Report

In this report three questions are answered and discussed:  
1. which members are likely to be admitted into the hospital;

2. what factors contribute to hospital admissions;

3. the members that have the most hospital admissions.

# Question 1

I transform this question into a classification problem, that is, given all the features we have (such as age, gender, scores), I separate customers into two groups: those who have been admitted to hospital at least once, and those who have never been admitted to hospital. So the question boils down to finding an appropriate algorithm that can learn from these two groups. Once new customer data is presented, the trained algorithm should be able to tell if the customer will be admitted to hospital.

Of course the problem can also be phrased differently. For instance, if we would like to know if a customer will have many hospital admissions (for example, >5), or few hospital admissions (<=5) or no admissions at all (=0), 3 groups can be given to the customers. That would give us more granularity in terms of number of admissions.

One challenge for this question is that once the customers are labelled with with/without hospital admissions, I quickly realize that the number of customers without admissions are roughly 6 times more than the number of customers with admissions. This is what we call ‘imbalanced dataset’ in machine learning. Imbalanced dataset could lead to misleading results if not taken care of. For instance, if we use this imbalanced dataset directly and accuracy score is what we care about, an algorithm that basically labels all customer to be without admission could still achieve an accuracy score of 86% (6/7). In other words, the algorithm does not need to care about the minority group because even though the algorithm labels entire minority group wrong, as long as it can label the majority group correctly, the accuracy score could still be satisfactory. This is something we should avoid.

There are multiple ways we can do to mitigate this issue. In general, under-sampling and over-sampling are two techniques that are popular in practice.

* Under-sampling. Sample the majority class to match the size of minority class
* Over-sampling. Sample the minority class to match the size of majority class

While under-sampling is straightforward, sometimes over-sampling requires more mathematically manipulation. If we just repeat minority class until the size matches that of majority class, we will end up with many duplicated data point from minority class, which will not add value for model training. The technique to over-sampling is in fact an active field for machine learning, and there is a Python package called ‘imbalance-learn’ for this problem.

A common technique for over-sampling is to generate pseudo data points from minority class. These pseudo data points tend to cluster with the original minority class to boost the number of minority class. This technique should be used with caution: pseudo points are never real observations, and model trained based it might be biased.

In this report, I do not use the aforementioned over-sampling method. Instead, for each fold of cross-validation, I keep the minority class the same, and only sample the majority class without replacement. Since majority class is only 6 times bigger than the minority class, I only need to use at most 6-fold cross-validation to ensure all data points from majority class are used. Under-sampling is also performed, and the performance of these two techniques are compared.

|  |  |
| --- | --- |
|  |  |
| Under-sampling | Over-sampling |

Figure 1. accuracy, precision and recall scores for cross-validation

I track 3 metrics for this problem: accuracy, precision and recall. I do not have enough domain knowledge to tell which one, false positive or false negative, to reduce, so having both precision and recall seem to be reasonable. Other than the first fold in over-sampling, there is no significant difference between the scores.

The machine learning algorithm I use in this project is called XGBoost. It is a boosting algorithm with great benefits such as fast and not likely to overfit. No other more sophisticated algorithms are tested in this project. If time permits, stacking algorithm could be another good choice.

Confusion matrix for the evaluation case is shown in Figure 2.

|  |  |
| --- | --- |
|  |  |
| Under-sampling | Over-sampling |

Figure 2. confusion matrix for evaluation

Over-sampling has slightly fewer false positives and false negatives than under-sampling. I would argue both techniques give very good results, considering only standard features are used and there is not much model turning.

# Question 2

This question boils down to which features are more important to determine hospital admissions. XGBoost calculates feature importance when the model is trained, which is a very good indicator.

|  |  |
| --- | --- |
|  |  |
| Feature importance from XGBoost | Shapley value |

Figure 3. feature importance analysis from XGBoost and Shapley value

With all features provided in the dataset, food insecurity is the most important factor that affects hospital admission, followed by depression, poverty, ACG\_CHG\_IND\_CD (?), chronic condition and age. Other factors have much less importance than the aforementioned ones. Gender and face have almost no effect on hospital admission.

Another powerful tool that can be used to determine feature importance is the Shapley value. Shapley value is from game theory, it indicates the contribution of each data points to the final average prediction. Figure 3 shows Shapley value based on XGBoost.

* the color bar indicates the relative place of a value in a single column.
* x axis is the shapley value, calculated for each data point
* the position of each data point shows its contribution to the prediction: negative means it drives the prediction down and positive means it drives the prediction up
* the sequence (top down) shows the magnitude of each feature to the prediction

Shapley value offer much more granularity than feature importance plot on the left, as it tracks each individual data points. For instance when we look at age, there is a red cluster of points with positive Shapley value. That mean older people (red) would move the prediction towards positive, which is 1 (with hospital admission), which makes logical sense. Another example is poverty. Lower poverty value (blue) tends to drive prediction to negative, which is 0 (without hospital admission).

# Question 3

This question can be answered with some EDA. A very informative way is to plot histogram for both customers with high admission rate and low admission rate. Note that the original question asks for the customer with highest hospital admission. Unfortunately the highest admission is 10, and there is only one customer for it. To have a meaningful plot, I relax the condition to high admission (>=6). The top 5 features from the feature importance plot are chosen for the histogram:

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |

Figure 4. value distribution for low admission (others) and high admission (max)

There seems to be not much difference between the value distribution for low and high admissions. That means at least for the current features provided, there is not enough information to distinguish high admission group from low admission group. We might need to dig deeper for other features that could determine the number of admissions.