CISC 451 - Assignment 2

Will Murray: 20054564, Emma Ritcey: 20007918, Rylen Sampson: 20051918

Preliminaries

For this assignment, we used Python as our programming language of choice. For reference, it can be downloaded from the official Python website. To extend the data analytic capabilities of Python, numerous libraries were utilized. Some of these libraries come pre-installed as they are popular enough among Python users but for those that are not, Python has a built-in library installer called Pip which we used to install those libraries. To install a library, enter the command “pip install name_of_library” into the command line, where name_of_library is the name of the library you wish to install. Below is a list of libraries used in our work:

- os: Allows one to navigate directories in a Python script.
- matplotlib: library to create static, animated, and interactive visualizations
- numpy: numpy provides data structures and algorithms designed for data analytics.
- pandas: pandas is a fundamental data analytics library.
- sklearn: sklearn provides easy-to-use popular data analytic algorithms.
- scipy: Offers many statistical methods useful when performing any type of analysis.
- plotly: plotly makes creating quick data-driven plots simple with very few lines of code.
- seaborn: another easy to use data visualization library built on Matplotlib.

Data Exploration

Rather than jumping right into visualizations of the data, we decided it would be better to understand the condition of the data. This proved to be an intuitive decision as we immediately noticed issues. The first issue being that some features had an abundance of missing values. For features where over 40% of the records were missing, the features were dropped.
Following the missing values, another notable issue were features composed entirely of one label aside from the odd-case. As this was an issue associated with the categorical variables, we further discuss it in the following exploration of the categorical variables.
To better understand which categorical features we may wish to include in our analysis we employed mutual information to gain insight on the features ability to foretell the target variable. Mutual information essentially says, given we know this feature, how much information does this knowledge give in predicting the target variable, so one would hope to have features with a high mutual information with respect to the target variable. In our case, the mutual information across all features was rather low and thus we had to look at the features in this context. In the first Plotly produced figure, we see the mutual information for continuous variables. In the second Plotly produced figure, we see the mutual information of categorical variables with the largest label ratio overlaid on top. We chose to do this as it highlights the features that are nearly consumed entirely by one response and consequently have a low or zero mutual information score. These results motivate our feature selection.
Lastly, we also looked at the correlation between numeric features. No pair of features possessed an overly high correlation but the *time_in_hospital* feature was correlated enough with the other features we felt comfortable removing it on this premise. This can be seen in the correlation matrix plot.

All exploratory analysis plots are generated by data_cleaning.py

Data Preparation

Based on our exploratory analysis, we were able to reduce the number of features to a select 17. These can be found in the run.py file.
The remaining features in the dataset are dropped from the analysis as we've deemed them to be not important for modeling.
Categorical features where the labels have an inherent ordering were converted using sklearn's *LabelEncoder* function whereas those which an order has no meaning were converted using the *OneHotEncoder* function. For example, *race* and *gender* were converted to numeric representations via *OneHotEncoder*.
Given the number of numeric converted categorical features we selected for our final collection of features, we opted to use a neural network as our model. Originally, we planned to implement a Support Vector Machine (SVM) classifier but SVM's tend to fail on categorical data as they rely on the concept of Euclidean distance which categorical data naturally does not possess. Therefore, we hypothesized a neural network may pose a better chance of learning the "unknown" classification function.

Modeling

To implement a neural network, we used the *MLPClassifier()* function from the sklearn library. This provides an out-of-the-box neural network ready for use with minimal parameters to tune and architecture choices. The *GridSearchCV()* function in the sklearn library provides an easy way to hunt for the optimal configuration of parameters; we apply this function prior to predicting the test labels in an attempt to obtain the best possible model.
As mentioned, the original model we had in mind was an SVM. We did test this model but, it only ended up achieving an accuracy of **57%**. Therefore, our reasoning for switching to a neural network model was warranted.
Below are the outputs of our neural network model which include the confusion matrix as well as the accuracy achieved with best configuration on the validation data. Included in our submission is a file containing the predictions on the test data that way our proposed model can be evaluated.
The neural network achieved a **58%** accuracy during validation where the confusion matrix is...

	Actual		
Predicted	**211**	4522	5440
	158	**12765**	18309
	34	9626	**39701**

This is not a large improvement on the SVM model and so we're led to believe the issue may be poor quality data as through cleaning and modeling we noticed many glaring pitfalls within the data.