Anirudh Rao
BE21B004

DA5401 – Data Analytics Lab

Data Challenge

Report

Dataset

The dataset consists of 198982 embeddings, each of dimension 1024. These are assigned to one or more ICD codes. The goal is to predict the ICD codes associated with an embedding. There are 1400 possible ICD codes in this dataset.

The ICD codes have the format alphabet-number-dot-number. For example, H35.82 is a valid ICD code. There are 24 possible alphabets with which an ICD code in the dataset can start. 'P' and 'U' are not part of this list of alphabets.

Exploratory Data Analysis

To understand the distribution of the ICD codes, the number of occurrences of an ICD code starting with a particular alphabet was found.

Alphabet	Occurrences
K	42711
J	38599
H	37743
Z	35455
N	32082
D	17044
M	16807
R	12431
G	10730
L	9866
C	7011
E	4758
T	4469
S	3230
Q	2940
F	2286
O	1847
I	1058

B	514
A	167
X	29
W	17
V	5
Y	5

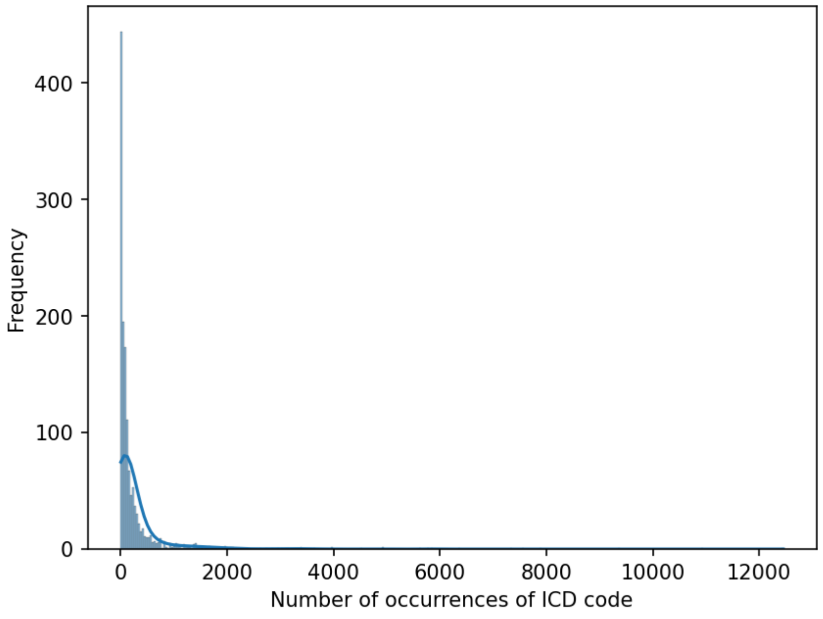
We can see that codes starting with 'K', 'J', 'H', 'Z', and 'N' are overrepresented in the dataset. Codes starting with 'X', 'W', 'V', and 'Y' occur fewer than 100 times.

The top 10 most and least frequently occurring ICD codes were found.

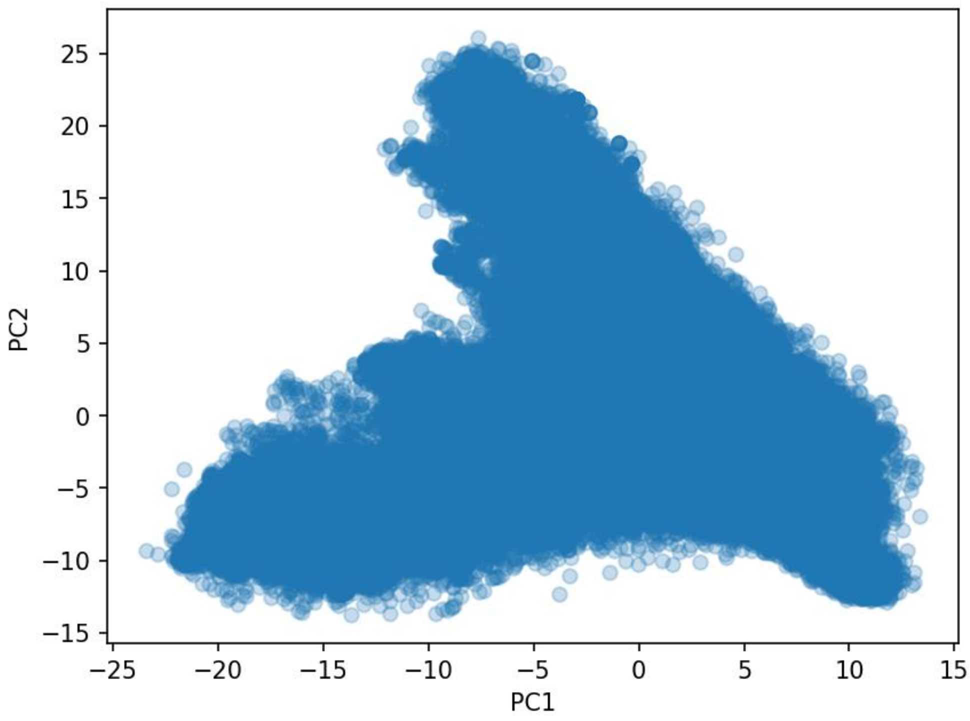
Code	Occurrences
Z12.11	12452
J34.3	10928
J34.2	9349
J35.3	7901
J34.89	7565
K57.30	7342
J35.2	6337
K63.5	5849
J35.01	5758
H66.93	5626

Code	Occurrences
M53.88	1
Z96.7	1
M23.342	1
M24.021	1
M93.271	1
M23.341	1
M24.272	1
M94.252	1
M24.511	1
M93.861	1

On average, an ICD code occurred 300 times in the dataset. 24 ICD codes occur only once in the dataset. 770 codes occur fewer than 100 times in the dataset. Thus, the data has a high imbalance.

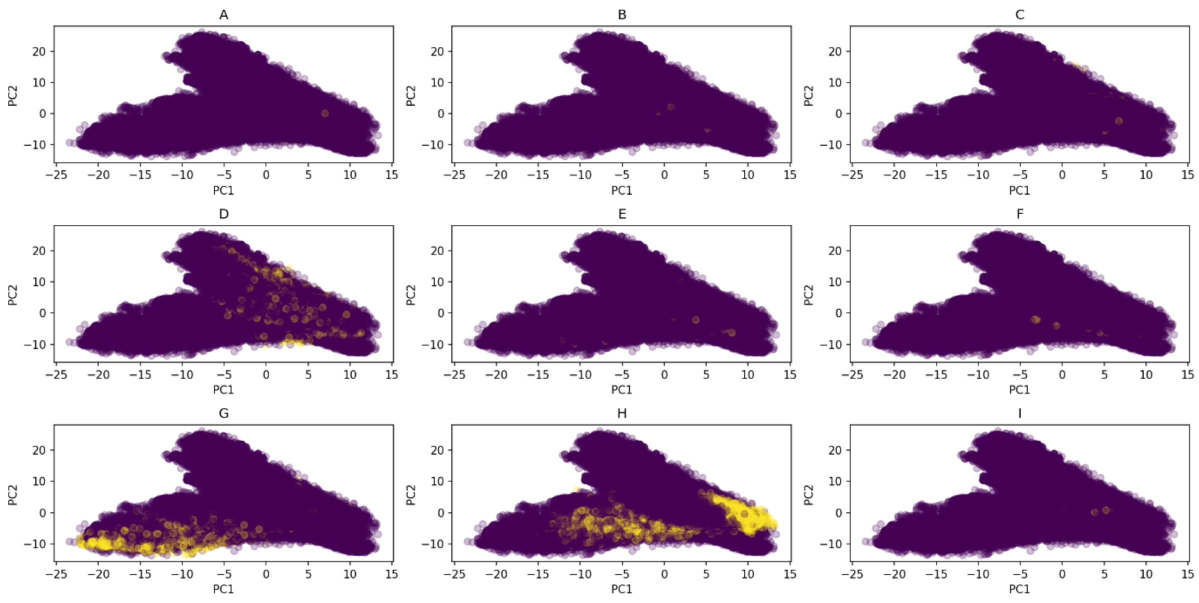


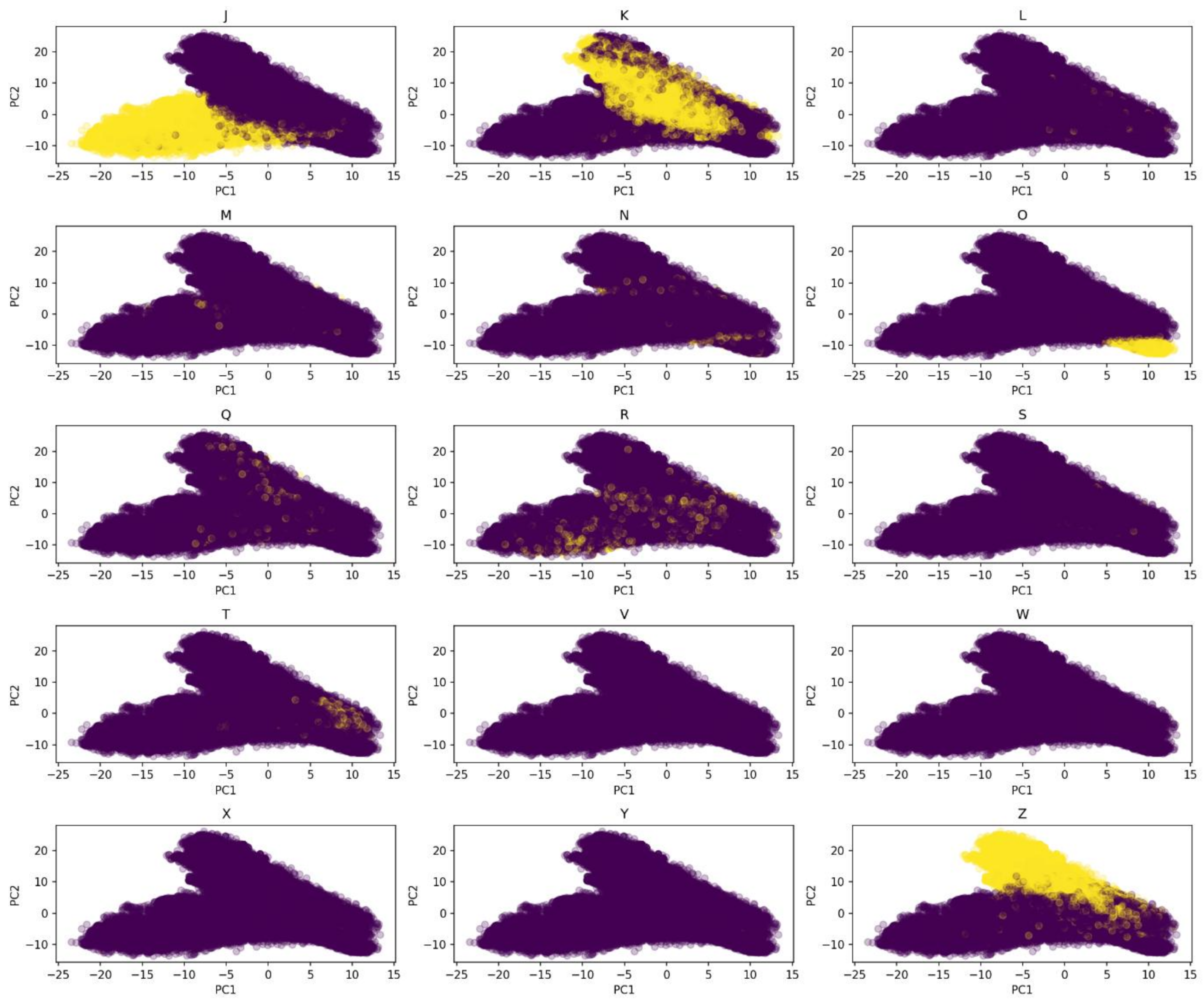
To visualize the data in a simple manner, PCA can be performed on the embeddings after mean centering. A scatterplot of the first two principal components are shown below.



From this plot, there does not appear to be any obvious clustering of the embeddings. PC1 explains only 0.04% of the total variance.

To better understand the distribution of the data, we highlight those points associated with ICD codes starting with the same alphabet.





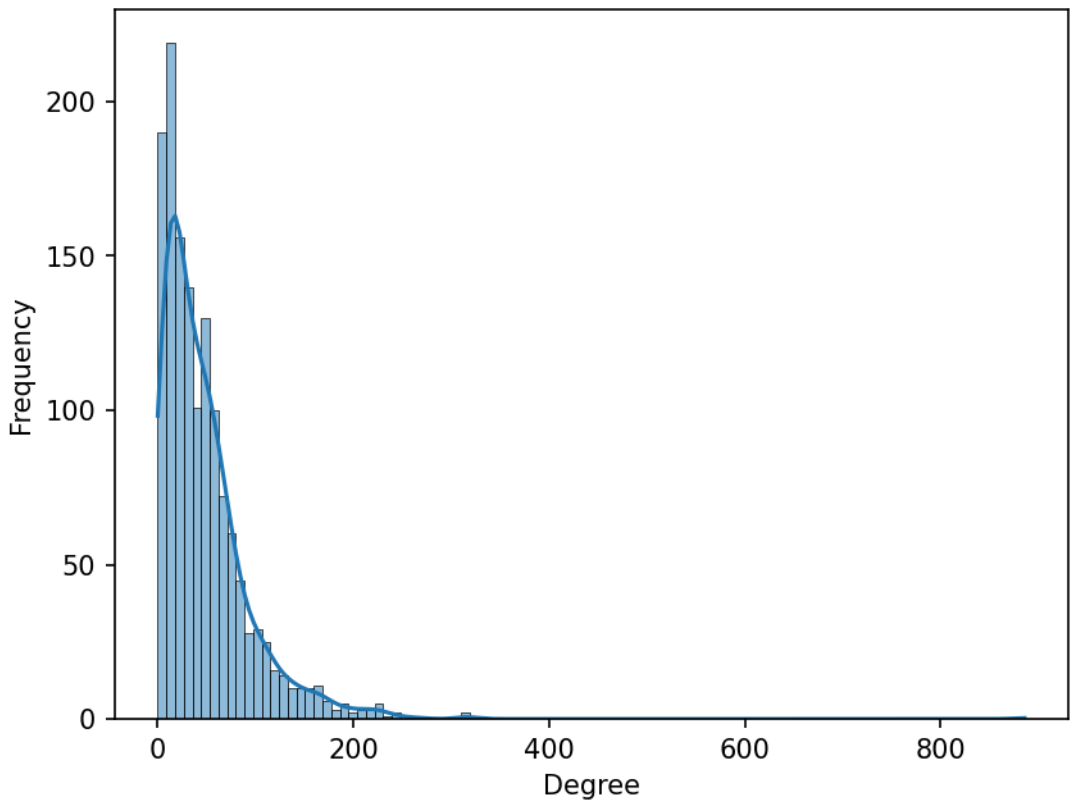
We can see that embeddings associated with ICD codes starting with ‘J’, ‘K’, ‘O’, and ‘Z’ tend to cluster together. Running non-linear dimensionality reduction on the embeddings proved to be computationally infeasible.

Next, the goal was to see which ICD codes frequently occur together. For this, a co-occurrence network was constructed using Python’s `networkx` library.

This network has 1400 nodes and 33434 edges. On average, a node was connected to 48 other nodes. The following ICD codes had the highest degrees:

Code	Degree
Z98.890	886
G89.29	318
G89.18	314
L90.5	266
K57.30	248

The degree distribution of the network is shown below:

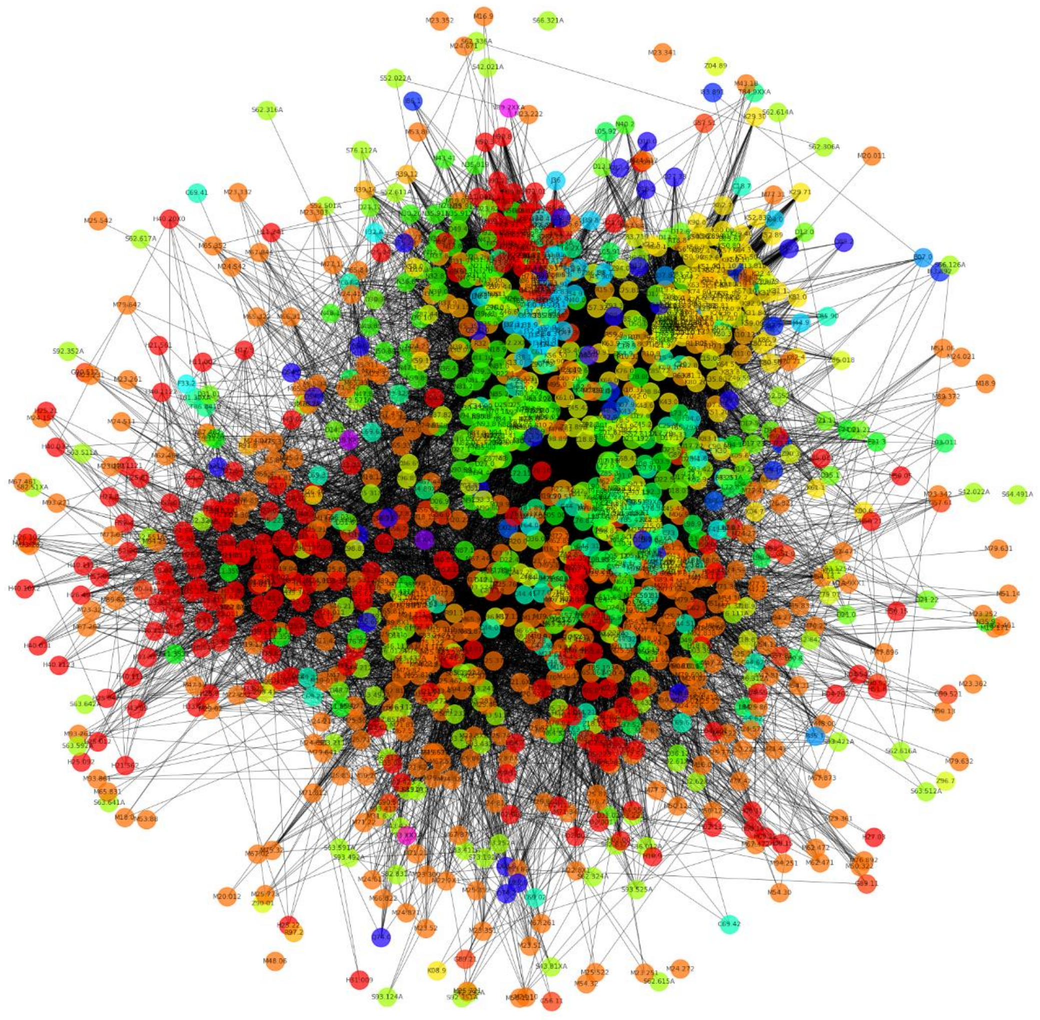


11 ICD codes did not occur with any other ICD code. They are S66.321A, M23.362, M19.171, Z04.89, M24.272, M23.352, N35.9, R97.2, S64.491A, M23.341, and M48.06.

The top 5 pairs of codes that occurred most frequently are tabulated below.

Code 1	Code 2	Number of co-occurrences
J34.3	J34.2	15986
K57.30	Z12.11	10110
J34.3	J34.89	9734
J34.2	J34.89	8766
Z12.11	K63.5	8258

The network was then visualized, assigning nodes based on their alphabet group.



There appears to be some clustering but it is not very clear.

With some understanding of the data, we can proceed to build a multilabel classifier to predict ICD codes from embeddings.

Model Selection

The features were defined as the embeddings. The labels were defined as a 1400-dimensional binary vector using `sklearn`'s `MultiLabelBinarizer`. A training-validation split was performed with 90% of the data being used for training and rest for validation. The test performance of the final model was evaluated on the test dataset provided as part of this Data Challenge.

As there are multiple labels to be predicted from the input features, `sklearn`'s `MultiOutputClassifier` class was used. This will train 1400 different classifiers (of the same type), one for each of the ICD codes.

The `DecisionTreeClassifier` was first used as the classification algorithm. The `class_weight` parameter was set to `balanced` and the `max_depth` was set to `2` to reduce training time. Micro-averaged F1 score was used as the performance metric. This achieved a training F1 score of 0.33 and a validation F1 score of just 0.11.

Next, `LogisticRegression` was used with the `MultiOutputClassifier`. The `class_weight` parameter was set to `balanced` and `max_iter` was set to `1000`. This achieved a training F1 score of 0.35 and a validation F1 score of 0.13.

From this, it was clear that standard machine learning algorithms are unlikely to perform well on this dataset. Although not covered in any of the courses done so far, deep learning was then used to build a neural network-based classifier. To reduce training time, Colab GPU was used as a hardware accelerator.

The first architecture was a simple feedforward model consisting of an input dense layer of 1024 neurons (for the embeddings) and an output dense layer of 1400 neurons (for the ICD codes). The input activation was set to ReLU while that of the output was set to sigmoid. The Adam optimizer was used and the loss was set to binary cross-entropy. The model was trained for 200 epochs with early stopping which had a patience of 3 epochs. This achieved a training score of 0.55 and a validation score of 0.27. This was much better than the standard ML algorithms.

To improve the current architecture, an additional batch normalization layer was added between the two existing layers. This was able to achieve a training score of 0.7 and a validation score of 0.45, marking an improvement from the previous architecture.

So far, the data was not subjected to any scaling. To see the impact of this, min-max scaling was performed on the embeddings prior to model training. However, the training score was reduced to 0.47 and validation score to 0.25. Thus, scaling was not necessary.

To test whether the inclusion of an additional layer will improve performance, a hidden layer of 512 neurons was added. This achieved a training score of 0.63 and a validation score of 0.39. Thus, additional layers might not improve model performance.

Based on the performances of all the models tested so far, it was decided that a simple model with batch normalization was the best one to use for the given dataset.

Final Performance Metrics

The best model was then used to predict based on the test embeddings provided. The predictions were submitted on Kaggle. This achieved a score of 0.44.

Model	Training Score	Validation Score	Test Score
Decision Tree	0.33	0.11	–
Logistic Regression	0.35	0.13	–
Two-layer NN	0.55	0.27	–
NN + BatchNorm	0.70	0.45	0.44
NN + MinMax	0.47	0.25	–
Three-layer NN	0.63	0.39	–