Abstract

According to the World Health organization, heart disease has remained the leading cause of death in the world for the past 18 years. In 2016, there are 10 million deaths due to heart disease worldwide. Early detection of heart disease would greatly increase the chances for more successful treatments. Thus, with the goal of developing a sound prediction model for heart disease, our team employed machine learning methods on a dataset consisting of data from heart disease patients at the Cleveland Clinic Foundation. Our team applied three types of feature selection and engineering techniques on the dataset and later performed modeling using seven classification algorithms – logistic regression, gradient boosting, support vector machine, random forest and naïve bayes. Upon analyzing the experimental results, it turned out that the Naïve Bayes classification model seems to be a good predictor of heart disease.

Keywords: data exploration, feature selection, feature engineering, modeling

I. Background

Our group analyzed the Heart Disease dataset from Driven Data. The dataset consists of 270 patient records. Each patient record has 14 attributes. The 14 attributes are health and heart-related data points that provide context to the overall patient's diagnosis. Each patient is diagnosed with the presence or absence of heart disease, which is the target variable in this dataset. As a team we identified that the goal of this project was to work with and use all of the data provided to run classification machine learning algorithms to predict the target variable, heart disease. With the use of the 13 other attributes provided in the dataset, our team has sought to predict and better understand what specific data points and classification models work best to predict heart disease. Our team has applied six different machine learning classification model algorithms to the dataset to identify and predict what variables and models can best predict the presence of heart disease.

II. Hypothesis

Before we begin manipulating the data by cleansing, visualizing, or classification model work, we want to put forward a hypothesis about the data. After analyzing and making sense of the 14 different attributes in the dataset, it appears that some attributes may be more insightful than others for predicting the target variable. Our team concluded that the attributes: age, chest pain, blood pressure, and cholesterol may be some of the most insightful data points that will build robust predictions as to whether or not a patient has heart disease.

Although the data points captured in the dataset did allow for some interesting analysis that we could glean insights from, there were several attributes that we thought were lacking from the dataset. According to research published by the Oxford Academy in 2013 indicated that tobacco is highly correlated with cardiovascular issues (Rigotti, 2013). Therefore, we think it would have been valuable to have this data point as an attribute in the dataset. Some other data points we would have liked to have had in the dataset were the weight, height, and/or BM of each patient, as other studies, such as the one conducted by Harvard Medical School by Kelly Bilodeau, has indicated that these factors are also highly correlated with heart disease (Bilodeau, 2018). Regardless, the data points provided were sufficient for performing a deep analysis of the dataset. It is a well-known fact that heart disease is the number one cause of death in America; according to the CDC and their 2016 Leading Cause of Death Report, heart disease topped the charts beating out all types of cancers combined as the leading cause of death in America. Based on some underlying assumptions, there are 2 main and common reasons for death old age and excessively unhealthy habits. From these two assumptions, our hypothesis is that age and high cholesterol will be the two main indicators and drivers for predicting our target variable.

III. Methodology

Importing, Viewing and Cleaning the Data

The raw data found on drivendata.org is in .dat format rather than in csv format. Rather than converting this dataset to a csv file using Excel, we decided to prep the data entirely using python. To do so, we imported the file using Panda's read_csv function and separate each column with a space delimiter. Since the original .dat file does not contain easily understandable column headings, we assigned them manually in order to easily understand our data in order and have included markups of what they are and how they are interpreted.

Once the data had been imported into the Jupyter Notebook, visualizing the data became the initial step to begin making sense of the scope of the raw data. The decision to visualize the newly imported data is necessary to gain some preliminary insights of the data. Pandas, Matplotlib, and Seaborn are the packages commonly used to bring the tabular data to life by presenting the data in the form of graphs and other visualization methods. Prior to creating any detailed or robust visualizations it is important to first draw insights from the data using some tools that provide a general summary of each of the collected data points.

A good first step in performing this analysis is the *.describe* line of code. This simple line of code analyzes the data by verifying the number of collected data points for each row, as well as provides a cursory view of the mean, standard deviation, minimum and maximum values for each column of data.

Once this analysis has been performed it is helpful to identify the type of values in the data set as well as analyzing if the dataset has any null values. A simple line of code that allows one to easily perform this analysis is *.info().* This will provide an analysis of the type of data stored in each of your columns as well as indicate how many null values are in each column. Fortunately, the heart Disease dataset did not have any null values stored in any of the columns, and the types were all floats or integers which will increase our data processing abilities.

Next, the first real visualization was employed which was a histogram. The use of histograms easily show the distribution of collected data. We plotted each column into a histogram because every column was either made up of integers or floats, which both easily plot a onto a histogram. By plotting a histogram you can easily determine which variables are binomial or polynomial, which will then provide insight into how you drive further data analysis.

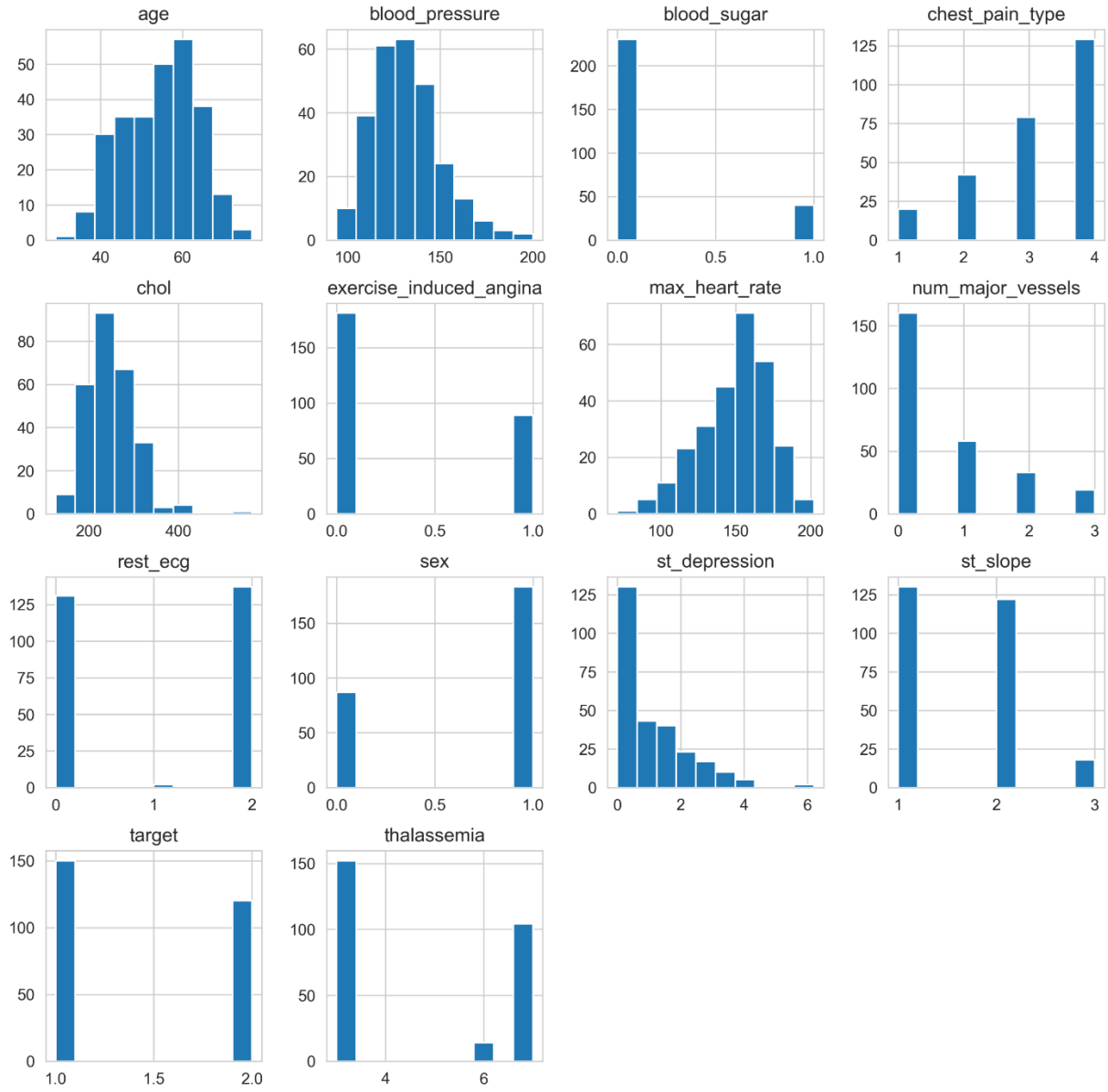


Figure 1. Histograms of 14 features of heart disease dataset

After analyzing the distribution of each variable we sought to see how these variables were correlated with each other by the use of a correlation matrix. By running a correlation matrix of the data we can see what attributes are higher indicators of predicting whether or not the patient has heart disease. A correlation matrix helps indicate what variables are more highly correlated with other variables. Correlation Matrices are great for exploring and understanding the data relationships. The matrix indicates that the following variables are correlated with our target variable which is the presence or absence of heart disease: 'chest_pain_type', 'max_heart_rate_achieved', 'exercise_induced_angina', 'st_depression', 'num_major_vessels', and 'thalassemia' are our best indicators of heart disease.

In order to gain insight into the target variable and dive deeper into the classification analysis, it is important to understand the target variable and see what the distribution of that specific variable is. As used before in the initial visual analysis, a histogram is one of the clearest ways to see the distribution of a binomial variable. Visualizing the target variable with a histogram was used to isolate this variable and see the distribution of the patient diagnosis in the data set. The target variable is represented by a binary variable where 1 is a negative diagnosis and 2 represents a positive diagnosis.

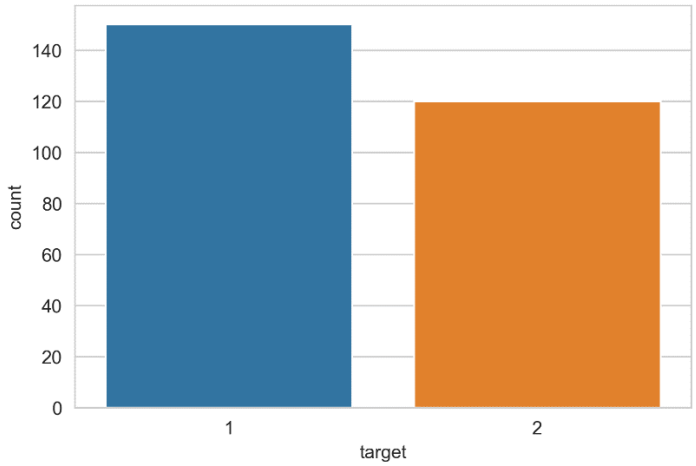


Figure 2. Histogram of target variable

The distribution is not perfectly equal, but there is a substantial population of each diagnosis that will help drive toward a classification prediction. In regard to the sample, we wanted to see if there were other influential binominal attributes that could skew the data one way or another. The variable that was most noticeable that could influence this was sex. Based on the initial histogram it is noticeable that sex is not an evenly distributed variable across the dataset. This is one variable that was worth exploring and seeing how this might affect our analysis and classification models. We found that the distribution of sex was a two-thirds to one-third split, heavily favoring males:

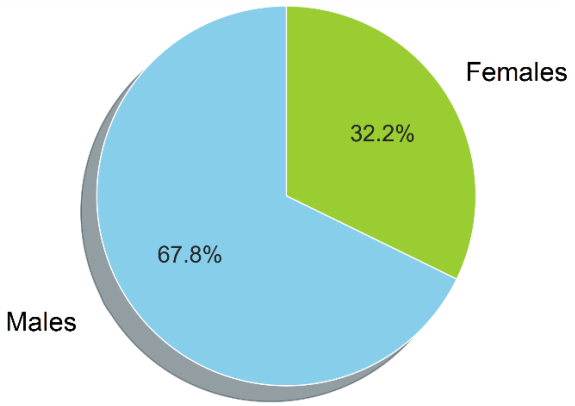


Figure 3. Distribution of sex in the heart disease dataset

This finding was alarming as we feared this sample data would provide a biased view toward predicting heart disease in male rather than females. However, this imbalance was mitigated once we dove slightly deeper into how this variable was distributed. As it turns out the distribution of heart disease was more evenly distributed among males and females as is indicated by the distribution of the blue bar.

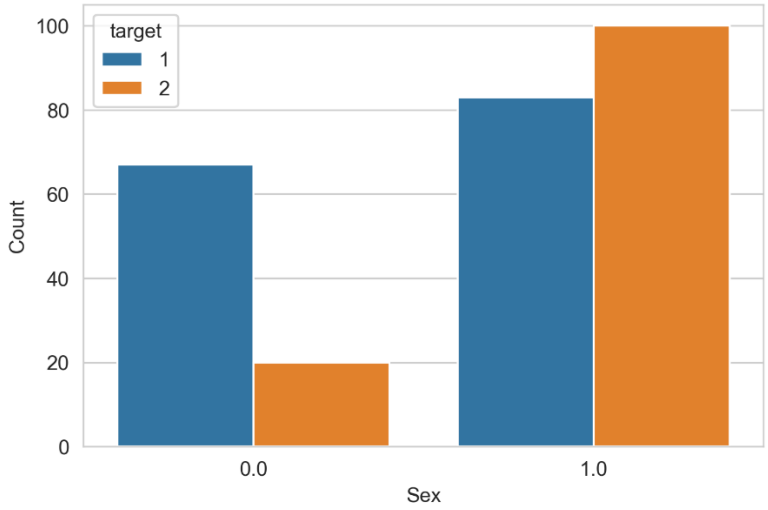


Figure 4. Sex and target variable distributions

A finding from the correlation table that we have generated earlier on, thalassemia was the highest correlated attribute with our target variable. This specific variable was interesting to discover that it correlated highly with heart disease since no one in the five-person group has heard of this before. This drove us to research what this variable was. What was discovered is that thalassemia is a hereditary blood disorder by which one's red blood cells circulate less oxygen throughout one's body than the body needs. This is disorder is determined by a blood test (National Human Genome Research Institute). This research provided another insight into our data as this indicated that all the patients have had blood work performed. 150 patients did not have heart disease in our dataset. When we compare that with the 152 patients whose thalassemia is normal, this drove us to suspect that this variable may be one that drives the accuracy of our classification algorithms. This variable is one that we hope to gain insight into is whether or not there is a meaningful overlap in these two numbers.

Feature Selection

After cleaning and exploring the data, we continue our data preparation with feature selection. Feature selection is essentially a task to remove unnecessary features. One of the primary motivations for feature selection is the curse of dimensionality (Liu & Motoda 2014). Feature selection effectively reduces the hypothesis space by removing redundant and irrelevant features. The smaller the space, the easier it is to find correct hypotheses.

There are many tangible benefits when feature selection is performed correctly, such as an improvement of the inductive learner, either in terms of learning speed, generalization capacity or simplicity of the induced model (Bolón-Canedo & Alonso-Betanzos 2018).

There is a variety of feature selection taxonomies; two wildly accepted ones are (1) filter, wrapper and embedded, and (2) multivariate and univariate (Ni 2012). For our python project, we decided to cover a range of these selection methods – (i) Selection using Correlation Matrix, (ii) Univariate Selection; (iii) Feature Importance. We have examined Correlation Matrix, a filter technique, in an earlier section, thus we will be focusing on the other two methods.

Univariate Selection

Univariate feature selection techniques assess the discriminative power of each feature individually by evaluating, pairwise, the dependency between response and each feature based on specific metric (Ni 2012). This method is mainly employed in tackling the extremely high-dimensional dataset because of its speed and cost-effectiveness. The univariate selection technique that we have chosen to apply is a Chi-Square test. The chi-square test examines the difference between expected and observed distributions. We chose this as it works well on categorical variables (Weaver, Morales, Dunn, Godde, Weaver 2017).

Chi-Square Test:

$$\chi_c^2 = \sum \frac{(O_i - E_i)^2}{E_i}$$

We specify our number of features as 10, denoted by k=10, to analyze the top 10 features based on their chi-square score. We identified *max_heart_rate*, *num_major_vessels*, *thalassemia*, and *st_depression* as some of the top features in our dataset.

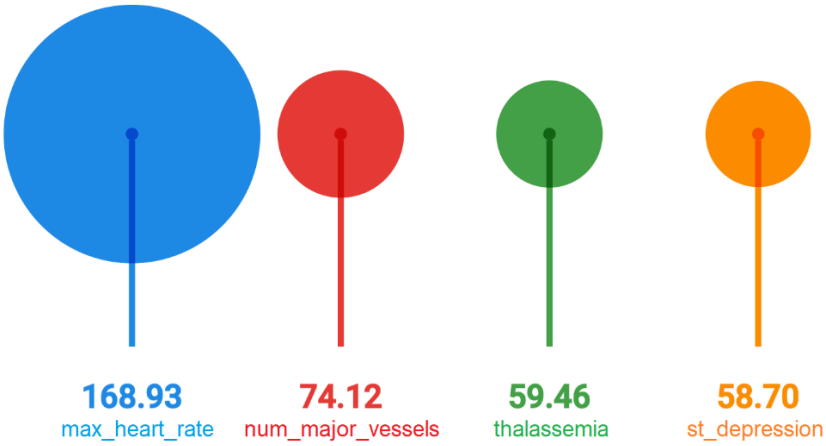


Figure 5. Top 4 features selected by Chi-Square

Feature Importance

The Correlation Matrix and Chi-Square test are filter techniques that measure the relevance of features by their correlation, either linearly or using a frequency distribution, with a dependent variable. Feature Importance, on the other hand, is a wrapper method that measure the usefulness of a subset of feature by actually training a model on it.

In Python, Feature Importance is a common inbuilt class that comes with Tree Based Classifier, making it an embedded technique. A score is given to each feature, the higher the score, the more important or useful the feature is to the output model. Using this method, we define a model with 100 trees and run feature importance on our independent variables. We then plotted the result of the top 10 features (based on their scores) on a bar chart.

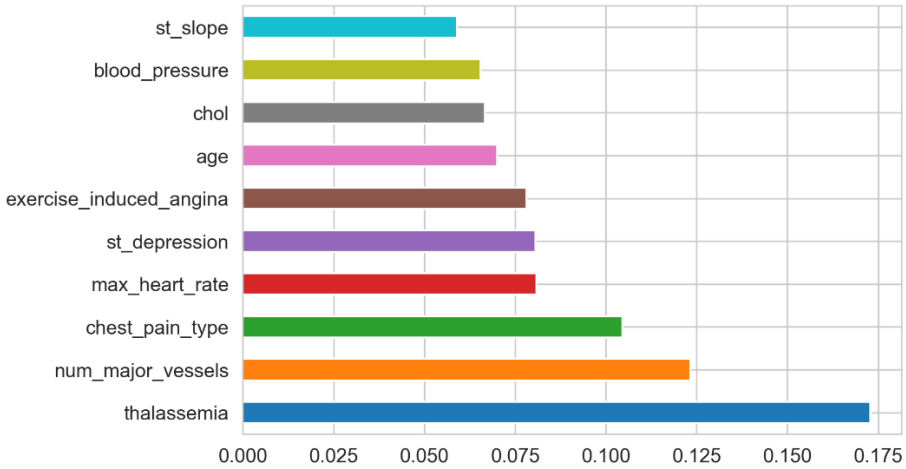


Figure 6. Top 10 features (based on their scores) selected by Feature Importance

Feature Engineering

Another pre-modeling preparation that we have applied is Feature Engineering. Feature engineering is an act of extracting features from raw data and transforming them into formats that are suitable for the machine learning model (Zheng & Casari 2018). There is a multitude of feature engineering techniques. The feature selection that we have chosen is an encoding technique. There are three ways of encoding, namely one-hot, dummy, and effect coding. We used one-hot encoding in our project.

According to Zheng & Casari's Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists (2018):

> "One-hot encoding is redundant, which allows for multiple valid models for the same problem. The non-uniqueness is sometimes problematic for interpretation, but the advantage is that each feature clearly corresponds to a category."

Although we do not have missing data in our project, they can be encoded as the all-zeros vector, and the output would be the overall mean of the target variable.

The *pandas.get_dummies* function helps us convert categorical variables into dummy/indicator variables. The resulting shape of our engineered dataset as we print it shows that 15 new (dummy) variables have been generated. Prior to creating dummy variables, we ran all our machine learning models and recorded their performances (i.e. accuracy, precision, and recall). With feature engineering, we observed an overall improvement (ranging 1 - 5%) in performances throughout all models.

Modeling: Machine Learning Algorithms

In machine learning, an algorithm or method extracts patterns from data to help us solve problems. These are problems that machine learning can solve:

Problem	Machine learning category
Fitting some data to a function or function approximation	Supervised learning
Figuring out what the data is without any feedback	Unsupervised learning
Maximizing rewards over time	Reinforcement learning

Figure 7: The problems that machine learning can solve. Source: Kirk, M. (2017).

Our project is a case of supervised learning. The table below summarizes some of the supervised type algorithms that are widely adopted. For our modeling stage, we have chosen to apply and evaluate 7 ML algorithms: K-Nearest Neighbors (KNN), Logistic Regression, Gradient Boosting, Support Vector Machine (SVM), Decision Tree, Random Forest, and Naive Bayes.

Algorithm	Learning type	Class	Restriction bias	Preference bias
K-Nearest Neighbors	Supervised	Instance based	Generally speaking, KNN is good for measuring distance-based approximations; it suffers from the curse of dimensionality	Prefers problems that are distance based
Naive Bayes	Supervised	Probabilistic	Works on problems where the inputs are independent from each other	Prefers problems where the probability will always be greater than zero for each class
Decision Trees/ Random Forests	Supervised	Tree	Becomes less useful on problems with low covariance	Prefers problems with categorical data
Support Vector Machines	Supervised	Decision boundary	Works where there is a definite distinction between two classifications	Prefers binary classification problems
Neural Networks	Supervised	Nonlinear functional approximation	Little restriction bias	Prefers binary inputs
Hidden Markov Models	Supervised/ Unsupervised	Markovian	Generally works well for system information where the Markov assumption holds	Prefers time-series data and memoryless information

Figure 8: Machine learning algorithm matrix. Source: Kirk, M. (2017).

I. K-Nearest Neighbor (KNN)

KNN is a typical example of a lazy learner, not because of its apparent simplicity, but because it does not learn a discriminative function from the training data; instead, it memorizes the training dataset (Raschka 2015).

The algorithm is straightforward. Traditionally, we first choose the number of k and a distance metric. We then find the k nearest neighbors of the sample we want to classify. Finally, we assign a class label by majority vote. In our python project, we started with an arbitrary k value of 10 and scored the model based on accuracy. This returns an accuracy of 66.7%.

Using a re-iteration method by defining a range between 1 and 100 for k, we plotted a graph that shows us the accuracy given this range of k values. We identified that at k=22, the highest accuracy score is obtained

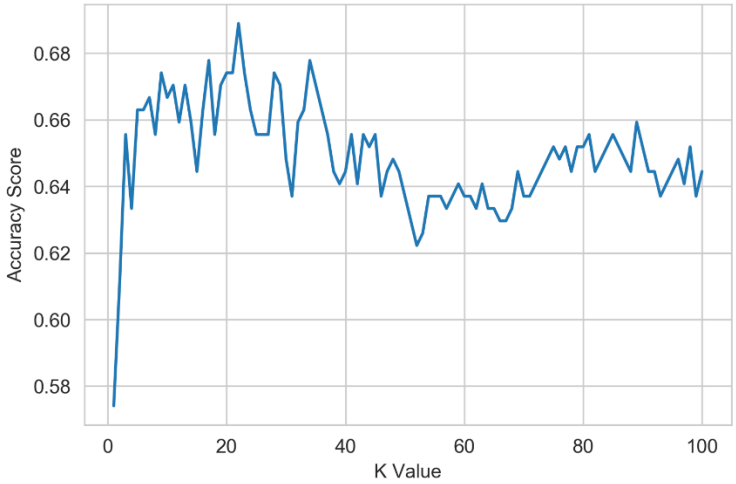


Figure 9. K-Value accuracy score graph

Performance Metric	Score
Accuracy	68.889%
Precision	68.436%
Recall	81.333%
F1 Score	74.276%

Table 1. KNN Performance (k=22)

Cross Validation

The most basic modeling methodology is arguably one developed on train data and evaluated on test data (i.e. train-test split). In this scenario, there is a chance that train and test might not have been homogeneously selected, resulting in extreme cases that might appear in the test data, reducing performance (Dangeti 2017).

To overcome this problem, we use a cross-validation technique that ensures robustness in the model (at the expense of computation). In cross-validation, data is divided into equal parts and training performed on all the other parts of the data except one part, on which performance is evaluated. This process is repeated as many parts (i.e. folds) as a user has chosen.

According to Russell & Cohn's Cross Validation (2012):

> "One round of cross-validation involves partitioning a sample of data into complementary subsets, performing the analysis on one subset (called the training set), and validating the analysis on the other subset (called the validation set or testing set). To reduce variability, multiple rounds of cross-validation are performed using different partitions, and the validation results are averaged over the rounds."

An industrial norm for the number of folds is 10 (Kuhn & Johnson 2013). This is the value that we have defined for our number of cross-validation folds in our KNN algorithm, and subsequently tested on our other models: logistic regression, gradient boosting, support vector machine, random forest and naive bayes.

II. Logistic Regression

Logistic regression is one of the most used statistical procedures employed by statisticians and researchers for the analysis of binary and proportional response data (Hilbe 2009). This is why we have decided to use it on our largely binary/polynomial dataset. True enough, logistic regression performed relatively well:

Performance Metric	Score
Accuracy	86.296%
Precision	86.352%
Recall	90.667%
F1 Score	88.034%

Table 2. Logistic Regression Performance

There is a wide range of statistical methods in logistic regression, many software packages have different variations of executing logistic regression. We used a package imported from *sklearn.linear_model.* This class implements a regularized logistic regression using the 'liblinear' library, 'newton-cg', 'sag', 'saga' and 'lbfgs' solvers by default.

III. Gradient Boosting

Gradient boosting (GB) is a competition-winning algorithm that works on the principle of boosting weak learners iteratively by shifting focus towards problematic observations that were difficult to predict in previous iterations and performing an ensemble of weak learners, typically decision trees (Dangeti 2017).

According to Dangeti (2017), gradient boosting involves three elements:

1. Loss function to be optimized. In boosting, at each stage, unexplained loss from prior iterations will be optimized rather than starting from scratch.
2. Weak learner to make predictions: Decision trees are used as a week learner.
3. Addictive model to add weak learners to minimize the loss function: Trees are added one at at time and existing trees in the model are not changed. The gradient descent procedure is used to minimize the loss when adding trees.

For our model, we defined the value of "loss" to be "exponential", with a "learning rate" of 0.03, this shrinks the contribution of each tree by 0.03. We set the number of boosting stages to 75. Since gradient boosting is fairly robust to overfitting; a large number usually results in better performance. Finally, we set the maximum depth, which limits the number of nodes in the tree, to be 6.

Performance Metric	Score
Accuracy	75.926%
Precision	78.292%
Recall	78.667%
F1 Score	78.508%

Table 3. Gradient Boosting Performance

IV. Support Vector Machine (SVM)

SVM is a powerful supervised machine learning algorithm (Chang & Lin 2011). This model achieved the highest recall rate (98%) out of all our models. We observed accuracy and precision scores to be relatively low. To improve the accuracy score, we feature engineered our SVM by applying feature scaling, a tool especially useful on SVM (Juszczak, Tax, & Duin 2002), on six variables – age, blood pressure, cholesterol, maximum heart rate, and st depression. Feature scaling normalizes the range of these independent variables of data.

Performance Metric	Score
Accuracy	57.407%
Precision	56.786%
Recall	98%
F1 Score	71.899%

Table 4. SVM Performance

After applying feature scaling, we ran the model with an initial cross-validation of 10-folds; however, we realized that 5-folds produced better performance. With feature scaling, the performance of SVM improved significantly.

This encouraged us to explore other means of data manipulation. We implemented chi-square feature selection and ran results from percentiles ranging from 1 to 100 searching for all accuracy and recall values greater than their mean values. We noticed that the highest accuracy score is at the 63rd percentile of our Chi-square statistic. Overall, performance has improved. Since Chi-square is a univariate selection method that selects features with the strongest relationship to the target; in this experiment, 18 of the original 28 features were used.

Performance Metric	Score
Accuracy	84.444%
Precision	83.436%
Recall	90%
F1 Score	86.572%

Table 5. After Feature Scaling and Cross Validation of 5-Folds

```
[[137  13]
 [ 23  97]]
the best percentage so far: 63
the best recall so far 0.9133333333333334

[[137  13]
 [ 23  97]]
the best percentage so far: 63
the best accuracy so far 0.8666666666666666
```

Figure 10. Highest Recall and Accuracy Score after Applying Feature Selection and Chi-Square Statistics

V. Decision Tree

The decision tree classifier is a simple, yet effective machine learning algorithm (Banfield, Hall, Bowyer, & Kegelmeyer 2006) as it performs classification without requiring much computation. It helps us to calculate possible consequences such as chance event outcomes. We split our data 70-30 (i.e. 70% trained, 30% tested). Here's our result:

Performance Metric	Score
Accuracy	70.370%
Precision	67.500%
Recall	71.053%
F1 Score	69.231%

Table 6. Train-Test Split Ratio 70-30 Performance

We were interested in seeing how changing the train-test split ratio to 80-20 would affect the score results as it feeds more data to the training set. Based on the results below, it appears a 80-20 split generated an overall better performance than the 70-30 split.

Performance Metric	Score
Accuracy	79.630%
Precision	85.714%
Recall	69.231%
F1 Score	76.596%

Table 7. Train-Test Split Ratio 80-20 Performance

VI. Random Forest

Random Forest is an ensemble of decision trees, essentially creating a "forest" with randomly chosen subsets of features. We defined 100 trees in our parameters and ran two experiments: (i) 80-20 train-test split; (ii) cross validation of 10-folds.

Performance Metric	Score
Accuracy	85.185%
Precision	79.545%
Recall	92.105%
F1 Score	85.366%

Table 8. Train-Test Split Ratio Performance

Performance Metric	Score
Accuracy	82.593%
Precision	83.660%
Recall	86.667%
F1 Score	84.749%

Table 9. Using Cross Validation of 10-Folds

Surprisingly, the 80-20 split had a slightly better overall performance. This was against our expectation that cross validation would process data better in all aspects, apparently, this was not the case. It is important, however, to note that cross validation yield better precision.

VII. Naive Bayes

Naive Bayes is adopted from Bayesian' theorem and a family of algorithms that shares a common principle in which every pair of features being classified is independent of each other. Naive Bayes is an effective tool in data mining (Kim, Han, Rim, & Myaeng 2006). We used Naive Bayes classifier because the parameters were estimated using maximum likelihood. Just like the other models, we ran train-test split ratios of 70-30 and 80-20, and a cross validation of 10-folds.

Performance Metric	Score
Accuracy	85.185%
Precision	79.545%
Recall	92.105%
F1 Score	85.366%

Table 10. Train-Test Split Ratio 70-30 Performance

Performance Metric	Score
Accuracy	88.889%
Precision	85.714%
Recall	92.308%
F1 Score	88.889%

Table 11. Train-Test Split Ratio 80-20 Performance

Performance Metric	Score
Accuracy	81.852%
Precision	84.319%
Recall	84.667%
F1 Score	83.935%

Table 12. Cross Validation of 10-Folds

Comparing the three results obtained, 80-20 train-test split produced the highest results in terms of accuracy, precision, recall, and F1 score. Recall score on 70-30 was approximately similar to 80-20, while cross-validation of 10-folds generated uniform results (ie. most scores in the 80% range).

Learnings

Appendix I provides an overview of performances across all models in our project. Naive Bayes seems to be the leading contender in terms of accuracy, recall, and F1 performances, while Logistic Regression with cross validation of 5-folds achieved the highest precision result.

In evaluating the different metrics of performance and relating them back to the context of predicting heart disease, it might be useful to examine the number of false positives and false negatives. Our team concluded that a model with less false negative would serve as a better model since there are detrimental consequences when a patient who has heart disease is misdiagnosed and miss treatment as a result. Therefore, a model with high recall performance (denoting a lower number of false negative relative to true positive) is one that we strive to achieve in this heart disease prediction problem.

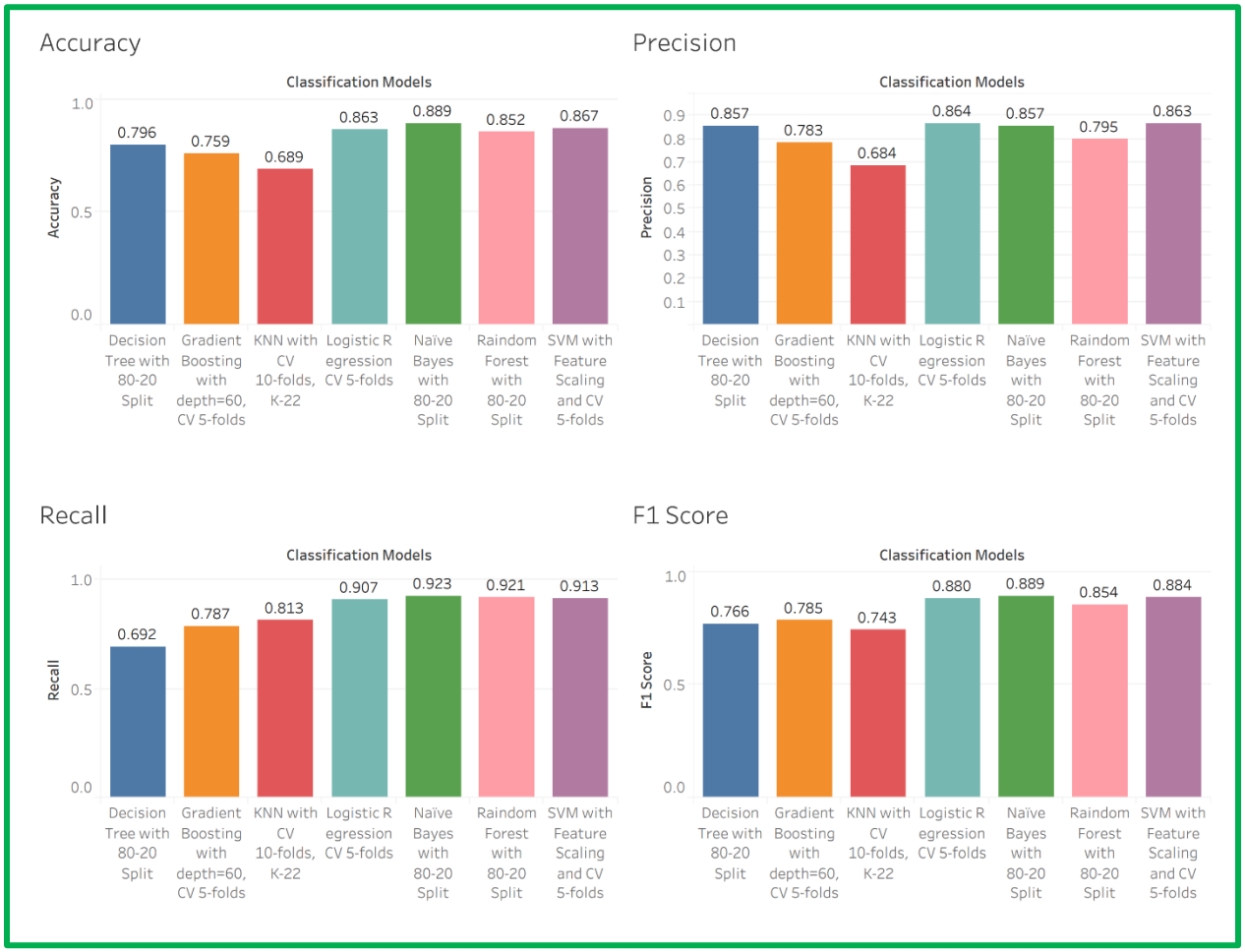
All in all, a large part of our project was a series of trial-and-error experimentation. There are three main take-aways that we would like to allude to:

Firstly, feature selection is a great way to improve performance. By extracting only relevant features from the dataset, we could train our data more accurately and build a better predictive model.

Secondly, feature engineering techniques such as one-hot encoding creates additional candidate features to choose from and use for training, while feature scaling normalizes to provide more compatible and comparable features. Evidently from our project, performances across the board improved drastically when feature selection and feature engineering methods were used.

Lastly, we noticed that cross validation performs differently on every model. The goal of this technique is to avoid underfitting, overfitting, and estimate the level of fit of a model to a data set. While cross validation does not necessarily improve accuracy, our research has shown that it is a better alternative to a regular 70-30 train-test split.

Appendix I: Performance Across All Models



References

Banfield, R., Hall, L., Bowyer, K., & Kegelmeyer, W. (2006). A Comparison of Decision Tree Ensemble Creation Techniques. IEEE Transactions on Pattern Analysis and Machine Intelligence. Volume 29, Issue 1.

Biau, G., & Devroye, L. (2015). Lectures on the Nearest Neighbor Method. Springer International Publishing.

Bilodeau, K. (2018). Belly Fat Linked With Higher Heart Disease Risk. Harvard Health Publishing. Retrieved from: https://www.health.harvard.edu/blog/belly-fat-linked-with-higher-heart-disease-risk-2018072614354

Bolón-Canedo, V., & Alonso-Betanzos, A. (2018). Recent Advances in Ensembles for Feature Selection. Springer

Chang, C., & Lin, C. (2011). LIBSVM: A library for support vector machines. ACM Transactions on Intelligent Systems and Technology. Volume 2 Issue 3.

Dangeti, P. (2017). Statistics for Machine Learning. Packt Publishing.

Heron, M. (2018). Deaths: Leading Cause for 2016. National Vital Statistics Reports Volume 67, Number 6. Retrieved from: https://www.cdc.gov/nchs/data/nvsr/nvsr67/nvsr67_06.pdf

Hilbe, J. (2009). Logistic Regression Models. Boca Raton, FL: Chapman & Hall/CRC.

Juszczak, P., Tax, D., & Duin, R. (2002). Feature scaling in support vector data description. Advanced School for Computing and Imaging.

Kim, S., Han, K., Rim, H., & Myaeng, S. (2006). Some Effective Techniques for Naive Bayes Text Classification. IEEE Transactions on Knowledge and Data Engineering. Volume 18, Issue 11.

Kirk, M. (2017). Thoughtful Machine Learning with Python: A Test-Driven Approach. Sebastopol, CA: O'Reilly Media Inc.

Kuhn, M., & Johnson, K. (2013). Applied Predictive Model. Springer Publishing.

Liu, H., & Motoda, H. (2014). Computational Methods of Feature Selection. Boca Raton, FL: Chapman & Hall/CRC.

National Human Genome Research Institute. (2014, May 14). Learning About Wilson Disease. Retrieved from: https://www.genome.gov/

Ni, W. (2012). A Review and Comparative Study on Univariate Feature Selection Techniques. University of Cincinnati.

Raschka, S. (2015). Python Machine Learning. Packt Publishing.). Python Machine Learning. Packt Publishing.

Rigotti, N. (2013). Managing Tobacco Use: The Neglected Cardiovascular Disease Risk Factor. European Heart Journal Volume 34, Issue 42.
Retrieved from: https://academic.oup.com/eurheartj/article/34/42/3259/519402

Russell, J., & Cohn, R. (2012). Cross-Validation. Book on Demand.

Weaver, K., Morales, V., Dunn, S., Godde, K., Weaver, P. (2017). An Introduction to Statistical Analysis in Research. Wiley.

Zheng, A., & Casari, A. (2018). Feature Engineering for Machine Learning: Principles and Techniques for Data Scientists. Sebastopol, CA: O'Reilly Media Inc.