Stroke Prediction Case Study: Logistic Regression vs XGBoost

Justin Heer

The length of this paper is 5.5 pages without any figures/tables/references

Abstract

A stroke is a life threatening disease caused by a blockage of blood flow in the brain. Predicting its occurrence could potentially save many lives, thus a dataset was constructed for this purpose. In this paper, we perform an analysis of the dataset and test the performance of Regularized Logistic Regression (RLR) against an XGBoost implementation for stroke classification. The XGBoost model is able to capture an additional 1.4% of the true positive class compared to RLR, while requiring 44.7 minutes of computational time, 321% longer than RLR. The benefit of the increased captured proportion of the true positive class outweighs the computational advantages of RLR.
keywords: Stroke, XGBoost, Logistic Regression

1 Introduction

1.1 Strokes

A stroke, or brain attack, is a medical condition causing a sudden loss of brain function because of a blockage of the blood supply inside the brain [FF20], preventing the brain from getting enough oxygen and nutrients to respond to the body's metabolic demand. As a consequence, the brain cells die in the impacted area; the severity of the stroke depends on where it breaks out and how extensive the impacted area is [FF20]. In the United States in 2018, one in six deaths from cardiovascular disease was due to a stroke; every 40 seconds someone has a stroke and every four minutes someone dies of a stroke [DP18]. The financial burden of stroke related medical costs in the USA is nearly 46 billion dollars between 2014 and 2015 [DP18]. In Canada, stroke is the top cause of premature death in women [HS]. Hence, the ability to predict a stroke is extremely useful from both a public health and financial perspective. In this paper, we will be exploring two classification algorithms and their ability to predict stroke occurrence.

1.2 Regularized Logistic Regression

Regression methods have become an integral component of any data analysis concerned with describing the relationship between a response variable and one or more explanatory variables [HLS13]. Quite often the outcome variable is discrete, taking on two or more possible values. The Logistic Regression model is the most frequently used regression model for the analysis of this type of data [HLS13].

In case when several parameters are used Logistic Regression will be prone to overfitting. To avoid that a regularized parameter (λ) is introduced. This variation is called Regularized Logistic Regression (RLR). Higher values of λ will cause under-fitting and smaller values can lead to overfitting, thus, the value of λ needs to be tuned as a hyper-parameter in the model building process[Nas+20]. The regularized loss function using least squares regression is

$$\sum_{i}^{n}(y_i - \hat{y_i})^2 + \lambda \sum_{j}^{k} \beta_j^2 \tag{1}$$

Where λ is the regularization coefficient and the β_j are the coefficients of the variables that we are trying to regularize, oftentimes this is every parameter except β_0, the constant term. Significance testing can also be performed to determine if the β coefficients are statistically significant in the model's predictive ability. The Wald test can be used to calculate the z-score, such that

$$Z = \frac{\hat{\beta}_i}{SE(\beta_i e)} \tag{2}$$

As was covered in the lecture material

1.3 XGBoost

Among the many machine learning methods, *tree* boosting is a highly effective and widely used technique [Fri01].

Definition 1.1. A *tree* in mathematics is a diagram of nodes connected by lines, representing paths that continue outwards and do not loop back. The starting node is called the root, each following node is a child and the incoming node is the parent.

XGBoost is a scalable machine learning system for gradient tree boosting, one of the most effective machine learning models for predictive analytics [SG15]. It belongs to the family of decision tree learning methods which map observations about an item to conclusions about the item's target value in a tree structure. Depending on

the characteristics of the target value they can be used for regression or classification [SG15].

The most important factor behind the success of XGBoost is its scalability in all scenarios [CG16], which is due to several important systems and algorithmic optimizations. These innovations include: a novel tree learning algorithm for handling sparse data; a theoretically justified weighted quantile sketch procedure enables handling instance weights in approximate tree learning [CG16].

For a given dataset with n examples and m features, a tree *ensemble* model uses K additive functions to predict the output

$$\hat{y_i} = \phi(\mathbf{X}_i) = \sum_{k=1}^{K} f_k(\mathbf{X}_i) \tag{3}$$

Where $f_k \in \mathcal{F}$, and

$$\mathcal{F} := f(\mathbf{X}) = w_{q(\mathbf{X})}, q : \mathbb{R}^m \to T, w \in \mathbb{R}^T$$

Thus $\mathcal{F}$ is the mathematical space of the regression trees. Here q represents the structure of each tree that maps an example to the corresponding leaf index. T is the number of leaves in the tree. Each f_k corresponds to an independent tree structure q and leaf weights w [CG16].

Definition 1.2. A group of predictors is called an *ensemble*. Ensemble learning methods involve creating multiple predictors and using predictions from multiple models to generate combined - or ensembled - predictions

To learn the set of functions used in the model, XGBoost minimizes the following regularized objective function

$$L(\phi) = \sum_i l(\hat{y_i}, y_i) + \Omega(f_k) \text{ where } \Omega(f) = \gamma T + \frac{1}{2}\lambda ||w||^2 \tag{4}$$

The Ω term penalizes the complexity of regression tree functions and helps smooth the final learned weights to avoid overfitting [CG16]. A standard boosting algorithm would train sub-sequential functions on the residuals of the previous iteration, XGBoost does this by performing gradient decent to minimize the residual error function and optimize the initiation of the next function.

The system is available as an open source package [CG16]. The impact of the system has been widely recognized in a number of machine learning and data mining challenges [CG16].

1.4 Coordinate Descent Method

To tune the hyper-parameters in our prediction algorithms we will be using Coordinate Descent (CD). CD algorithms have been shown to yield fast estimates for optimal hyper-parameters [LR10]. In a regression setting, CD has the advantage of not requiring complicated matrix inversions or eigen-decompositions, making it computationally efficient when there are a large number of features [LR10]. The main advantage of these methods is the simplicity of each iteration, both in generating the search direction and in performing the update of variables [Nes12].

$$\text{Let } P := \{p_1, .., p_n\} \text{be the set of all tunable hyper-parameters for a given algorithm} \tag{5}$$

$$\text{Let } l(P) = L(p_1, .., p_n) \text{be the loss function for a model, given an input of parameters} \tag{6}$$

Then coordinate descent is implemented as described in **Algorithm 1**. This algorithm can by cycled multiple times to generate improved estimates of the optimal hyper-parameters, in the case of this paper we cycle the algorithm only once.

1.5 Correlation

Correlation or dependence is any statistical relationship, whether causal or not, between two random variables or bi-variate data. In the broadest sense correlation is any statistical association, though it commonly refers to the degree to which a pair of variables are linearly related [Cro39]

1.5.1 Pearson's r

Pearsons's r coefficient measures only the degree of linear dependency between two variables x and y [EL09]. The value for the coefficient can vary from $[-1, 1]$, it can be calculated as

$$r = \frac{n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right)}{\sqrt{n\left(\sum_{i=1}^{n} x_i^2\right) - \left(\sum_{i=1}^{n} x_i\right)^2}\sqrt{n\left(\sum_{i=1}^{n} y_i^2\right) - \left(\sum_{i=1}^{n} y_i\right)^2}} \tag{7}$$

Algorithm 1 Coordinate Descent

Create set S of n elements such that S_i will be the optimal value for parameter p_i
for $p_i \in P$ **do**
  Set values to iterate through to optimize p_i
  $p\_test\_vals = \{\text{set of values}\}$
  $min\_error = \infty$
  for $p \in p\_test\_vals$ **do**
    $error = l(p_1, ..p, ..p_n)$
    if $error < min\_error$ **then**
      $min\_error = error$
      $best\_p = p$
    end if
  end for
  $S[i] = best\_p$
end for

1.5.2 Spearman's r - Rank Order Correlation

Spearman's r also takes values in $[-1, 1]$ where a value of 1 indicates a perfect association of ranks [WGP16]. Spearman's coefficient measures how well the relationship between two variables x and y can be represented by a *monotonic* function.

Definition 1.3. A *monotonic* function is a function which is either completely non-increasing or non-decreasing. A function is monotonic if its first derivative does not change sign.

Spearman's coefficient can be calculated using the procedure outlined in **Algorithm 2**. The advantage of using Spearman's coefficient is that it can capture non-linear correlations between variables. In this paper we use both to interpret the correlation between covariates.

Algorithm 2 Spearman's r coefficient

Start with 2 variables x and y of length n
define $r_x := \{r_{x_i} \in [1, n], r_{x_i} \in \mathbb{N}\}$ and r_y similarly
rank the values in the x,y from highest to lowest
Store the ranking such that if x_i is the j^{th} largest value then $r_{x_i} = j$
$d = r_x - r_y$, element wise subtraction
$\rho = 1 - \frac{6 \sum_i d_i^2}{n(n^2-1)}$

2 Methods

2.1 The dataset

The dataset used for this case study is available online at kaggle (url: https://www.kaggle.com/fedesoriano/stroke-prediction-dataset). The source of the data is confidential as the data pertains to health care, and it is to be used for educational purposes only. The dataset contains 5110 observations with 11 covariates. The covariates are listed in **table** 1

The intent behind the construction of this dataset is to use it to predict stroke occurrence. Each row in the data-set provides relevant information regarding a patient. There are a total of 249 patients with stroke and 4861 patients without a stroke, thus the data-set is unbalanced and this is addressed in the modelling process.

2.2 Preprocessing

The dataset contained missing values for BMI, these were imputed with the mean value to maintain a consistent numerical representation for the feature.

2.3 Numerical Features Analysis

The pairwise plot (**figure** 1) show that there are weak correlations between age and BMI, and age and average glucose level, with no correlation between BMI and average glucose level. We compare the differences in the distributions of the recorded observations and whether a stroke had occurred (**figure** 2). This shows that

Name	Data Type	Description
Gender	Categorical	Male, Female or Other
Work Type	Categorical	Children, Govt job, never worked, private or self-employed
Residence Type	Categorical	Rural or urban
Smoking Status	Categorical	Formerly smoked, never smoked, smokes or unknown
Age	Numerical	Age of the patient
Avg Glucose Level	Numerical	Average glucose level in blood
BMI	Numerical	Body mass index
Hypertension	Boolean	Diagnosis results
Heart Disease	Boolean	Diagnosis results
Ever Married	Boolean	Martial History
Stroke	Boolean	Target

Table 1: Caption

patients that had strokes were often older, may have had higher average glucose levels as well as a slightly higher BMI. The histograms also reveal that a significantly high number of patients that had strokes also had missing BMI values.

2.3.1 T test

The T test was conducted to determine if mean age, glucose level, or BMI differed for patients with or without a previous stroke. The hypotheses were $h_0 := \mu_{stroke} = \mu_{no\ stroke}, h_a := \mu_{stroke} \neq \mu_{no\ stroke}$, for the three numerical variables. The results of the T test are stated in **table** 2. In all three of the cases we reject the null hypothesis that the means for stroke and no stroke are the same at the 99% confidence level.

Covariate	mean Stroke	mean No Stroke	T statistic	p-value
Age	67.73	41.97	29.69	≈ 0
Avg glucose	132.545	104.80	6.98	$2.4\ e-11$
BMI	30.55	28.84	4.27	$2.59\ e-5$

Table 2: T test results for numerical covariates

2.4 Categorical Feature Analysis : Logistic Regression

To determine the significance of the categorical variables logistic regression test was conducted.

Variable	coef	std err	z	p-value	[0.025	0.975]
gender	-0.0703	0.143	-0.492	0.623	-0.350	0.210
age	0.0750	0.005	14.097	0.000	0.065	0.085
hypertension	1.4902	0.162	9.212	0.000	1.173	1.807
heart disease	1.6569	0.190	8.725	0.000	1.285	2.029
ever married	-1.5056	0.223	-6.748	0.000	-1.943	-1.068
work type	-0.2998	0.076	-3.965	0.000	-0.448	-0.152
Residence type	-0.0598	0.142	-0.423	0.673	-0.337	0.218
avg glucose level	00.0112	0.001	9.245	0.000	0.009	0.014
bmi	0.0242	0.008	2.972	0.003	0.008	0.040
smoking status	-0.3616	0.069	-5.237	0.000	-0.497	-0.226

Table 3: Logistic Regression Significance Test Results

The results presented in **table** 3 show that gender and residence type are the only variables that are not significant, the rest of the categorical variables are statistically significant at the 99% confidence level. The test used to determine statistical significance was the aforementioned Wald Test. The numerical variables were included in the test as well, all three are once again significant.

2.5 Testing Hypothesis

In this experiment we will be using XGBoost with the same objective function as RLR. It is expected that the accuracy of the XGBoost model will be superior to standard RLR, due to the added advantage of the gradient

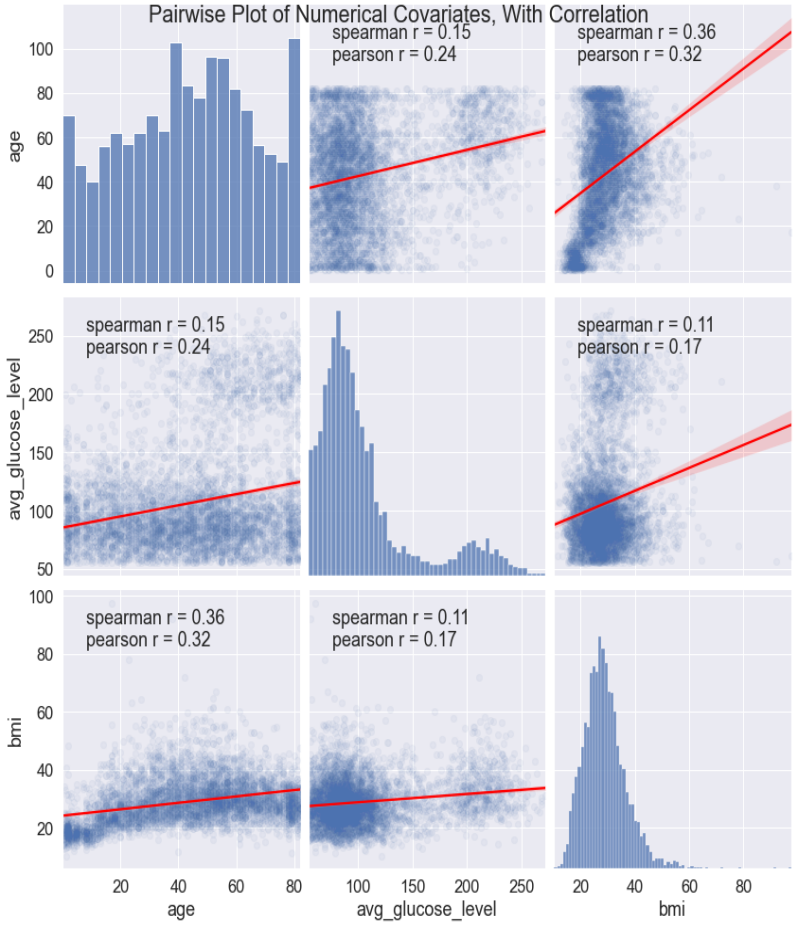

Figure 1: Scatter plots with spearman and pearson correlation values, histograms show somewhat normal distribution for BMI

boosted tree structure. However, we also expect that the training time for the XGBoost model will be longer than logistic regression, thus we will be examining the performance gains of the XGBoost model comparatively with the time efficiency loss.

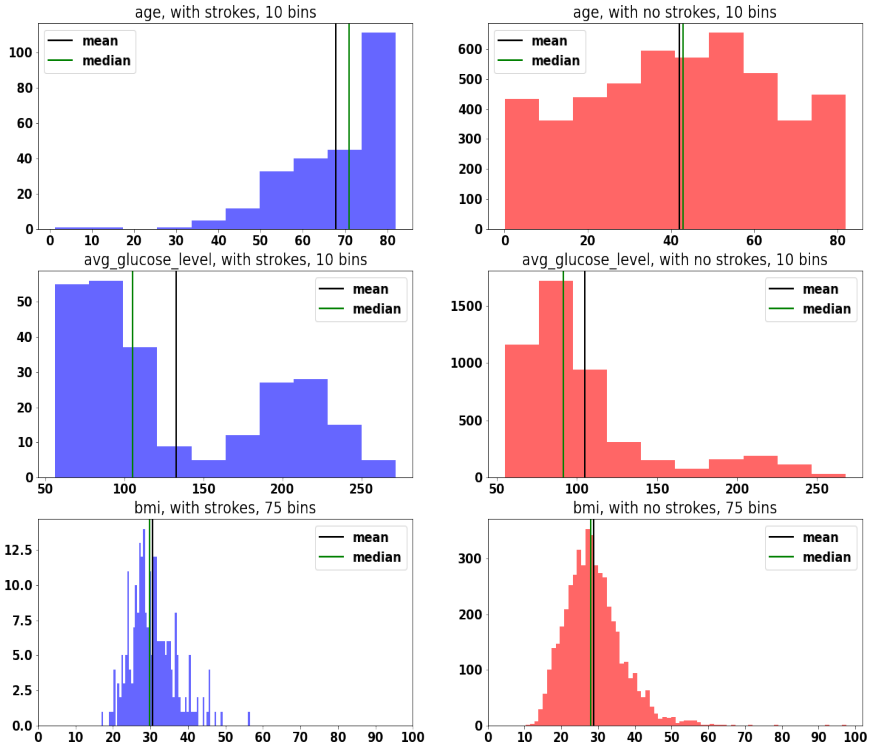

Figure 2: Histograms showing observation distribution aggregated by stroke occurrence

3 Model Development

A total of 247 observations out of 5110 contain a positive stroke result, a predictive model that simply predicts "no stroke" for every observation would have an accuracy of 95%. This behaviour is unwanted as the model would not be useful in practice. To defer a model from learning this behaviour we train the model on a balanced subset of the dataset and calculate a custom score as our primary optimization metric.

3.1 Sampling and validation

We take all 247 positive predictions and randomly sample 247 negative predictions, for a combined total of 494 observations in the sampled dataset. 50 such samples were generated. For each sample the data was split into two subsets, training and validation, each consisting of 80% and 20% of the original sample, respectively. The training set is used to train the model and the validation set is used to measure the model's performance. The hyper-parameters are tuned to give the best performance on the validation data, however the model is never trained on the validation set. Once the model has been tuned another 50 random samples, similar to above, are generated, this sub-sample contains 80% training data and 20% testing data. The model with its tuned hyper-parameters is then trained on this training data and its performance is measured on the test and reported.

Since we had opted to create a balanced dataset, the effective dataset size was reduced to only 50 random samples of each consisting of 494 observations. In order to have enough training data to build a well performing model we needed to ensure at least 80% of the sample was used for the training set. In a typical train/validation/test split this would have resulted in small validation and testing sets, increasing the variability of the results. Thus to combat this the dataset was sampled once to create the training/validation sets and once again to create the training/testing sets. The random sampling of the non-stroke data ensures that at least half the data is randomized for each individual sample.

3.2 Metrics

3.2.1 Classification Metric

The classification accuracy of a model can be a poor measure of the models predictive ability, particularly for unbalanced datasets. Although we have balanced our dataset in the previous step, we still need to use an intelligent metric to accurately measure the usefulness of our model. We will use a custom variation of the F1-score to measure the model's performance.

Definition 3.1. *Precision* is the number of true positives, TP, out of the sum of the true positives and the false positives,FP, succinctly

$$Precision = \frac{TP}{TP + FP}$$

Definition 3.2. *Recall* is the number of true positives out of the sum of the true positives and the false negatives, FN, succinctly

$$Recall = \frac{TP}{TP + FN}$$

In the context of our model, the precision measures how often a positive prediction is correct and the recall measures what proportion of the true positives were actually identified. Considering our model is predicting stroke occurrence, a life threatening disease, it is desirable to capture as many true positives as possible. Thus our primary optimization metric will be recall. However, focusing only on recall will lead to many false positives and poor precision, false positives could potentially burden the healthcare system and have a greater long term negative impact. Thus our metric should account for both, with a preference towards recall.

Definition 3.3. For the duration of this paper, *Score* is defined

$$Score := Recall \cdot 0.6 + Precision \cdot 0.4$$

This combined metric takes into account both recall and precision but stresses the importance of a higher recall score.

3.2.2 Training Time

We will be measuring time needed to train and tune all hyper-parameters for both models, this will be used to compare the benefit of XGBoost with the added time cost.

4 Results

4.1 Hyper-parameter tuning

4.1.1 Regularized Logistic Regression

We utilized an implementation of RLR by Scikit-Learn [Ped+11]. We tuned five hyper-parameters to generate the final model using coordinate descent method, and only two of the five hyper-parameters required precise tuning. The tuned hyper-parameter values along with constant parameters are stated in **table** 4. The total training time was 10.6 minutes.

Parameter	value	# tested	Description
C	0.03	1000	regularization,equivalent to λ
intercept scaling	0.9	1000	intercept scaling value
max iter	4000	5	max steps the solver takes
solver	lib-linear	5	optimization algorithm
fit intercept	True	2	fit bias parameter or not
penalty	L2	1	loss function
tolerance	$1e-14$	1	smallest observed difference
random state	1	1	sets random seed for solver

Table 4: Tuned hyper-parameter values for regularized logistic regression model, # tested represents the number of parameter values that were tested during tuning

4.1.2 XGBoost

We tuned 8 hyper-parameters to generate the final tuned model. The tuned hyper-parameter values along with 1 constant parameter are stated in **table** 5. The total training time to tune eight hyper-parameters was 44.7 minutes, which is 34.1 minutes longer than RLR.

Parameter	value	# tested	Description
lambda	0.4	1000	L2 Regularization
alpha	0	1000	L1 Regularization
max depth	2	1000	Maximum tree depth
min child weight	47.5	1000	Min sum of weights needed in leaf
gamma	0.4	1000	Min loss reduction to make partition
num rounds	13	50	Akin to "K", total number of models
eta	0.1	10	Learning rate
booster	gbtree	2	Akin to solver
objective	binary:hinge	1	Loss Function

Table 5: Tuned hyper-parameter values for XGBoost model, # tested represents the number of parameter values that were tested during tuning

4.2 Discussion

The results from both models using the tuned hyper-parameter values previously listed are stated in **table** 6. The XGBoost model outperforms logistic regression in every accuracy category, achieving a higher recall, precision and score. The XGBoost model is using the RLR loss function for computation, with the added benefit of the gradient boosted trees. Thus, it is sensible that the XGBoost model outperforms standard RLR. The tuning time required to achieve this performance is 321% more than what RLR required.

Using Canada as a reference example, with an estimated 50,000 strokes per year using XGBoost over Logistic Regression would accurately predict 700 more cases of stroke. Although logistic regression models are faster to train, the benefit of capturing an additional 1.4% of the true positive class outweighs the computational advantages as both models are still relatively efficient and can test results instantaneously on a modest computer. The added tuning time for XGBoost stems largely from the additional parameters that must be tuned, the runtime of an individual iteration of the algorithm is only 102% more than a single iteration of logistic regression.

Model	Score	Recall	Precision	Training Time (mins)
Regularized Logistic Regression	0.807	0.930	0.623	10.6
XGBoost	0.816	0.944	0.625	44.7

Table 6: Resulting scores from tuned models

5 Limitations

5.1 Dataset source

One limitation is the unknown performance abilities on new dataset sources. The source of the current dataset is unknown thus we cannot make claims about the performance of the model on unseen data. We also do not know if the data was collected in a stratified manner from all ethnic groups, thus the model may not perform as well on marginalized members of the population.

5.2 Low Positive Class Occurrences

As previously addressed, the dataset is unbalanced. Every sampled dataset used in the training and tuning process contained the same positive stroke class observations. Although no data leakage occurred, the model still has not been sufficiently tested on a wide variety of training data for the positive test class.

6 Recommendations

6.1 Handling Missing Data

The preliminary analysis of the data revealed that patients with missing BMI scores had noticeably higher stroke occurrences. In the case of the current models missing data was simply imputed, however, several sophisticated packages - including XGBoost - include capabilities to handle missing data using more sophisticated back end methods. Since some information is lost when missing data is imputed it would be interesting to see the performance impact when using these more sophisticated methods of handling missing data.

6.2 Time sensitive predictions

Rather than just predicting if a stroke occurred or not it could be more beneficial to include t he time after which the data was collected that the stroke occurred. This would enable researchers to build a model that could predict stroke onset time, which would help medical professionals prioritize high risk patients.

7 Conclusion

We have presented a comparison between regularized logistic regression and XGBoost in their abilities to predict stroke occurrence. Stroke is a deadly disease and being able to predict its occurrence can potentially save many lives. XGBoost outperforms logistic regression in recall, precision and in our custom score, while taking 321% longer to tune. Despite the increasing time required to tune it is still feasible to compute on a modest machine, making it a good choice for application in industry.

References

[CG16] Tianqi Chen and Carlos Guestrin. "XGBoost: A Scalable Tree Boosting System". eng. In: *Proceedings of the 22nd ACM SIGKDD International Conference on knowledge discovery and data mining*. KDD '16. ACM, 2016, pp. 785–794. ISBN: 1450342329.

[Cro39] Frederick Emory Croxton. *Applied general statistics / by Frederick E. Croxton ... and Dudley J. Cowden.* eng. New York: Prentice-Hall, Inc., 1939.

[DP18] Centers for Disease Control and Prevention. "Underlying Cause of Death". In: *CDC WONDER Online Database* (2018).

[EL09] L Egghe and L Leydesdorff. "The relation between Pearson's correlation coefficient r and Salton's cosine measure". eng. In: *Journal of the American Society for Information Science and Technology* 60.5 (2009), pp. 1027–1036. ISSN: 1532-2882.

[FF20] Gérard Foucher and Sébastien Faure. "What is a stroke?" In: *Actualités Pharmaceutiques* 59.600 (2020), pp. 57–60. ISSN: 0515-3700. DOI: https://doi.org/10.1016/j.actpha.2020.09.020. URL: https://www.sciencedirect.com/science/article/pii/S0515370020303803.

[Fri01] Jerome H Friedman. "Greedy function approximation: a gradient boosting machine". In: *Annals of statistics* (2001), pp. 1189–1232.

[HLS13] David W Hosmer, Stanley Lemeshow, and Rodney X Sturdivant. *Applied logistic regression, third edition*. eng. 3rd ed. Wiley series in probability and statistics. Hoboken, NJ: John Wiley and Sons, 2013. ISBN: 0470582472.

[HS] Heart and Stroke. *Stroke report*. URL: https://www.heartandstroke.ca/what-we-do/media-centre/stroke-report.

[LR10] Alexander Lorbert and Peter Ramadge. "Descent methods for tuning parameter refinement". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 469–476.

[Nas+20] Subham Naskar et al. "Application of Machine Learning in Various Fields of Medical Science". In: *Smart Healthcare Analytics in IoT Enabled Environment*. Ed. by Prasant Kumar Pattnaik, Suneeta Mohanty, and Satarupa Mohanty. Cham: Springer International Publishing, 2020, pp. 109–126. ISBN: 978-3-030-37551-5. DOI: 10.1007/978-3-030-37551-5_7. URL: https://doi.org/10.1007/978-3-030-37551-5_7.

[Nes12] Yu Nesterov. "Efficiency of Coordinate Descent Methods on Huge-Scale Optimization Problems". eng. In: *SIAM journal on optimization* 22.2 (2012), pp. 341–362. ISSN: 1052-6234.

[Ped+11] F. Pedregosa et al. “Scikit-learn: Machine Learning in Python”. In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[SG15] Ivana Semanjski and Sidharta Gautama. “Smart City Mobility Application–Gradient Boosting Trees for Mobility Prediction and Analysis Based on Crowdsourced Data”. eng. In: *Sensors (Basel, Switzerland)* 15.7 (2015), pp. 15974–15987. ISSN: 1424-8220.

[WGP16] Joost C. F de Winter, Samuel D Gosling, and Jeff Potter. “Comparing the Pearson and Spearman Correlation Coefficients Across Distributions and Sample Sizes: A Tutorial Using Simulations and Empirical Data”. eng. In: *Psychological methods* 21.3 (2016), pp. 273–290. ISSN: 1082-989X.