Final Project – Stroke Prediction Using Machine Learning Algorithms

Adnan Shilleh

CSCI 4364: Machine Learning

Professor Trott

April 28th, 2021

According to the World Health Organization (WHO) stroke is the 2nd leading cause of death globally, responsible for approximately 11% of total deaths. For my project, I thought it would be interesting to use Machine Learning algorithms to try and predict patients who are at highest risk for strokes. A lot of computer science (and technology in general) is focused on the next technological invention/breakthrough. Topics that usually dominate the news include Mars exploration, self-driving cars, and next-generation hardware. However, technological innovation should be most important when it comes to improving quality of life such as revolutionizing medicine, creating programs that will aid in the fight for environmental justice, and using machine learning to solve other social issues. Kaggle provides a dataset that includes parameters like gender, age, various diseases, and smoking status. Each row in the data provides relevant information about the patient.

To analyze the data, I used all of the machine learning algorithms discussed in class (SVM, Naives Bayes, Logistic Regression, Decision Trees, and Random Forests). In addition to this, I researched two other algorithms to use in my project that are relevant to the course material. Originally developed by Microsoft, the Light Gradient Boosting Machine (LightGBM) , is a gradient boosting framework that uses tree based learning algorithms. The algorithm is meant to be optimized for speed and memory usage. XGBoost also utilizes machine learning algorithms under the Gradient Boosting framework.

Throughout the semester, I have focused on improving my data pre-processing skills. I learned that the more analysis I conduct of training sets, the better the accuracy of each algorithm is. Using the method .describe( ) gives me a good snapshot of what each attribute looks like (count, average, standard deviation, and max). At the beginning

of the semester, I struggled with preparing datasets so that each classifier could read through them without returning some type of runtime error. I used the final project as an opportunity to research extensively how to pre-process data correctly.

Whenever examining a dataset, the first thing I do is check for any NULL values (indicated by NaN). From using .describe( ), the BMI feature had some missing values. To solve this, I used the average BMI to replace the NULL values. Re-examination of the data showed that fields no longer had any null values. Converting string data to integer data was something I spent a lot of time on during the homework assignments. I learned how to use LabelEncoder( ) to address this issue. I converted the attributes gender, married, work type, residence type, and smoking status from categorical data to numerical variables.

From my research, I came to understand how important feature correlation is. Understanding the relationship between multiple variables could help identify any dependencies or associations. I used a heatmap to try and identify any such relationships. Several features had negative correlations while the strongest positive relationship was between “age” and “ever married”. When a feature is highly correlated with other features, we could remove one of them to reduce the complexity while potentially improving the model's learning.

If the target class is highly imbalanced, the model will be biased towards the majority class. For this project, I used Synthetic Minority Oversampling Technique (SMOTE). To summarize, this technique oversamples the examples in the minority class. This can be achieved by duplicating instances from the minority class in the training dataset prior to fitting a model. SMOTE uses the first algorithm we learned this semester, K-Nearest Neighbors, to synthesize minority data.

At this point, the data is almost ready for analysis by the classifiers. After splitting the data, I conducted some normalization. Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1. There is another feature scaling technique, standardization, which I chose not to do. This is because the dataset contains inherent important differences between features. The data is now finally read to be put through each classifier.

To analyze each algorithm, I used sensitivity score, specificity score, AUC, precision, recall, and F1. Sensitivity is a measure of the proportion of actual positive cases that got predicted as positive (or true positive). Specificity is defined as the proportion of actual negatives, which got predicted as the negative (or true negative).An The receiver operating characteristic curve (ROC) is a graph showing the performance of a classification model at all classification thresholds. The area under the curve (AUC) measures the entire area under the ROC curve and provides us an aggregate measure of performance across all possible classification thresholds. Precision determines the proportion of positive classifications that were correct while Recall determines the proportion of positive instances that were identified correctly. Finally, F1 is a function of Precision and Recall (this offers us a balance between the two measures).

The algorithm that completed its analysis the fastest was (unsurprisingly) Naive Bayes while the longest was SVM. When using the analysis metrics above, XGBoost was slightly more accurate than LightGBM and the Random Forest classifier. The later two algorithms were neck and neck in terms of performance while the rest of the algorithms tended to lag behind in comparison. However, the top three algorithms (Random Forest, LightGBM, XGBoost) all had an AUC value of 0.99. The Naive Bayes classifier was the least accurate of any algorithm - which is a result I was expecting.

In the final part of my project, I calculated feature importance with the Random Forest classifier and XGBoost. Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. Both of the algorithms I used have inbuilt methods to directly return the feature performance. According to my research, using XGBoost is better for this calculation because it is much faster and is much more reliable than linear models. According to the graphs that were created, it seems that age is the most important factor when predicting if a patient will have a stroke or not. This would make sense because according to the CDC: “Age increases your chances for stroke. The older you are, the more likely you are to have a stroke.”

I benefited greatly from doing this project mainly because I was able to vastly improve my data pre-processing skills. When doing the homework assignments, my models would return accuracy scores that ranged from 60% to low/mid 90%. Models are only as good as the data you give them, so being able to rigorously optimize my data helped improve all of my models vastly. Additionally, I was also able to learn about two new machine learning methods, XGBoost and LightGBM. In the future, I will likely use XGBoost for a lot of data analysis considering it is extremely accurate and has a lot of useful methods built into it.

On the next page, I am going to provide all of the links I used for my research. I am also going to specify what those links were used for. In the page after, I will provide a bibliography page cited in APA format.

LabelEncoder( ):

- sklearn.preprocessing.LabelEncoder — scikit-learn 0.24.2 documentation

Feature Correlation:

- Why Feature Correlation Matters …. A Lot! | by Will Badr | Towards Data Science

Heatmaps:

- Better Heatmaps and Correlation Matrix Plots in Python | by Drazen Zaric | Towards Data Science

Histograms:

- numpy.histogram — NumPy v1.20 Manual

Data Oversampling:

- SMOTE — Version 0.8.0 (imbalanced-learn.org)
- Resampling to Properly Handle Imbalanced Datasets in Machine Learning | by Younes Charfaoui | Heartbeat (fritz.ai)
- SMOTE for Imbalanced Classification with Python (machinelearningmastery.com)

Feature Scaling:

- Feature Scaling | Standardization Vs Normalization (analyticsvidhya.com)

AUC-ROC Curves:

- AUC-ROC Curve in Machine Learning Clearly Explained - Analytics Vidhya
- Classification: ROC Curve and AUC | Machine Learning Crash Course (google.com)

Receiver Operating Characteristic:

- Receiver Operating Characteristic (ROC) — scikit-learn 0.24.2 documentation

LightGBM:

- Welcome to LightGBM's documentation! — LightGBM 3.2.1.99 documentation

XGBoost:

- XGBoost Documentation — xgboost 1.5.0-SNAPSHOT documentation

Feature Importance:

- Feature Importance and Feature Selection With XGBoost in Python (machinelearningmastery.com)

Accuracy Metrics:

- ML Metrics: Sensitivity vs. Specificity - DZone AI
- Classification: Precision and Recall | Machine Learning Crash Course (google.com)
- Accuracy, Precision, Recall or F1? | by Koo Ping Shung | Towards Data Science

Bibliography

Badr, W. (2019, January 18). *Why Feature Correlation Matters….A Lot!* Towards Data Science. https://towardsdatascience.com/why-feature-correlation-matters-a-lot-847e8ba439c4

Bhandari, A. (2020, April 3). *Feature Scaling for Machine Learning: Understanding the Difference Between Normalization vs. Standardization*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/04/feature-scaling-machine-learning-normalization-standardization/

Bhandari, A. (2020, June 16). *AUC-ROC Curve in Machine Learning Clearly Explained*. Analytics Vidhya. https://www.analyticsvidhya.com/blog/2020/06/auc-roc-curve-machine-learning/

Brownlee, J. (2016, August 31). *Feature Importance and Feature Selection With XGBoost in Python*. Machine Learning Mastery. https://machinelearningmastery.com/feature-importance-and-feature-selection-with-xgboost-in-python/

Brownlee, J. (2020, January 17). *SMOTE for Imbalanced Classification with Python*. Machine Learning Mastery. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/

Centers for Disease Control and Prevention. (2017, January 17). *Stroke Risk*. Stroke. https://www.cdc.gov/stroke/risk_factors.htm

Charfaoui, Y. (2019, December 6). *Resampling to Properly Handle Imbalanced Datasets in Machine Learning*. Heartbeat. https://heartbeat.fritz.ai/resampling-to-properly-handle-imbalanced-datasets-in-machine-learning-64d82c16ceaa

Cournapeau, D. (2020). *sklearn.preprocessing.LabelEncoder*. Scikit Learn. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html

Google. (2020, February 10). *Classification: Precision and Recall*. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall

Google. (2020, February 10). *Classification: ROC Curve and AUC*. Machine Learning Crash Course. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc

Holtz, Y. (2018, January 1). *Grouped Barplot*. Python Graph Gallery. https://www.python-graph-gallery.com/11-grouped-barplot

Imbalanced Learn. (2021, April 29). *SMOTE*. SMOTE Documentation. https://imbalanced-learn.org/stable/references/generated/imblearn.over_sampling.SMOTE.html

Kumar, A. (2018, September 27). *ML Metrics: Sensitivity vs. Specificity*. DZone. https://dzone.com/articles/ml-metrics-sensitivity-vs-specificity-difference

Microsoft. (2021, January 1). *LightGBM*. LightGBM Documentation. https://lightgbm.readthedocs.io/en/latest/

NumPy. (2021, January 31). *numpy.histogram*. Numpy v1.20 Manual. https://numpy.org/doc/stable/reference/generated/numpy.histogram.html

Scikit Learn. (2020, April 28). *Receiver Operating Characteristic (ROC)*. Scikit-learn 0.24.2 Documentation. https://scikit-learn.org/stable/auto_examples/model_selection/plot_roc.html

Shung, K. P. (2018, March 15). *Accuracy, Precision, Recall or F1?* Towards Data Science. https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9

XGBoost. (2020). *XGBoost Documentation*. XGBoost 1.5.0-SNAPSHOT Documentation.

Zaric, D. (2019, April 15). *Better Heatmaps and Correlation Matrix Plots in Python*. Towards Data Science. https://towardsdatascience.com/better-heatmaps-and-correlation-matrix-plots-in-python-41445d0f2bec