Milestone II Project Report: Predicting Movie Genres

Simeng Cai Xuanchen Liu

Part A. Supervised Learning

I. Motivation and Abstraction

Movies are one of the most popular forms of entertainment and make us learn something. Both of our team members are film enthusiasts and are interested in using Natural Language Processing to predict movie genres based on the plot summary of the movies listed in the Full Movies Dataset. Each movie in the dataset can belong to multiple genres concurrently, therefore we used multi-label classification models for our project. The dataset was split into 80%-20%: 80% training set and 20% testing set. We converted each movie plot into TF-IDF features and used the MultiLabelBinarizer class from sklearn to transform the labels into a binary format. We then used problem adaptation methods such as Decision Tree Classifier, KNN Classifier and Random Forest Classifier that inherently handle multi-label classifications. We also used problem transformation methods one-vs.-rest (OvR) that divides the multi-label problem into multiple single-label problems. By applying evaluation metrics of f1_mirco, f1_macro, precision, recall, hamming loss, and AUC, we selected the best model. OneVsRestClassifier&Random Forest ended up to be the model with the best performance for the movie genre multi-label classification problem.

II.Data Source

Wen used the movies_metadata.csv from the Movies Dataset from Kaggle (https://www.kaggle.com/rounakbanik/the-movies-dataset?select=ratings.csv) and the datasets retrieved from the TMDB website(https://developers.themoviedb.org/3/people/get-person-details). The original dataset contains 24 columns such as budget, original language, popularity, etc. The important features that we will use from this dataset for supervised learning will be genres, id, imbd_id, and overview. The genres column contains a list of dictionaries with genre's IDs and genre names. 'id' and 'imbd_id' columns contain unique ids (integers) for each movie, and 'overview' has a datatype of object which contains plot summary of each movie. We used all of the 45466 records from the dataset.

III. Methods & Evaluation

1.Data Cleaning and Pre-processing: We dropped rows with invalid values or empty cells. We unpacked the genres column from the dictionary and created a new column 'newgenres' that contains a list of genres that each movie belongs to. For example, a movie with id 862 has a list of three genres [Animation,Comedy,Family]. We have 43024 records after data cleaning. There are 20 unique genres in the cleaned dataset as shown in Figure 1. Figure 2 shows the number of genres per movie.

We then processed the overview column with the following steps:

- Remove all punctuation and special characters, single characters
- Separate the individual words in each overview into tokens
- Lemmatize the text

- Lowercase tokens
- Remove all stop words

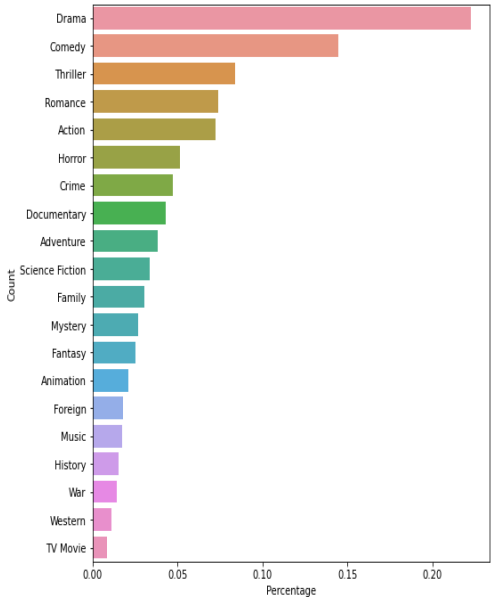


Figure1. Distribution of Movie Genres

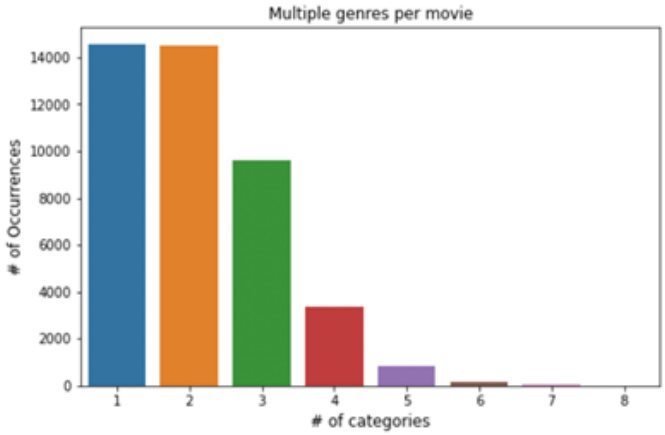


Figure 2. Number of Genres per Movie

	Step	**Method**	**Library/Module**
1	Data Cleaning and preprocessing	Invalid values and empty rows are removed from the dataset.Genres are unpacked from dictionary and processed into a list of genres for each movie	Nltk, pandas
2	Label Transformation	Binarization of labels	MultiLabelBinarizer
3	Feature Transformation	split dataset into training and validation set, frequency-inverse document frequency (TF-IDF) transformation on each movie overview	TfidfVectorizer, train_test_split (sklearn)
4	Model Building	1)Problem-Adaptation method, 2)Problem-Transformation method: One-vs-rest	OneVsRestClassifier, DecisionTreeClassifier, KNeighborsClassifier, RandomForestClassifier
5	Evaluation and model selection	precision, recall, F1-score, accuracy,hamming loss, classification report, AUC	Metrics (sklearn)

2.Label Transformation: This is a multi-label classification so we one hot encoded our labels/genres.By one hot encoding we represent the genres as binary vectors. For example, for the movie with id 862 which has genres [Animation,Comedy,Family], after applying sklearn's MultilabelBinarizer(), each genre is mapped into a vector with values of either 0 or 1. Movie with id 862 then has a label of array([0, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]). All values are 0 except the indexes associated for the three genres [Animation,Comedy,Family] in its label, which have values of 1.

3.Feature Transformation: In this step, we extracted features from the cleaned movie overview data which came from the data cleaning pre-processing step (step 1). We split our data into 80% and 20% using train_test_split: 80%in the training set and 20% in the testing set. We split data before TF-IDF vectorization and transformation instead of on the entire dataset to avoid data leakage. We chose TF-IDF because TF-IDF measures how important a particular word is with respect to a movie overview and all of the overviews of the dataset. Words that are unique to a single movie will have a higher score in the TF-IDF vector. Therefore TF-IDF is a better method compared to bag of words because bag of words just creates a set of vectors containing word occurrences while TF-IDF takes the importance of words into consideration.We tried various combinations of unigrams, bigrams and trigrams, and unigrams and bigrams gave the highest f1 and accuracy scores. So we used the 10,000 most frequent unigrams and bigrams in the data as our features.

4.Model Building:

1) Problem adaptation method: We built models using Decision Tree, KNN Classifier, and Random Forest Classifier that inherently handle multi-label classification problems. These methods address the multi-label classification problem in its full form rather than convert the problem to simpler problems.

2) Problem Transformation method: We used sk-learn's OneVsRestClassifier class together with Decision Tree, KNN Classifier and Random Forest Classifier to solve the multi-label problem by decomposing into multiple independent binary classification problems and then handled by single-class classifiers.

5.Evaluation metrics: movie genre prediction is a multi-label classification problem, which means predictions can be fully correct, partially correct or fully incorrect. A lot of metrics we use for multi-class, such as accuracy, do not readily translate to multi-label as they fail to capture the partially correct predictions. To evaluate the results, in this case we chose to use several classification metrics: accuracy, hamming loss, global micro and macro F1-score, f1 score by label/genre and area under ROC-curve by label/genre..

1)Accuracy Score: In a multi-label classification problem, only if the entire list of predicted genres for a movie overview strictly matches with the true list of genres, then the accuracy is 1; otherwise it is 0. For example, for a movie with a true list of genres [horror, romantic, history],only if all three genres are predicted, it is considered as accurately predicted. Because partially correct labels are not captured and also because we have imbalanced data, we will use other metrics together with accuracy to select the best model. Random Forest, Decision Tree and OneVsRestClassifier&Random Forest models have significantly higher accuracy scores than other models (highlighted yellow).

Model	Accuracy_score	F1_macro	F1_micro	Hamming_loss
Random Forest	0.16	0.13	0.36	0.09
Decision Tree	0.12	0.25	0.37	0.12
KNN	0.08	0.04	0.22	0.11

OneVsRestClassifier&KNN	0.08	0.04	0.22	0.11
OneVsRestClassifier& Decision Tree	0.08	0.31	0.41	0.12
OneVsRestClassifier&Random Forest	0.16	0.20	0.39	0.09

2)Hamming loss: Hamming loss is the fraction of labels that are incorrectly predicted. Hamming loss is more forgiving than accuracy as it penalizes only the individual labels.The lower the Hamming loss, the better the performance of the model. All of these models have very low Hamming loss values, while Random Forest and OneVsRestClassifier&Random Forest have the lowest hamming loss of 0.09 (highlighted yellow).

3) F1-micro and F1-macro: F1 score is a good measure for our model performance because we want to seek a balance between Precision and Recall for our imbalanced data. We compared both F1-micro and F1-macro scores across the models, and OneVsRestClassifier& Decision Tree, OneVsRestClassifier&Random Forest, and Decision Tree have significantly higher f1_macro and f1_micro scores than the rest of the models (highlighted yellow).

4) Precision, Recall, and F1 by label: We also went through the classification_reports for each model with the precision, recall, and F1 scores for each genre.OneVsRestClassifier& Decision Tree have the highest F1 scores by label for almost all genres compared to other models, and Decision Tree has the second highest F1 scores by label. (An example of a classification report will be shown in the failure analysis section).

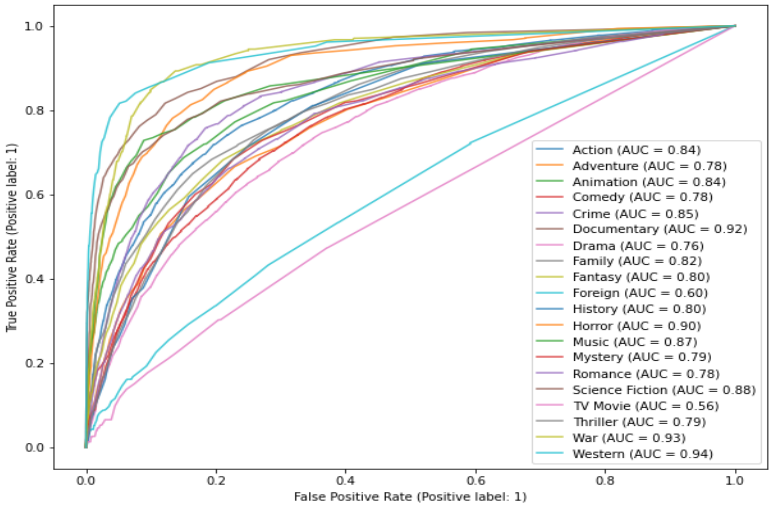


Figure 3: ROC for OneVsRestClassifier&Random Forest model

5) Area under ROC curve by label:Random Forest and OneVsRestClassifier&Random Forest have the highest AUC values by label.The higher the AUC value , the better the performance of the model at distinguishing between the positive and negative classes for each label.

After all of these evaluation metrics, we decided to choose the model with the best performance which is OneVsRestClassifier&Random Forest. It has the highest accuracy score, lowest hamming loss, second best F1 micro and macro scores, and highest AUC values.

VI.Failure analysis

Though the accuracy for our selected model-- OneVsRestClassifier&Random Forest seems low, which is only 16%. Note that accuracy score in a multi-label classification problem only captures completely correct cases and ignores the partially correct cases. The low hamming loss of 9% indicates that there are few partially incorrect labels. OneVsRestClassifier&Random Forest model has a low False Positive error (type I) and a higher False Negative error (type II). A type I error in this problem is that the model result says a movie belongs to the genre, but it actually doesn't. And a type II error in this case is that the test result says a movie doesn't belong to this genre , but it actually does.

We used multilabel_confusion_matrix from sklearn.metrics to create a normalized and an unnormalized confusion matrix for each movie genre. As we can see from the classification report, OneVsRestClassifier&Random Forest model has very high Precision scores by label and much lower Recall scores. When we increase the precision of a classifier, the downside is that we reduce recall. For example, the confusion matrix for genre Action has a low type I error of 0.0046 and much higher type II error of 0.13. This means that the model classifies 0.46% of the movies that don't belong to Action as Action movies, and the model classifies 13% of actual Action movies as non-Action movies. The precision score for Action is 0.79 and recall is 0.12 which is significantly lower. Unfortunately, we can't have both precision and recall high. If we increase precision, it will reduce recall and vice versa. This is the trade off between precision and recall for our model.

	precision	recall	f1-score	s
Action	0.79	0.12	0.20	
Adventure	0.71	0.02	0.03	
Animation	0.64	0.16	0.26	
Comedy	0.71	0.28	0.40	
Crime	0.49	0.04	0.07	
Documentary	0.92	0.39	0.54	
Drama	0.66	0.72	0.69	
Family	0.65	0.04	0.07	
Fantasy	0.75	0.02	0.04	
Foreign	0.28	0.02	0.03	
History	0.00	0.00	0.00	
Horror	0.76	0.22	0.34	
Music	0.68	0.07	0.13	
Mystery	0.59	0.02	0.04	
Romance	0.61	0.10	0.17	
Science Fiction	0.78	0.30	0.43	
TV Movie	0.14	0.01	0.01	
Thriller	0.63	0.07	0.13	
War	0.83	0.02	0.04	
Western	0.83	0.18	0.30	
micro avg	0.68	0.27	0.39	
macro avg	0.62	0.14	0.20	
weighted avg	0.67	0.27	0.33	
samples avg	0.50	0.33	0.38	

```
Confusion matrix for label Action:
[[0.84288205 0.00546194]
 [0.1323649  0.01929111]]
Confusion matrix for label Adventure:
[[0.91690877 0.0010459 ]
 [0.08006973 0.0019756 ]]
Confusion matrix for label Animation:
[[0.95258571 0.00348635]
 [0.03660662 0.00732132]]
Confusion matrix for label Comedy:
[[0.65322487 0.03765253]
 [0.22045322 0.08866938]]
Confusion matrix for label Crime:
[[0.89831493 0.00383498]
 [0.0938989  0.00395119]]
Confusion matrix for label Documentary:
[[0.90191749 0.00371877]
 [0.0585706  0.03579314]]
Confusion matrix for label Drama:
[[0.35711795 0.17606043]
 [0.13561883 0.33120279]]
Confusion matrix for label Family:
[[0.93608367 0.00116212]
 [0.06066241 0.00209181]]
Confusion matrix for label Fantasy:
[[9.45613016e-01 3.48634515e-04]
 [5.29924463e-02 1.04590354e-03]]
Confusion matrix for label Foreign:
[[9.61301569e-01 5.81057525e-04]
 [3.77687391e-02 3.48634515e-04]]
```

Figure 4: Classification Report and Confusion Matrix for OneVsRestClassifier&Random Forest

V. Discussion

We solved the multi-label movie genre classification problem with two methods--problem adaptation methods and problem transformation methods, and out of the six models the OneVsRestClassifier&Random Forest model outperformed the rest of them. Multi-label classification problem is completely self explored by our team members and we learned how to solve the problem step by step- from binarization of labels, problem adaptation and transformation algorithm methods, to model selection using multi-label classification evaluation metrics such as hamming loss. We learnt that accuracy for multi-label classification only captures the completely correct cases and misses the partially correct cases. What surprised us is that KNN and OneVsRestClassifier&KNN produces exactly the same output even for metrics by label. However, for Random Forest and Decision tree models, problem adaptation method and transformation method produce completely different results and have different model performance. If we have an opportunity to explore further, we would like to research on what caused the similarity and differences between adaptation and transformation methods as well as apply other multi-label classification methods such as ensemble approaches.

One ethical concern would be that if producers would like to have movies published in a country where the movie genre is forbidden. For example in some countries certain horror or violence movies are forbidden from publishing. If this model is used by the government to classify movie genres and the producer knows plot summary is used to make the prediction. They can alter their plot summary and avoid words that will result in predicting the movie as a violence movie. This can cause harm to the audience such as children because some violence or horor content can harm their mental health.

Part B. Unsupervised Learning

I Motivation

The motivation is similar to that of the supervised learning addressed above. The difference is that for the unsupervised learning portion, we particularly would like to explore whether the clustering of movie keywords will align with the 20 unique movie genres. Specifically, we will use unsupervised learning techniques to do data clustering and group different movies based on keywords manually, and finally we will do a comparison between the most popular(common) movie genres that we come up from the unsupervised clustering methods and the existing classification to see whether the result accords with, or to what extent accords with each other.

II Data Source

The datasets applied for the unsupervised part are shared with the supervised portion illustrated above. But other than movies_metadata.csv, we will use the dataset of "keywords.csv" as well.

III Unsupervised Learning Methods

1. Data Preprocessing: two columns "id"(representing movie_id) and "genres"(representing movie genres) coming from the datasets "movies_metadata.csv", and columns "id" and "keywords"(representing

movie keywords) coming from the datasets “keywords.csv” were chosen and preprocessed for further data analysis.

Specifically, missing or malformed data were removed or transformed. For example, “id” columns include multiple “datetime” format data, which should be converted to “numeric” format. Most importantly, we need to extract all movie genres and keywords by applying ast.literal_eval().

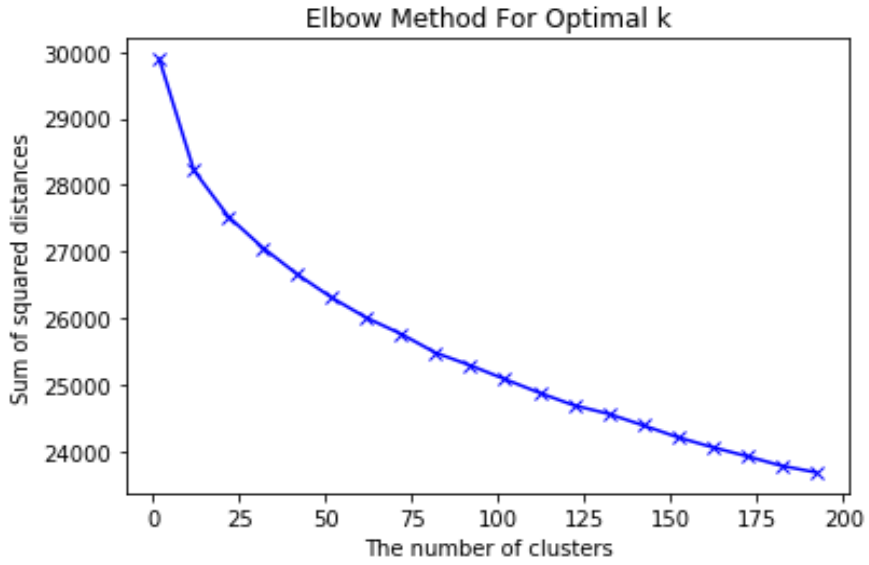
After completing data cleaning and extracting, we merged the two dataframes on “id” and only kept three columns: “id”, “genres”, “keywords”.

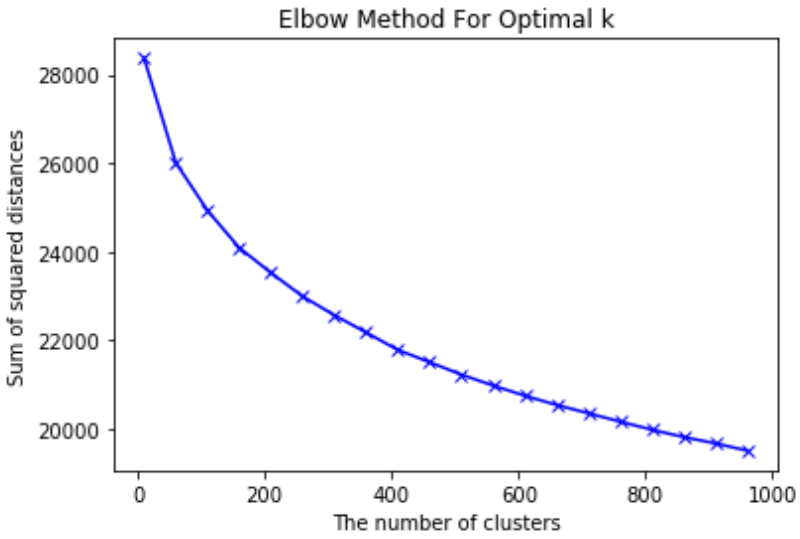
2. Words Vectorization: text vectorization is an essential step for unsupervised learning. In this case, we applied TfidfVectorizer() to the entire keywords column to get it ready for data analysis, and removed all “stopwords”.

3. Algorithm & Validation Techniques Selection: We chose K-means clustering as the main method to find clusters. K-means performs well in finding groups in the data with the number of groups defined by the parameter “k”. In this case, we used K-means clustering to the vectorized “keywords”, the features, and assigned them to different groups based on the similarity.

However, the question is how do we decide upon the parameter “K”? We used a common criteria called “elbow diagram” to find the approximate K.

Specifically, we chose setting the range of K from 2 to 200 with a step of 10, and drew an Elbow graph to observe the optimal K. There was not a clear “elbow point” for the graph. So we adjusted the range of K and set it from 10 to 1000 with a step of 50. Similarly, the elbow point was not quite clear as well. In other words, the elbow graph did not show a kind of sharp shift in the rate of decrease which is normally treated as the desired value of K.





But what we observed from the two “elbow graphs” was that though no clear sharp shift appeared, the decreased rate decreased more rapidly when the value of K was around 25. Combining our original research goal, we decided to manually define the K value as the number of unique movie genres, namely 20.

Furthermore, we assigned each dataframe row, which represents each movie, to a specific group(cluster) by applying the KMeans() method again, and added another new column(feature) called “cluster” in the dataframe. Then we will extract the most common genre in each group (Please note that each group consists of multiple movies with diverse genres) to compare the clustering result of movie genres with the existing classification.

4. Challenges: One challenge or trick we met is the process of keywords vectorization. TfidfVectorizer() applied simply to a string format. But the keywords we obtained after data preprocessing is in a list format, or technically speaking, is a list of lists. This is a trick that will be easily ignored without much attention.

Another challenge is that normally we have no idea about how many clusters exist and whether the K value we set is the optimal one. In this context, though trying the K-means clustering method and tuning the K values to different ranges, we did not find the optimal value based on the evaluation criteria and the clusters were shown without clear boundaries. In addition, we also tried applying the Principal Component Analysis method but we cannot observe the well-defined clusterings as well.

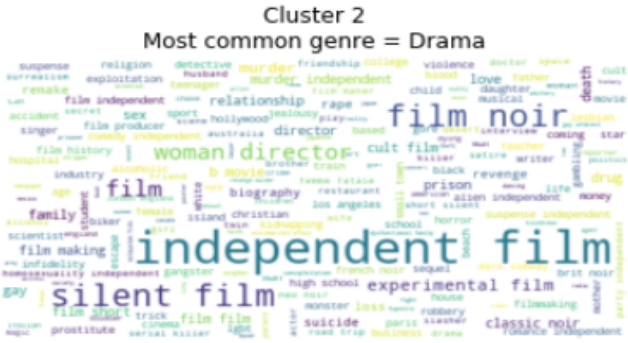
The challenges demonstrated above is also another reason why we finally decided to set the K value to be the same with the number of unique movie genres to compare their results.

IV. Unsupervised Evaluation

What we found interesting is that the result of grouping different movies based on keywords is NOT quite aligned with the existing unique 20 movie genres. Ideally, when we manually set K value to 20, we would like to see the most common genre in each movie cluster can cover all the 20 or most of the 20 movie

genres, such as “Comedy”, “Romance”, “Family”, “Music”, “Drama”, “Action”, “Adventure”, etc. However, the result showed that the overwhelming majority of the most common movie genres is “Drama”(A total of 14).

It seems like everyone loves “Drama”! By generating word clouds, we can prove this data analysis result as well. The words that appear more frequently are more aligned with the movie genre of “Drama”.



V. Discussion

The process of applying unsupervised learning methods for data exploration and data analysis leaves the data itself to find structure in its inputs, since no labels are given. Sometimes, we can discover hidden patterns in data but sometimes we cannot. In this case, we can find clusters based on predefined parameters, but we cannot conclude that this is the optimal choice since the clusterings are not well separated from each other. In other words, clusterings are mixed with each other without clear boundaries.

In addition, if comparing the most common movie genre in each clustering with the existing movie genres, the result actually does not reflect a good sign of well-performed groupings. So to a great extent, we will say that unsupervised learning can be a goal in itself.

In terms of ethical issues, an obvious one is that it is more difficult to recognize whether a specific movie is categorized correctly. For example, some movies were mistakenly assigned to some banned movie genres based on some keywords, which may negatively affect its reputation and revenue performance. In addition, even though we know problems of bias exist, it will be difficult to recognize and address them.

VI. Statement of Work

Both team members worked on the unsupervised and supervised learning portions and drafted the final report. We cooperated on model selection, evaluation metrics and parameter tuning. Xuanchen's was more focused on the unsupervised portion and Simeng was more focused on the supervised portion. During the entire process, we kept exchanging our progress and difficulties we encountered.