant of the Naive Bayes algorithm and suitable for classification problems with discrete features (e.g., word counts or frequency of terms in a document). It assumes that features follow a multinomial distribution. MultinomialNB estimates the parameters (probabilities) of the features for each class using maximum likelihood estimation or smoothing techniques. It's known for its computational efficiency, scalability, and ability to handle high-dimensional data (Ratz 2021).
- Support Vector Classifier (SVC): SVC aims to find a decision boundary (hyperplane in high-dimensional space) that best separates different classes while maximizing the margin between them. This margin is defined by support vectors, which are the data points closest to the decision boundary. SVC can handle high-dimensional data efficiently, making it suitable for text data with a large number of features (such as word frequencies, TF-IDF scores, or word embeddings) (Pranckevičius 2017).
- XGBoost Classifier: XGBoost employs the boosting ensemble technique, which involves building a series of decision trees sequentially. Each tree tries to correct the errors made by the previous tree. It optimizes the model's performance by minimizing

the loss function using gradient descent, gradually improving the model's accuracy. XGBClassifier handles high-dimensional data efficiently, making it suitable for text data with a large number of features (e.g., word frequencies, TF-IDF scores, or word embeddings) (Bhatnagar 2023).

By testing these diverse algorithms, it allows us to make an informed decision about which algorithm is best suited for accurately predicting UNSPSC codes. The most effective approach will be the one that demonstrates the highest predictive accuracy and generalization ability on unseen data. This chosen model will serve as the basis for predicting UNSPSC codes for new product descriptions.

2.4 Model Evaluation

Recognizing the potential challenges posed by class imbalance of our data, we addressed this issue through oversampling techniques, ensuring that the model is trained on a balanced representation of both majority and minority classes. It's also important to evaluate the model's performance after applying oversampling to ensure that it effectively addresses class imbalance and leads to improved classification results. The evaluation parameters, namely precision, recall and F1, are used to assess the effect of UNSPSC classification to ensure that the chosen algorithm is equipped to reliably predict UNSPSC codes for product descriptions and numbers in real-world scenarios.

3. Experimental Result and Analysis

3.1 Experiment design

Our study initiates a comprehensive exploration to optimize the predictive performance of classification models for UNSPSC categorization. Our investigation spans five distinct experimental phases, each specifically designed to explore different aspects of model construction. The first batch, served as our baseline models, is conducted to find the optimal predictors (Product # and Descriptions) and data preparation methods (tokenization and vectorization techniques) for logistic regression. The following experiments, the core of our study, explore oversampling techniques via Synthetic Minority Oversampling Technique (SMOTE), and various ways of vectorization—BagofWords, GloVe embeddings, TF-IDF, and Word2Vec on all the proposed algorithms. All our experiments employ the top 50 Commodity UNSPSC codes, which constituted approximately 92% of the data, totaling 19,720 out of 21,475 usable rows. The dataset was partitioned into three subsets: 80% for training, 10% for validation, and 10% for testing purposes. Table 2,3,4 and 5 present the details of our experimental design.

3.2 Experimental Results

3.2.1 Batch 1 baseline models

Table 2 shows the results of various experiments conducted in Batch 1 for establishing baseline models. Here are some insights:

- Predictors: Using "Product #" as the predictor seems to improve the performance metrics.
- Data Preparation: Removing digits, fractions, or percentages from Descriptions during tokenization did not enhance the model performance, compared to using delimiters and removing stopwords for tokenization.
- Vectorization Methods: The "Bag of Words" vectorization method shows better performance compared to TF-IDF and Word2Vec vectorization.

Overall, using both Product Descriptions and Product # as the predictor along with Bag of Words as our vectorization method on logistic regression has already shown very promising model performance results.

3.2.2 Batch 2 Experiments

In the results obtained from batch 1 experiments, we maintain the use of Product # and Description as predictors. The tokenization process which involves utilizing delimiters and removing stopwords also remains consistent across subsequent experiments. Batch 2 experiments (Table 3) focus on oversampling with SMOTE alongside Bag of Words vectorization across various algorithms. The results show that Keras Sequential Neural Network yields the highest accuracy, precision, recall, and F1-Score, indicating a strong combination of factors contributing to better results compared to our baseline model.

3.2.3 Batch 3 Experiments

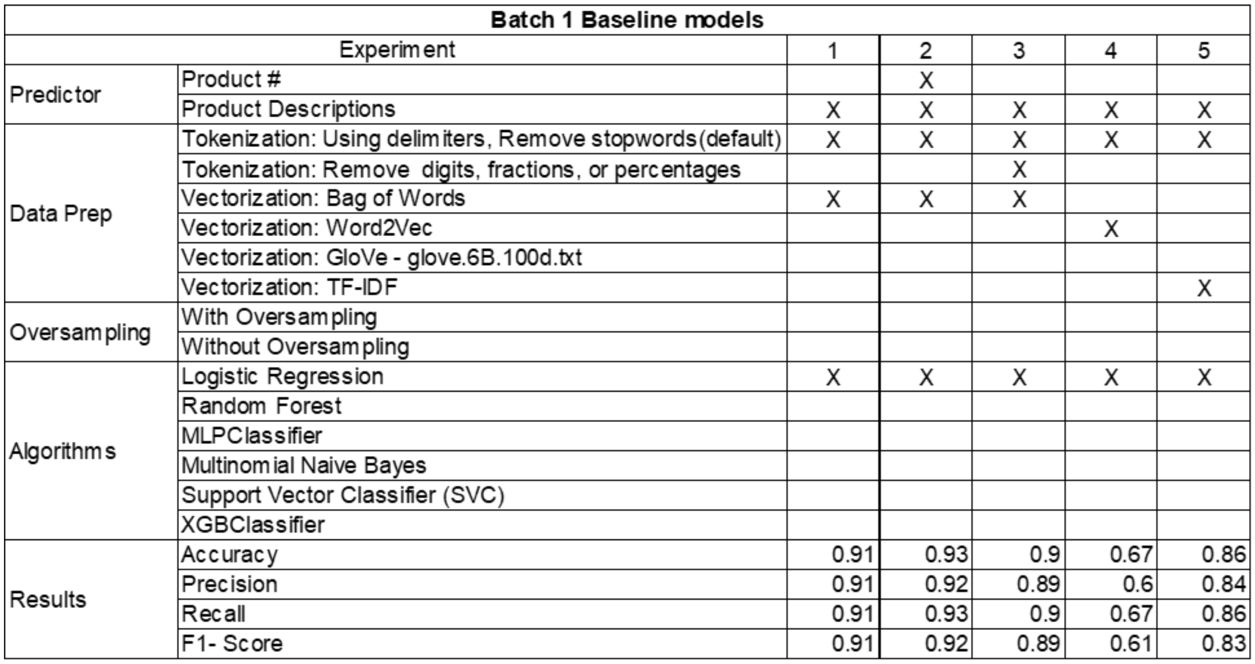
Batch 3 (Table 4) introduces GloVe as a vectorization method without oversampling. Experiments 4 and 7 show higher scores in metrics, suggesting that the Keras Sequential Neural Network and XGBClassifier algorithms along with GloVe vectorization are more reliable models.

3.2.4 Batch 4 Experiments

TF-IDF as the vectorization method alongside SMOTE oversampling are consistently employed in Batch 4 experiments (Table 5). All experiments show notably higher scores in performance metrics except experiment 5 (Multinomial Naive Bayes) compared to our baseline model and even the best results from batch 2 and batch 3.

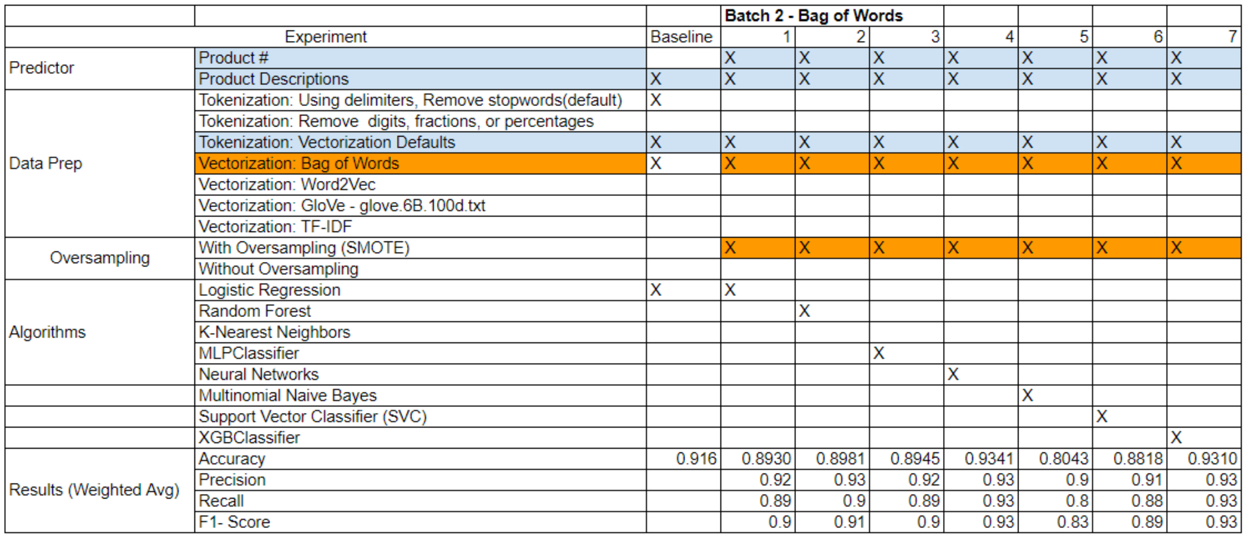
3.2.5 Batch 5 Experiments

Batch 5 experiments (Table 6) focus on exploring the combination of Word2Vec with SMOTE oversampling across all algorithms. Experiment 6 (SVC) and 7(XGBClassifer) stand out with higher scores across all metrics, suggesting their effectiveness in classification. Given these results, though, the best-performing algorithms using this vectorization do not achieve as good performance in accuracy compared to the same algorithms used under TF-IDF vectorization.



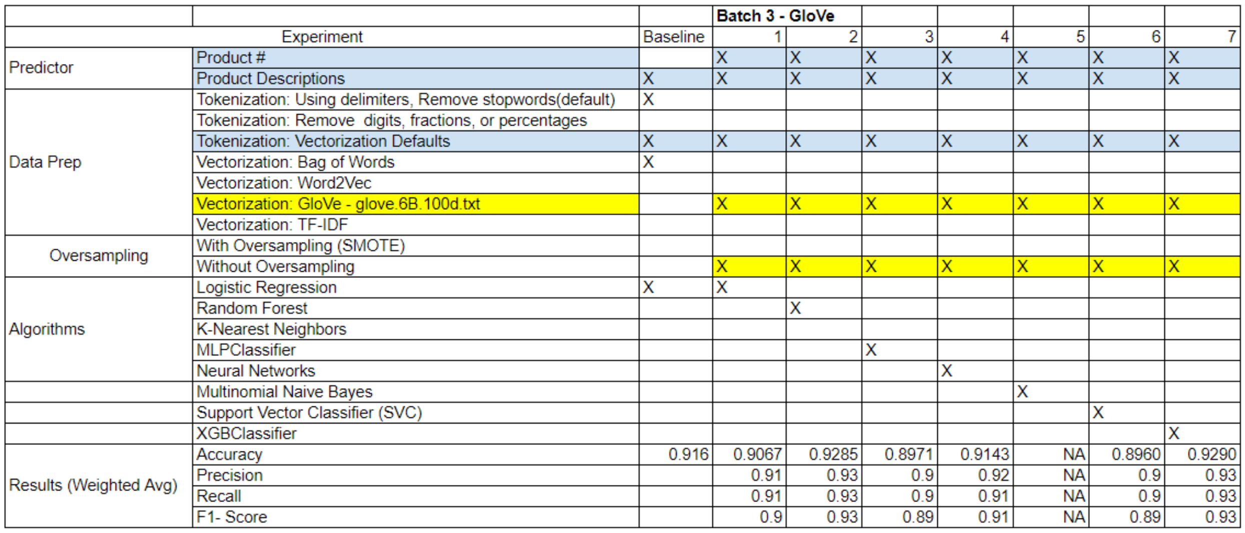
Batch 1 Baseline models						
Experiment		1	2	3	4	5
Predictor	Product #		X			
	Product Descriptions	X	X	X	X	X
Data Prep	Tokenization: Using delimiters, Remove stopwords(default)	X	X	X	X	X
	Tokenization: Remove digits, fractions, or percentages			X		
	Vectorization: Bag of Words	X	X	X		
	Vectorization: Word2Vec				X	
	Vectorization: GloVe - glove.6B.100d.txt					
	Vectorization: TF-IDF					X
Oversampling	With Oversampling					
	Without Oversampling					
Algorithms	Logistic Regression	X	X	X	X	X
	Random Forest					
	MLPClassifier					
	Multinomial Naive Bayes					
	Support Vector Classifier (SVC)					
	XGBClassifier					
Results	Accuracy	0.91	0.93	0.9	0.67	0.86
	Precision	0.91	0.92	0.89	0.6	0.84
	Recall	0.91	0.93	0.9	0.67	0.86
	F1- Score	0.91	0.92	0.89	0.61	0.83

Table 2 Batch 1 Experiments - Baseline Models



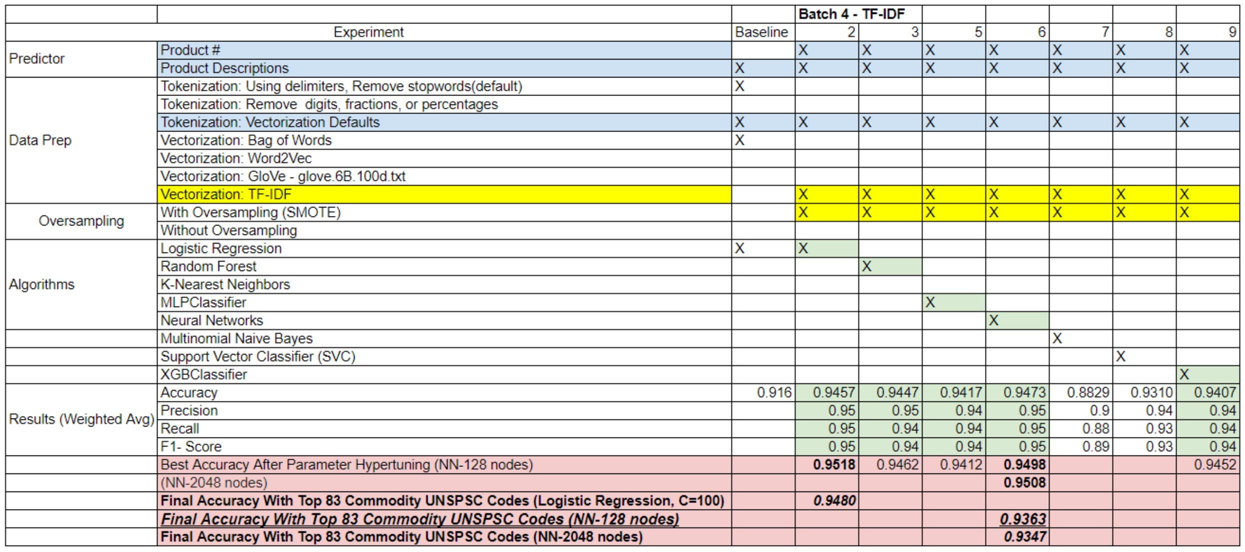
			Batch 2 - Bag of Words						
Experiment		Baseline	1	2	3	4	5	6	7
Predictor	Product #		X	X	X	X	X	X	X
	Product Descriptions	X	X	X	X	X	X	X	X
Data Prep	Tokenization: Using delimiters, Remove stopwords(default)	X							
	Tokenization: Remove digits, fractions, or percentages								
	Tokenization: Vectorization Defaults	X	X	X	X	X	X	X	X
	Vectorization: Bag of Words	X	X	X	X	X	X	X	X
	Vectorization: Word2Vec								
	Vectorization: GloVe - glove.6B.100d.txt								
	Vectorization: TF-IDF								
Oversampling	With Oversampling (SMOTE)		X	X	X	X	X	X	X
	Without Oversampling								
Algorithms	Logistic Regression	X	X						
	Random Forest			X					
	K-Nearest Neighbors								
	MLPClassifier				X				
	Neural Networks					X			
	Multinomial Naive Bayes						X		
	Support Vector Classifier (SVC)							X	
	XGBClassifier								X
Results (Weighted Avg)	Accuracy	0.916	0.8930	0.8981	0.8945	0.9341	0.8043	0.8818	0.9310
	Precision		0.92	0.93	0.92	0.93	0.9	0.91	0.93
	Recall		0.89	0.9	0.89	0.93	0.8	0.88	0.93
	F1- Score		0.9	0.91	0.9	0.93	0.83	0.89	0.93

Table 3 Batch 2 Experiments - Bag of Words Vectorization



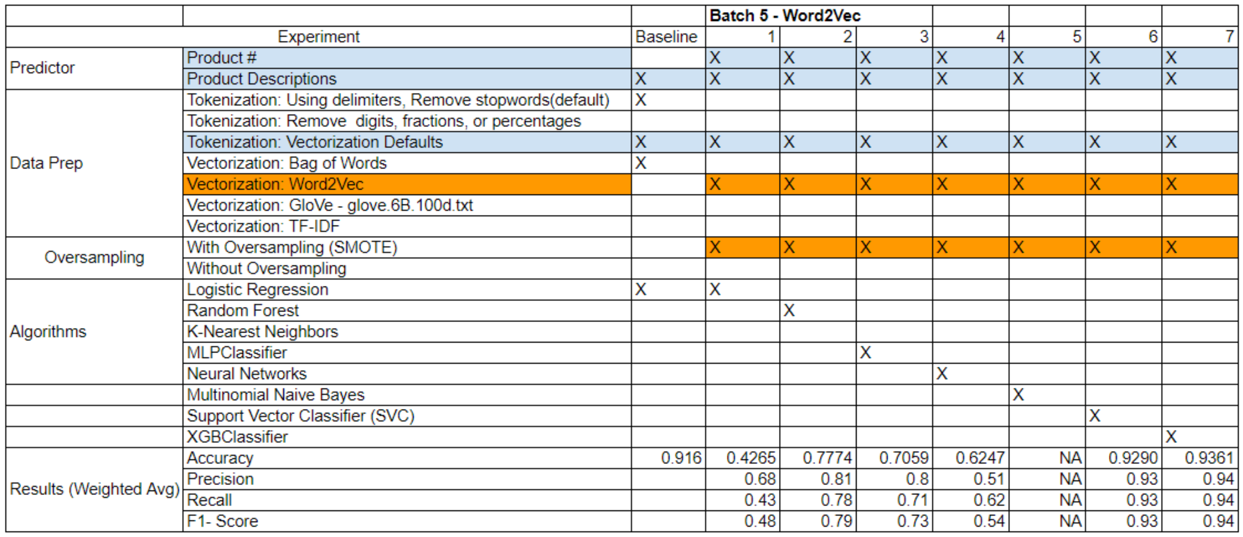
			Batch 3 - GloVe						
	Experiment	Baseline	1	2	3	4	5	6	7
Predictor	Product #		X	X	X	X	X	X	X
	Product Descriptions	X	X	X	X	X	X	X	X
Data Prep	Tokenization: Using delimiters, Remove stopwords(default)	X							
	Tokenization: Remove digits, fractions, or percentages								
	Tokenization: Vectorization Defaults	X	X	X	X	X	X	X	X
	Vectorization: Bag of Words	X							
	Vectorization: Word2Vec								
	Vectorization: GloVe - glove.6B.100d.txt		X	X	X	X	X	X	X
	Vectorization: TF-IDF								
Oversampling	With Oversampling (SMOTE)								
	Without Oversampling		X	X	X	X	X	X	X
Algorithms	Logistic Regression	X	X						
	Random Forest			X					
	K-Nearest Neighbors								
	MLPClassifier				X				
	Neural Networks					X			
	Multinomial Naive Bayes						X		
	Support Vector Classifier (SVC)							X	
	XGBClassifier								X
Results (Weighted Avg)	Accuracy	0.916	0.9067	0.9285	0.8971	0.9143	NA	0.8960	0.9290
	Precision		0.91	0.93	0.9	0.92	NA	0.9	0.93
	Recall		0.91	0.93	0.9	0.91	NA	0.9	0.93
	F1- Score		0.9	0.93	0.89	0.91	NA	0.89	0.93

Table 4 Batch 3 Experiments - GloVe Vectorization



			Batch 4 - TF-IDF						
	Experiment	Baseline	2	3	5	6	7	8	9
Predictor	Product #		X	X	X	X	X	X	X
	Product Descriptions	X	X	X	X	X	X	X	X
Data Prep	Tokenization: Using delimiters, Remove stopwords(default)	X							
	Tokenization: Remove digits, fractions, or percentages								
	Tokenization: Vectorization Defaults	X	X	X	X	X	X	X	X
	Vectorization: Bag of Words	X							
	Vectorization: Word2Vec								
	Vectorization: GloVe - glove.6B.100d.txt								
	Vectorization: TF-IDF		X	X	X	X	X	X	X
Oversampling	With Oversampling (SMOTE)		X	X	X	X	X	X	X
	Without Oversampling								
Algorithms	Logistic Regression	X	X						
	Random Forest			X					
	K-Nearest Neighbors								
	MLPClassifier				X				
	Neural Networks					X			
	Multinomial Naive Bayes						X		
	Support Vector Classifier (SVC)							X	
	XGBClassifier								X
Results (Weighted Avg)	Accuracy	0.916	0.9457	0.9447	0.9417	0.9473	0.8829	0.9310	0.9407
	Precision		0.95	0.95	0.94	0.95	0.9	0.94	0.94
	Recall		0.95	0.94	0.94	0.95	0.88	0.93	0.94
	F1- Score		0.95	0.94	0.94	0.95	0.89	0.93	0.94
	Best Accuracy After Parameter Hypertuning (NN-128 nodes)		**0.9518**	0.9462	0.9412	**0.9498**			0.9452
	(NN-2048 nodes)					**0.9508**			
	Final Accuracy With Top 83 Commodity UNSPSC Codes (Logistic Regression, C=100)		***0.9480***						
	Final Accuracy With Top 83 Commodity UNSPSC Codes (NN-128 nodes)					***0.9363***			
	Final Accuracy With Top 83 Commodity UNSPSC Codes (NN-2048 nodes)					***0.9347***			

Table 5 Batch 4 Experiments - TF-TDF Vectorization



			Batch 5 - Word2Vec						
	Experiment	Baseline	1	2	3	4	5	6	7
Predictor	Product #		X	X	X	X	X	X	X
	Product Descriptions	X	X	X	X	X	X	X	X
Data Prep	Tokenization: Using delimiters, Remove stopwords(default)	X							
	Tokenization: Remove digits, fractions, or percentages								
	Tokenization: Vectorization Defaults	X	X	X	X	X	X	X	X
	Vectorization: Bag of Words	X							
	Vectorization: Word2Vec		X	X	X	X	X	X	X
	Vectorization: GloVe - glove.6B.100d.txt								
	Vectorization: TF-IDF								
Oversampling	With Oversampling (SMOTE)		X	X	X	X	X	X	X
	Without Oversampling								
Algorithms	Logistic Regression	X	X						
	Random Forest			X					
	K-Nearest Neighbors								
	MLPClassifier				X				
	Neural Networks					X			
	Multinomial Naive Bayes						X		
	Support Vector Classifier (SVC)							X	
	XGBClassifier								X
Results (Weighted Avg)	Accuracy	0.916	0.4265	0.7774	0.7059	0.6247	NA	0.9290	0.9361
	Precision		0.68	0.81	0.8	0.51	NA	0.93	0.94
	Recall		0.43	0.78	0.71	0.62	NA	0.93	0.94
	F1- Score		0.48	0.79	0.73	0.54	NA	0.93	0.94

Table 6 Batch 5 Experiments - Word2Vec Vectorization

3.2.6 Additional Experiments with Hyperparameter Tuning

From the batches of experiments listed above, we found the models developed under TF-IDF vectorization (Batch 4) produced the highest yields in accuracy matching product number and description to UNSPSC commodity codes for most algorithms. Our next step is to perform hyperparameter tuning on these models to possibly find better performance. Due to time constraints, hyperparameter tuning involved only a small selection of variables for each algorithm for cross-validation training, such as the C parameter (the inverse of regularization strength) for logistic regression, hidden layer sizes for the Keras Sequential neural networks and MLP classifier, n_estimators (number of trees in forest) and max_depth (maximum depth of a tree) for random forest classifier, and n_estimators and learning_rate for XGBoost.

After hyperparameter tuning, we found the top three highest-performing models under TF-IDF vectorization r logistic regression with C=100 at 95.2% accuracy, a single-layer 128-node Keras Sequential neural network at 95%, and a single-layer 2048-node Kera Sequential neural network at 95.1%. With these models, our next step is to fit and train on a larger subset of data to determine our best overall model.

3.3 Final Model Evaluation

3.3.1 Experiment Setup Limitations

While our dataset contains 197 UNSPSC commodity codes, a vast majority of them are rarely used, with sporadic appearances in the data. These codes are utilized more as special cases than normal classification categories. From our EDA, 114 of these codes account for just over 3% of the usable data, which is 707 out of 21475 rows. Due to this very sparse distribution of codes on the tail end, we come to the realization that creating a model based on generating all codes is not feasible given the size of the current dataset, as these 114 codes do not provide enough data for model training.

Moving forward, we limit the dataset for evaluating our final models to UNSPSC commodity codes that have at least 20 samples in the original dataset, which results in 83 of the most frequently used codes. While a majority of codes have been cut, these 83 codes still represent almost 97% of the data (20768 out of 2145 rows), which would still cover most cases for UNSPSC commodity code generation. This limit on code provides us diverse samples of test and validation datasets, while also allows us to oversample the training data via SMOTE.

3.3.2 Final Model Results

Applying the hyperparameter-tuned models on the expanded data subset resulted with the logistic regression model with 94.8% accuracy, the single-layer 128-node Keras Sequential neural network with 93.6% accuracy, and the 2048-node Keras Sequential neural network with 93.5% accuracy. We expect these results to be slightly lower compared to the initial experiments from section 3.2.6 due to using the expanded data subset.

While the Logistic Regression model yielded the highest accuracy on the larger dataset, we also need to consider several factors when it comes to determining our best overall model. The Logistic Regression model yields higher accuracy over the neural network models as it utilizes a larger training set and no validation set. Also, hyperparameter tuning of the model from cross-validation training determines to set parameter C, the inverse of regularization strength, to 100, which could be considered an excessive value such that little to no regularization would be utilized, leading to the model overfitting the training data and possibly result in poor generalization when it comes to generating UNSPSC codes for new Product Number and Description data. For the 2048-node neural network, we find during model fitting that validation loss ceases to improve only after five epochs, which shows signs of the model also overfitting the training data, leading to poor generalization as well.

The training and validation cross-entropy loss chart below (Figure 3) does not demonstrate a smooth, gradual model fit.

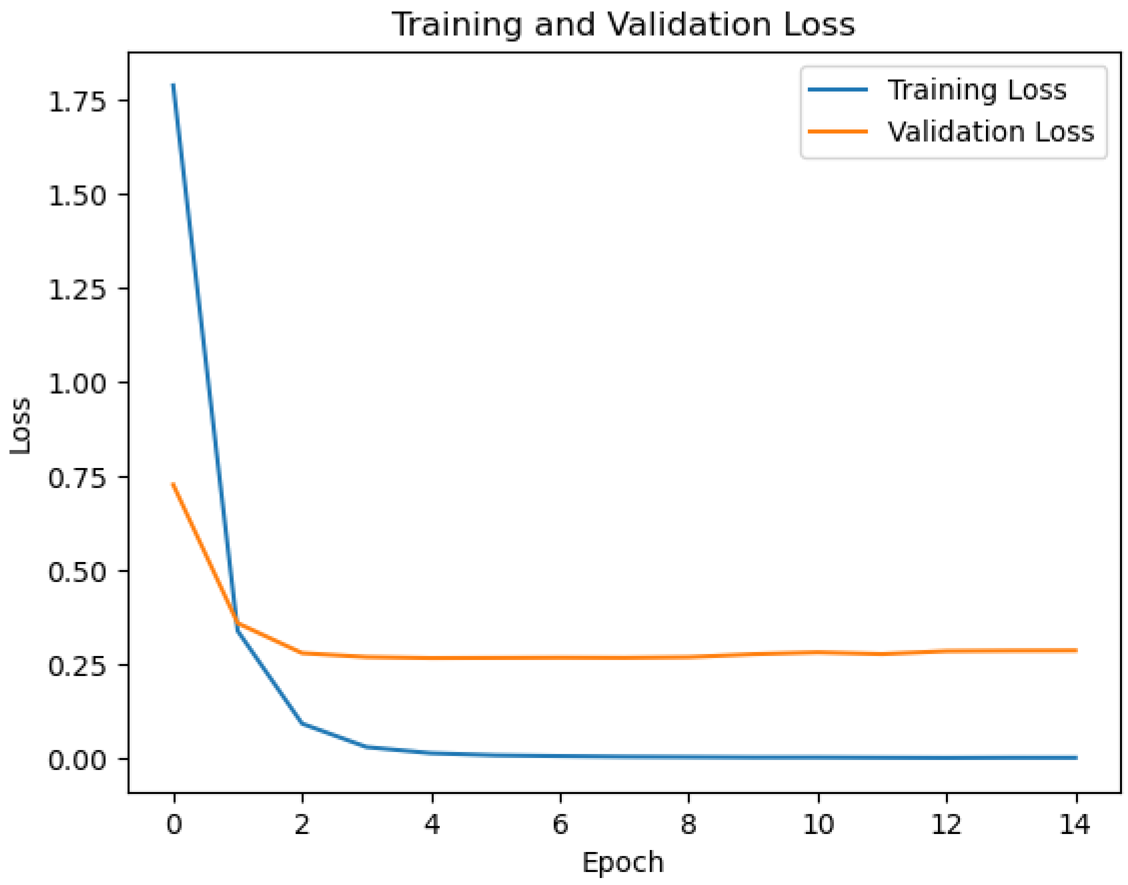


Figure 3 2048-Node Keras Sequential Neural Network Cross-Entropy Loss

Due to model overfitting and the potentially poor generalization issues with these two models, we would select the 128-node Keras Sequential Neural Network as our best overall model with 93.6% accuracy. The training and validation cross-entropy loss chart for the model on the expanded data subset (Figure 4) shows a smoother, gradual model fit over a larger number of epochs compared to the 2048-node neural network.

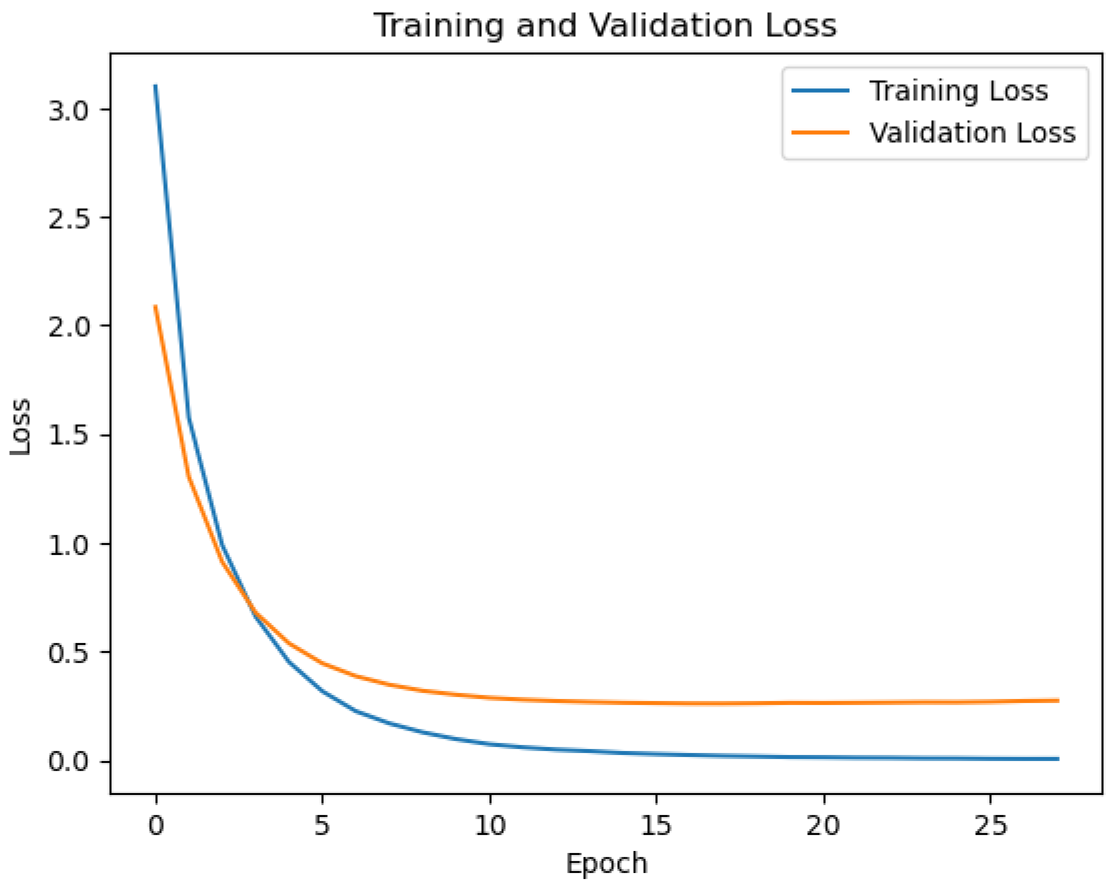


Figure 4 128-Node Keras Sequential Neural Network Cross-Entropy Loss

4. Conclusions

Through our experiments, we discover that our most effective model is a single-layer, 128-node Keras Sequential neural network integrated with TF-IDF vectorization. This model is trained on a specific data subset comprising 83 of the most frequently used UNSPSC commodity codes. Employing SMOTE oversampling has proven to be instrumental in preventing bias towards the predominant class within the dataset, thereby enhancing the overall performance of the model. This configuration showcases consistent stability and suitability for accurately classifying our particular dataset.

While our research has presented promising results, it is important to acknowledge its limitations:

- Limited Scope: Focusing on the top 83 UNSPSC commodity codes may restrict the model's generalization to less frequent or newer codes. If the dataset includes a broader range of codes, the model's performance on these less represented or newer codes might not be as reliable. That said, the 83 codes represent almost 97% of the original data, in which we feel the final model can confidently generate UNSPSC commodity codes for most new cases.
- Generalization Gap: Without testing the model's performance on entirely new or unseen data, its ability to perform with different datasets remains uncertain. This gap in assessing generalization could affect its reliability when applied to new products or descriptions outside the current data.
- External Factors: The model's performance might be influenced by external factors such as changes in market trends, product evolution, or the introduction of new UNSPSC codes.

Given the model has certain constraints listed above, it should be intended to function as a supplementary aid rather than a fully automated substitute for HP engineers when they categorize UNSPSC commodity codes for new product entries. It is crucial for subject matter experts to verify the accuracy of the code generated by the model before applying them to the products. Nevertheless, while some level of engineering oversight is still required, we anticipate that the model will significantly lighten their workload in this task, granting them the opportunity to allocate their time towards other projects more promptly. Over time, this shift will gradually yield a decrease in costs and a boost in operational efficiency for the company.

In the times ahead, following the initial deployment, the focus will shift towards ongoing enhancements, continuous training, and evaluating the model's performance as more new product data arrives. This endeavor of developing the model can serve as a gateway to delve into using machine learning across various facets of the business. Moreover, it presents an opportunity to disseminate the insights gained and the best practices discovered during the project to the broader HP community.

References

Bhatnagar Gauri. 2023. "XG Boost for text classification". Accessed November 2023. https://medium.com/@gaurishah143/xg-boost-for-text-classification-9c8b1f8f24aa

Ganegedara Thushan. 2019. "Intuitive Guide to Understanding GloVe Embeddings." Accessed November 2023. https://towardsdatascience.com/light-on-math-ml-intuitive-guide-to-understanding-glove-embeddings-b13b4f19c010

Jalal Nasir, Arif Mehmood, Gyu Sang Choi, Imran Ashraf. 2022. "A novel improved random forest for text classification using feature ranking and optimal number of trees." Journal of King Saud University - Computer and Information Sciences, 34(6): 2733-2742.

Machine Learning Mastery. 2019. "A Gentle Introduction to the Bag-of-Words Model". Accessed November 2023. https://machinelearningmastery.com/gentle-introduction-bag-words-model/.

Mielke, Sabrina J, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, et al. "Between Words and Characters: A Brief History of Open-Vocabulary Modeling and Tokenization in NLP." *ArXiv.Org*, 2021.

Pranckevičius, Tomas, and Virginijus Marcinkevičius. 2017. "Comparison of Naïve Bayes, Random Forest, Decision Tree, Support Vector Machines, and Logistic Regression Classifiers for Text Reviews Classification." Baltic Journal of Modern Computing 5, no. 2: 221-.

Ratz Arthur. 2021. "Multinomial Naïve Bayes' for Documents Classification and Natural Language Processing (NLP)". Accessed November 2023. https://towardsdatascience.com/multinomial-na%C3%AFve-bayes-for-documents-classification-and-natural-language-processing-nlp-e08cc848ce6

Singh Ravi. 2023. "Understanding MLP Classifiers: A Powerful Tool for Machine Learning." Accessed November 2023. https://www.linkedin.com/pulse/understanding-mlp-classifiers-powerful-tool-machine-learning-singh/

Suri Manan. 2022. “A Dummy’s Guide to Word2Vec”. Accessed November 2023. https://medium.com/@manansuri/a-dummys-guide-to-word2vec-456444f3c673.

UNSPSC.2022. Accessed Nov 2023. https://www.unspsc.org/

Wang, Haitao, Jie He, Xiaohong Zhang, and Shufen Liu. 2020. “A Short Text Classification Method Based on N-Gram and CNN.” *CHINESE JOURNAL OF ELECTRONICS* 29, no. 2: 248–54.

Appendix

Literature Review on Product Classification using Textual Descriptions

The task of categorizing products based on textual information has been an area of interest for researchers and industry experts. The following algorithms have been widely applied in product and text classification.

Context Matching:

Semantic matching algorithms are procedures that are capable of finding semantic relations between categories of different product catalogues. Two examples of matching algorithms, representing two different approaches, are CTXMatch algorithm described in Bouquet et al. (2003b) and the GoldenBullet system described in Ding et al. (2002). The former is based on Natural Languages Processing (NLP) techniques applied to labels occurring in the classification and the transformation of the matching problem into a satisfiability problem, while the latter uses techniques for information retrieval and machine learning applied to the content of the classification. While GoldenBullet requires a training set, CTXMatch is completely automatic and needs just two classifications as input.

Machine Learning Approaches:

In machine learning, the goal of classification is to group the items that have similar feature values. The particular ML model that is best suited often depends on the problem.

Support Vector Machines (SVMs) were amongst the earliest to be used, with Chen (2013) successfully applying multi-class SVM with margin re-scaling to optimize a proposed metric, instead of error rate or other common metrics in hierarchical classification of products.

A decision tree classifier is analogous to a flowchart diagram with the terminal nodes representing classification outputs/decisions. Hutama (2020) applied Multinomial Naïve Bayes and Decision Tree in text analysis of applicants for Personality Classification.

K-means clustering was also explored for hierarchical categorization problems. Qianhui Liang (2009) presented a clustering-based approach to Web service categorization in order to form a hierarchy of service taxonomy. They first implemented the clustering roughly by means of a simple mechanism of string matching. Secondly, they considered the underlying semantic relation among metadata structures and adopted a bisect-K Means algorithm to do clustering.

Due to the efficacy of deep learning, various neural networks have been applied in text classification, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), and the combination of CNNs and RNNs. When paired with word embeddings like Word2Vec or GloVe, they have shown remarkable capabilities in capturing semantic meanings from textual data. Kim (2014) reported on a series of experiments with convolutional neural networks (CNN) trained on top of pre-trained word vectors for sentence-level classification tasks. It showed that a simple CNN with little hyperparameter tuning and static vectors achieves excellent results on multiple benchmarks.

Our project is uniquely designed to predict UNSPSC codes specifically for the products sources from its suppliers. While there are many general classification systems, ours is tailored to fit with HP's specific operational needs. Instead of adopting a broad approach, we will craft a system that can acutely capture the distinct characteristics of the products. This specialized focus ensures more accurate categorization, streamlining the supply chain and enhancing the efficiency of HP's internal processes.

Appendix References

Blackburn, Patrick. "Modeling and Using Context: 4th International and Interdisciplinary Conference, CONTEXT 2003, Stanford, CA, USA, June 23-25, 2003: Proceedings." Berlin;: Springer, 2003.

Ding, Y., Korotkiy, M., Omelayenko, B., Kartseva, V., Zykov, V., Klein, M., Schulten, E. & Fensel, D. 2002 Goldenbullet: automated classification of product data in e-commerce. In BIS-2002: 5th International Conference on Business Information Systems, 2002.

Hutama, Nanda Yonda, Kemas Muslim Lhaksmana, and Isman Kurniawan. “Text Analysis of Applicants for Personality Classification Using Multinomial Naïve Bayes and Decision Tree.” Jurnal Infotel (Online) 12, no. 3 (2020): 72–81.

Kim, Yoon. “Convolutional Neural Networks for Sentence Classification,” 2014. https://doi.org/10.48550/arxiv.1408.5882.

Qianhui Liang, Peipei Li, P.C.K. Hung, and Xindong Wu. “Clustering Web Services for Automatic Categorization.” In *2009 IEEE International Conference on Services Computing*, 380–87. IEEE, 2009.