ISyE 6740 – Fall 2021
Project Final Report

Team Member Names: Zohreh Ebrahimi

Project Title: *Multi-Class-Multi-Label Classification of the Schools Budgeting*

Problem Statement

Budgets for schools in United states are huge, complex, and not standardized. In order to compare budget or expenditure data across districts, the related organization (ERS) assigns every line item to certain categories in a comprehensive financial spending framework. Each year hundreds of hours are spent on manual labelling. It's no easy task to digest where and how schools are using their resources.

Our project is a multi-class-multi-label classification problem with the goal of attaching canonical labels to the freeform text in budget line items. These labels let ERS, understand how schools are spending money and tailor their strategy recommendations to improve outcomes for students, teachers, and administrators.

Data Source

This project is part of the competition that is in progress in DrivenData website. The data set which is downloadable via (https://www.drivendata.org), is in the '.csv' format in two separate training and testing files. The point of the test data, which is held out from the development phase, is to provide a fair test for machine learning competitions while the labels are not known by anyone but the competition hosts.

Evaluating the type of data and encoding the labels

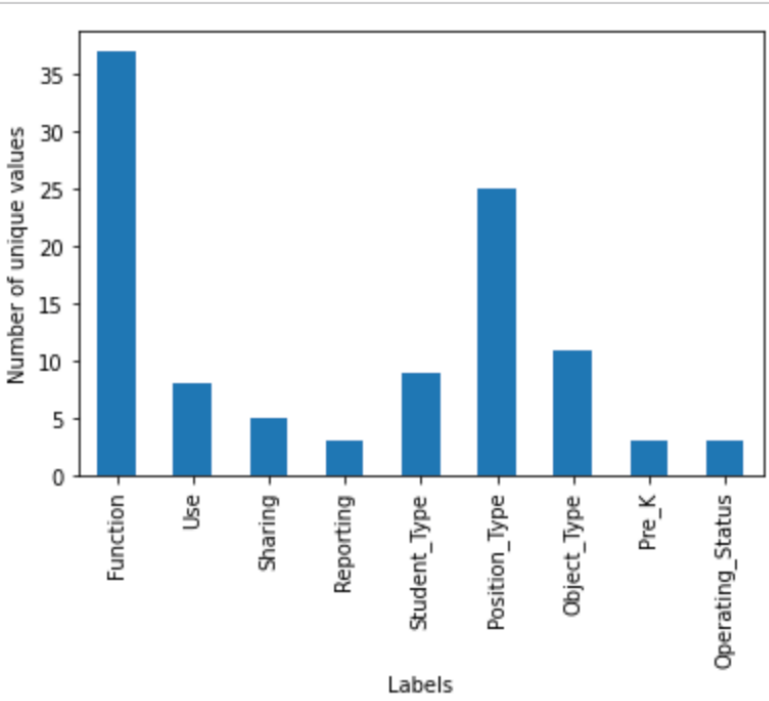
For each line-item we have a set of labels attached which are our target variables. So, we are going to create a supervised learning algorithm which can predict the label of each line-item for the unknown new line items. We have over 100 target variables which could attach to a single line-item. For instance, Object_Type describes what the spending "is"—Base Salary/Compensation, Benefits, Stipends & Other Compensation, Equipment & Equipment Lease, Property Rental, and so on.

Other categories describe what the spending "does," which groups of students benefit, and where the funds come from. For example, are we spending more on textbooks than the neighboring school? This task is very time-consuming and labor-intensive. Since we are going to predict a category for each line-item, the problem falls in classification field as opposed to the regression models which predicts the numerical output. Here are some of the actual categories which we need to determine.

- Pre-k:
  - No-label
  - Non PreK
  - PreK
- Reporting:
  - No-Label
  - Non-School
  - School
- Sharing:
  - Leadership and management
  - No-Label
  - School Reported
- Student_Type:
- Alternative
- At Risk
- …

Overall, there are 9 columns with many distinct categories in each column which is a multi-class-multi-label classification problem with 9 broad categories that each take on many possible sub-label instances. It is impossible for human to predict the label of the line items with 100% accuracy.

By taking this into account, we are not going that just say this line-item belongs to this category, but also want to say it is most likely to be in this category such as textbook. So, we will predict probabilities for each label and by making these suggestions the analyst can prioritize their time. By plotting the histogram of the labels, it is obvious that data is not distributed uniformly, and we are dealing with imbalance classification.



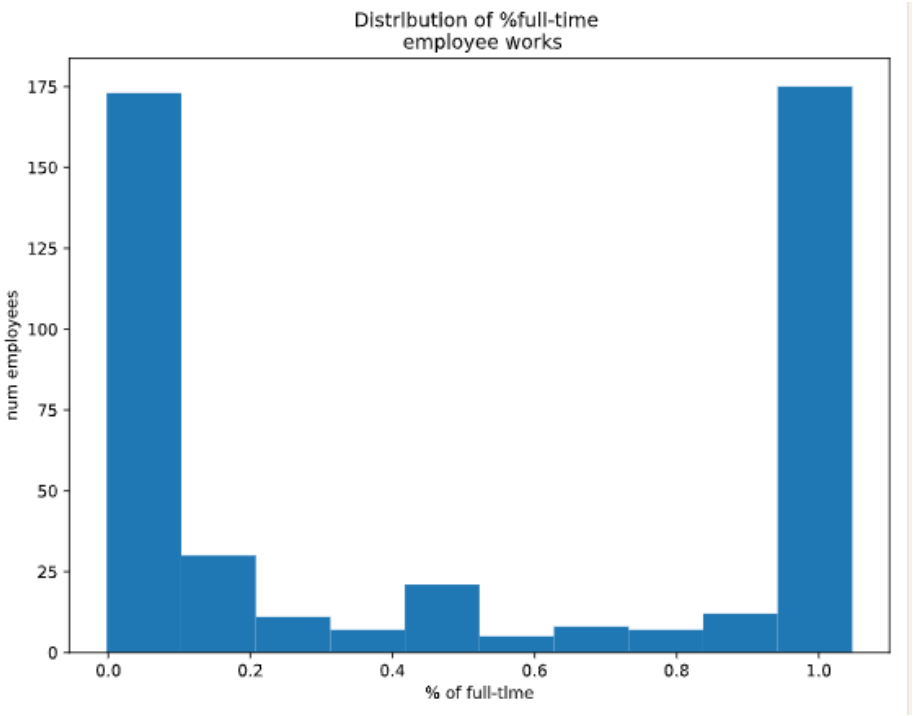
Some columns of the data frame correspond to features (description of the budget items) such as the job_Title_Description that the value in this column tell us if a budget item is for a teacher, custodian, or other employee. Some of the columns are related to the budget item labels which we will try to predict with our model.

	Function	Use	Sharing	Reporting	Student_Type	Position_Type	Object_Type	Pre_K	Operating_Status	Object_Description	...
109283	Professional Development	ISPD	Shared Services	Non-School	Unspecified	Instructional Coach	Other Compensation/Stipend	NO_LABEL	PreK-12 Operating	WORKSHOP PARTICIPANT	...
102430	Substitute Compensation	Instruction	School Reported	School	Unspecified	Substitute	Base Salary/Compensation	NO_LABEL	PreK-12 Operating	SALARIES OF PART TIME EMPLOYEE	...
413949	Parent & Community Relations	NO_LABEL	School Reported	School	NO_LABEL	Other	NO_LABEL	NO_LABEL	PreK-12 Operating	NaN	...
433672	Library & Media	Instruction	School on Central Budgets	Non-School	Unspecified	Librarian	Benefits	NO_LABEL	PreK-12 Operating	EMPLOYEE BENEFITS	...
415831	Substitute Compensation	Instruction	School Reported	School	Poverty	Substitute	Substitute Compensation	Non PreK	PreK-12 Operating	Salaries And Wages For Substitute Professionals	...

The information of dataset is shown in table below. As we can see the data has both numeric and text data. Based on this table, we have 23 objects and 2 float types in the data set. Since the objects are not efficient data type in pandas, we convert the objects to category type and, also, we need to encode the categorical data to numeric ones using the available options in scikit learn such as get_dummies or hot_encoding.

```
Data columns (total 25 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   Function                400277 non-null  object
 1   Use                     400277 non-null  object
 2   Sharing                 400277 non-null  object
 3   Reporting               400277 non-null  object
 4   Student_Type            400277 non-null  object
 5   Position_Type           400277 non-null  object
 6   Object_Type             400277 non-null  object
 7   Pre_K                   400277 non-null  object
 8   Operating_Status        400277 non-null  object
 9   Object_Description      375493 non-null  object
 10  Text_2                  88217 non-null   object
 11  SubFund_Description     306855 non-null  object
 12  Job_Title_Description   292743 non-null  object
 13  Text_3                  109152 non-null  object
 14  Text_4                  53746 non-null   object
 15  Sub_Object_Description  91603 non-null   object
 16  Location_Description    162054 non-null  object
 17  FTE                     126071 non-null  float64
 18  Function_Description    342195 non-null  object
 19  Facility_or_Department  53886 non-null   object
 20  Position_Extra          264764 non-null  object
 21  Total                   395722 non-null  float64
 22  Program_Description     304660 non-null  object
 23  Fund_Description        202877 non-null  object
 24  Text_1                  292285 non-null  object
dtypes: float64(2), object(23)
memory usage: 79.4+ MB
```

There are two numeric variables in the data set FTE which stands for Full-time equivalent and Total that stands for total cost of the expenditure. If the budget item is related to an employee, this variable shows us the percentage of full-time that the employee works. For example, a value of 1 tells that an employee works full-time.



Methodology

In this study, as the label of the samples are known and are used to train the model, we are dealing with a supervised classification model. Regarding the two sets of labels (main classes and the sub classes) the main work of this project is a multi-class-multi-label classification problem with the objective of assigning official classes to the unstructured text in budget line items with related probabilities. Most of the research conducted on classification in data mining has been devoted to single label problems. While single label classification, which assigns each rule in the classifier the most obvious label, has been widely studied, little work has been done on multi-label classification.

A general classification problem can be considered as follows: let D denote the range of possible training observation and Y be a list of class labels, let H denote the set of classifiers for $D \rightarrow Y$, each sample $d \in D$ is assigned a single class y that belongs to Y. The goal is to find a classifier $h \in H$ that maximizes the probability that $h(d)=y$ for each test case (d,y). In multi -label problems, however, each instance $d \in D$ can be assigned multiple labels $y1,y2,\ldots,y_k$ for $yj \in y$, and is represented as a pair $(d,(y1,y2,\ldots,y_k))$ where $(y1s,y2,\ldots,y_k$ is a list of ranked class labels from y attached to the observation d in the training data.

Different investigations in each step of the modeling such as different solvers for a specific classifier will be applied to compare different model efficiencies. The approach of modeling the

problem is starting by a naïve and basic model which only works with the numeric features, and gradually, by adding more details and include the text data as well, try to improve the performance of the model to predict the class of each item. Along the way we will use Natural language Processing (NLP), and efficiency boosting hashing tricks. After creating the robust pipeline, some more functionalities such as removing the stop words and stemming will be applied to the model.

Data Splitting and Model Selection

For imbalance classification we cannot use traditional split-test method in scikit learn library and we may end up with labels in test set that never show up in training set.
We have other approaches such as StratifiedShuffleSplit which works for single target variable but in this case study there are multiple target variables. So, we use a multi-label-multi-class function to split data to train and test sets proportional to observations.

Model Selection

Different classification models were used to evaluate the performance of each one. Multi-class logistic regression, which considers the columns independent of each other, is a viable choice to start training the model with and perform different analysis on it. Having said that, since the classifier should be trained on each label separately, we need to use the most used strategy for multiclass classification known as One-Vs-Rest Classifier or one-vs-all. Based on the scikit-learn documentation, this strategy consists of fitting one classifier per class against all the other classes.

In this method, since each class is represented by one and one classifier only, it is possible to gain knowledge about the class by inspecting its corresponding classifier.

Natural Language Processing

The first step in processing the text data is called "tokenization". Tokenization is the process of splitting a long string into segments. Usually, this means taking a string and splitting it into a list of strings where we have one string for each word. If we want to tokenize on whitespace, that is split into words every time there is a space, tab, or return in the text. For some datasets, we may want to split words based on other characters than whitespace. For example, in this dataset we may observe that often we have words that are combined with a hyphen, like we can opt in this case to tokenize on whitespace and punctuation. Here we break into tokens every time we see a space or any mark of punctuation.

When we have the tokens, we want to use them as part of our machine learning algorithm. Often, the first way to do this is to simply count the number of times that a particular token appears in a row. This is called a "bag of words" representation, because you can imagine our vocabulary as a bag of all our words, and we just count the number of times a particular word was pulled out of that bag.

The bag of words is an effortless way to represent text in numeric form to be prepared for machine learning models. The Bag of words discards information about grammar and word order to count the frequency of occurrences.

Scikit-learn tool for bag of words is called the count vectorizer. In order to create a bag of words representing all the text data, we should first convert the text data in each row into a single string.

The Count vectorizer works by taking an array of strings and doing 3 functions:

- Tokenizes all the strings
- Builds a vocabulary: It creates notes of all the words that exist in the string which is called vocabulary.
- Counts the occurrences of each token in the vocabulary

Count vectorizer accepts each row in a single string format, so we need to turn the list of strings into a single string. After combining the text columns into one string and passing the results to the vectorizer, it turns out that in total there are 19956 tokens in the dataset of which 15291 of these tokens are alpha-numeric tokens. We only need to work with alpha-numeric tokens.

Another useful concept in NLP is n-grams. An n-gram is a contiguous sequence of n items (words, tokens, symbols...) from a given sample of document. By defining the range of n-grams in a text processing problem, the order of words which appear together in a sequence will be considered. Usually by applying the n-grams rather than monogram the efficiency of the classification model improves.

As will be explained in next sections, to speed up the text vectorizer, we used the hashing vectorizer which implements a hashing trick.

Success Measurement

There are no standard evaluation techniques applicable to the multi-label classification problems (F.A. Thabtah, et.al.). Moreover, the right measure is often problematic and depends heavily on the features of the conducted problem. Also, when we have an imbalanced classification problem, accuracy can be misleading. In this case the metric we use to evaluate the model is multi class log loss. Log loss is what is called a "loss function," and is a probabilistic measure of accuracy in terms of error.

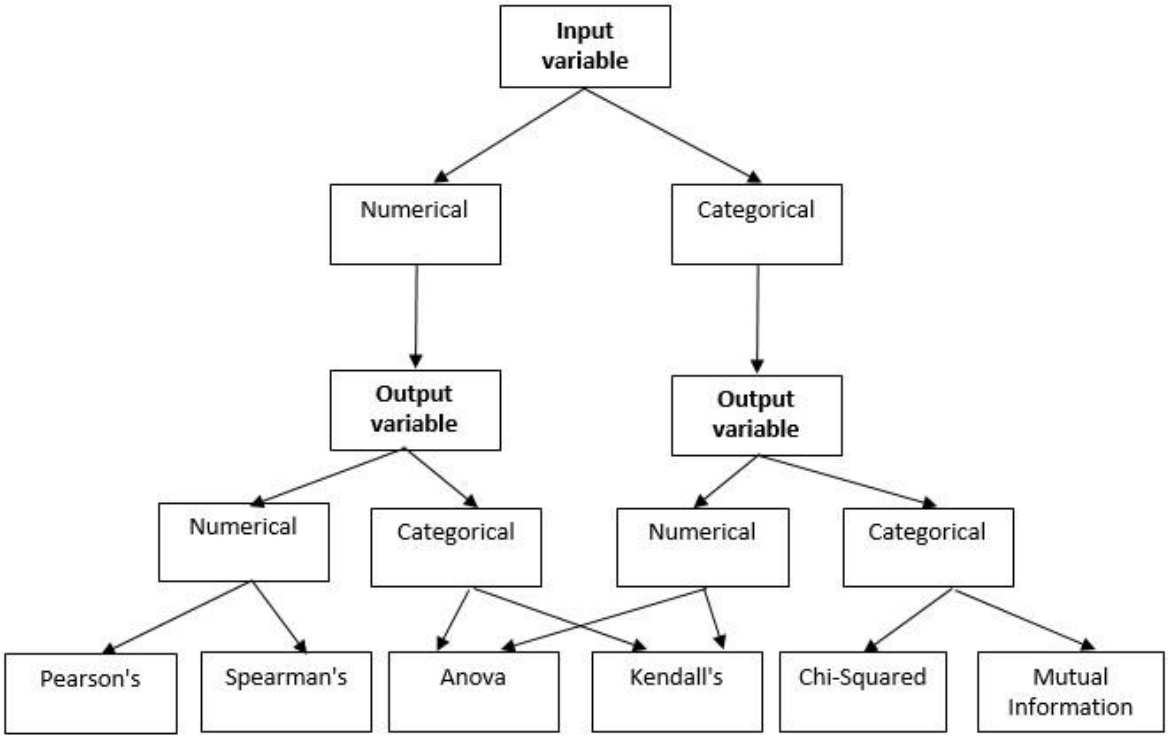
Log loss penalizes both types of errors, but especially those wrong predictions that are highly confident much more than any other type; therefore, this is a good metric to use on our model. The log loss function formula is as follows:

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \cdot \log(p(y_i)) + (1 - y_i) \cdot \log(1 - p(y_i))$$

For our evaluation, we used the multi class version of Logarithmic Loss metric as implemented on DrivenData.org website. Since log loss is an error function, the minimum log loss is desired.

Feature Selection

The following diagram is good guidance to select a feature selection method based on the data type (copyright@MachineLearningMastery.com). Regarding this framework the best feature selection approach for our model would be Chi-squared k best selection method. We should apply this method only to text features of data set.



By adding chi-squared feature selection method, features are selected according to the k highest scores. In our study, the feature selection was working well with models without considering ngrams in text preprocessing but after using bigrams, the model performance decreases (log loss score increases). By increasing the number of features (k) we could get a better result and by reducing the number of features, the execution time improved. So, we need to tradeoff between time and performance.

Evaluation and Final Results

The modeling step started with considering only numerical data. In the next step by adding the text data to the model, the dimension of the problem increases drastically and with 400,277 rows it will be too many dimensions to work with locally while developing the model. So, we sample down data set to 10000 rows so it would be easy and quicker to run the model and do further analysis. The sampling method is a multi-label approach as well.

In the next step, the text features were preprocessed using the bag of words method which was mentioned before. After integrating 2 types of data, the model was trained using the complete features set. By having only numerical features (no data sampling), the log loss would be 1.95 which this amount for the model with all the feature vectors, reduces to 1.27. This result was expected as the textual data provides extra information for the classification task. The text data in this step was not preprocessed and clean, as it was used to get a general view of the model structure.

Improving the model's performance

Text processing, statistical methods and computational efficiency are the possible options of the model performance improvement. The following list is some of the options in this regard:

- **NLP:**
  - **Hashing trick:** In order to preprocess the text features and vectorizing the text features, the count vectorizer for a small sample of data works well. But by applying this method to the big data set, the execution time increases exponentially. One of the interesting findings in this case study is that for vectorizing the text features of big data set, the hashing trick which is a mathematical trick reduces the time and memory capacity of the model training significantly.
    - Based on scikit learn documentation hashing text vectorizer works by applying a hash function to the features and using their hash values as indices directly, rather than looking the indices up in an associative array.
    - This strategy has several advantages such as being very low memory scalable to large datasets as there is no need to store a vocabulary dictionary in memory. Also, it is fast to pickle and un-pickle as it holds no state besides the constructor parameters. On the other hand, it can be used in a streaming (partial fit) or parallel pipeline as there is no state computed during fit.
    - While training the model by the full dataset is so important in this project, making improvements to the model in order to reduce time and memory usage is one of the main concerns. Also in the text preprocessing part, by adding ngrams range to the vectorizer, we could improve the model performance.

  - **stop-word removal:** Stop words are words like “the”, “with”, “her”, which are presumed to be uninformative in representing the content of a text, and which may be removed to avoid them being construed as signal for prediction. In this project the built-in ‘english’ stop-words list in scikit library has been considered.

- **Numeric Preprocessing:** Try different Imputation strategies such as median instead of mean. By changing the imputer strategy from mean to median, as is shown in the table, performance will increase.

- **Model:** Try different models such as Logistic Regression, Stochastic Gradient Descent classifier, Random Forest and K-Nearest Neighbors.

The resulting scores after adding each option to each model are shown in table below. Regarding this table, the best score is for logistic regression with median imputer strategy and number of

features of 2000. We should mention that with small number of samples these large loss score were expected. Now we have an overall view of different algorithms performance and can train the model with the full data set.

Classification Scores	Stop words	Ngrams:(1,2) (1,3)	Median Imputer	K =1000 K = 2000
Logistic Regression	2.3	2.3 2.4	2	2.3 2
SGD Classifier	2.6	2.6 2.8	2.6	3.05 2.6
Random Forest	2.8	2.4 2.6	3.91	2.8 2.4
KNN	3.7	3.7 4.09	3.9	4.48 3.7

- **Optimization:** I haven't come up with a solution for this part yet. The grid search folding is based on stratification. What stratification means for multi-label data and how it can be accomplished has not been addressed in the literature (Vens, C., et.al). So, the solver selection and parameter tunning are done manually.

Conclusion

The classification model which is used in this project is multi-class-multi-label classification to find the class of each line item of school's budgets. The challenge of this study is multi label and multi class nature of the problem with text data type for a big data set which takes so much research to find the best methods and tricks to deal with each kind of data processing and make the model more efficient.

The model training with sample data set is smooth and without any issue but by adding more grams to tokenizer and using the whole dataset, the execution time increases exponentially, and some tricks should be applied to the model. One of these tricks is hashing trick which is included in hashing vectorizer method. Also, we need to reduce the dimensions of the problem by dimensionality reduction methods.

For future work, we can try one method of processing text data called embedded bag of words which doesn't need very clean text data. Another unsolved problem in this project is hyperparameter tunning to avoid over fitting or under fitting so finding a way to do the hyperparameter tunning over the model pipeline will help to achieve a more accurate result.

Reference

- Scikitlearn documentation
- DrivanData ((https://www.drivendata.org)
- https://www.machinelearningmastery.com
- https://www.wikipedia.org
- Course lectures (spam filtering)
- Vens, C., Struyf, J., Schietgat, L., Dˇzeroski, S., Blockeel, H.: Decision trees for hierarchical multi-label classification. Machine Learning 73(2), 185–214 (2008)
- F. A. Thabtah, P. Cowling and Yonghong Peng, "MMAC: a new multi-class, multi-label associative classification approach," Fourth IEEE International Conference on Data Mining (ICDM'04), 2004, pp. 217-224, doi: 10.1109/ICDM.2004.10117.