Preventative Measures for Heart Disease

Xing Yang Lan, Arya Rajpal, Sruthi Rayasam, David Wong, Shileng Zhang.
Statistics Department
University of California, Davis
lan@ucdavis.edu, ajrajpal@ucdavis.edu, srayasam@ucdavis.edu
dggwong@ucdavis.edu, seazhang@ucdavis.edu

Abstract

A cleaned dataset of 21 indicators for heart disease was used to explore the information available and application of the data towards making a sensitive predictive model that can identify risk factors early on. Individual features are fairly weak predictors, but our entropy calculations reveal that together they provide lots of information. We concluded that a gradient boosted random forest ensemble was the most optimal model and implemented one for modelling. At a predictive outcome cutoff representing all people with heart disease, the model had over 92.7% sensitivity. At cutoff representing patients diagnosed with with heart disease, the specificity was over 93.7%.

1 Introduction

Heart disease is the leading cause of death in the United States with identified causes and risks including age, inflammation, high blood pressure, high cholesterol, smoking, and diabetes. It is also a risking healthcare issue, with an estimated 92.1 million Americans with some form of cardiovascular disease in 2015, and . Preventative measures and predictive capabilities are incredibly important in the medical field in order to save lives, increase quality of life, and save on expensive medical procedures and other economic expenses. The federal agency, the CDC, is one of the principle collators of data related to heart disease, of which will be used to perform data exploration and analysis. We will explore the data's authentic information content in order to determine the statistically significant causes for heart disease and to possibly formulate a more effective and efficient health screening to prevent and mitigate heart disease. Particularly, to what extent can survey data predict the risk for heart disease?

The dataset comes from This Kaggle Notebook retrieved from a subsection of the data from the Behavioral Risk Factor Surveillance System (BRFSS) from the CDC from the year 2015 with 253,680 cleaned survey responses. Of the responders, 23,893 have had heart disease.

2 Data Explanation - Variables

Data Set - Heart Disease Health Indicators Data set

The shape of data set is (253680, 22). It contains 253,680 survey responses and 22 variables from the Behavioral Risk Factor Surveillance System collected by the CDC. It contains a large number of observations and moderate number of variables. The first variable 'HeartDiseaseorAttack', a binary response variable, records if the respondent has ever had any heart disease or attack. The rest of the 21 variables are mostly related to respondents' health condition such as 'BMI', 'HIGHBP', 'Diabetes' or life conditions such as 'Smoker', 'Age', 'Education' etc. One thing to be noticed is that the data set is in combination of 14 nominal variables, 7 ordinal variables and 1 numerical variable. Variables like 'Sex' or 'Smoker' are nominal data containing 0 and 1. Ordinal variables such as 'Age' has been

rescaled from 1 to 13 instead of real age. The only numerical variable is the 'BMI'. There are no missing values since the data has already been cleaned.

3 Exploratory Data Analysis

3.1 Correlation

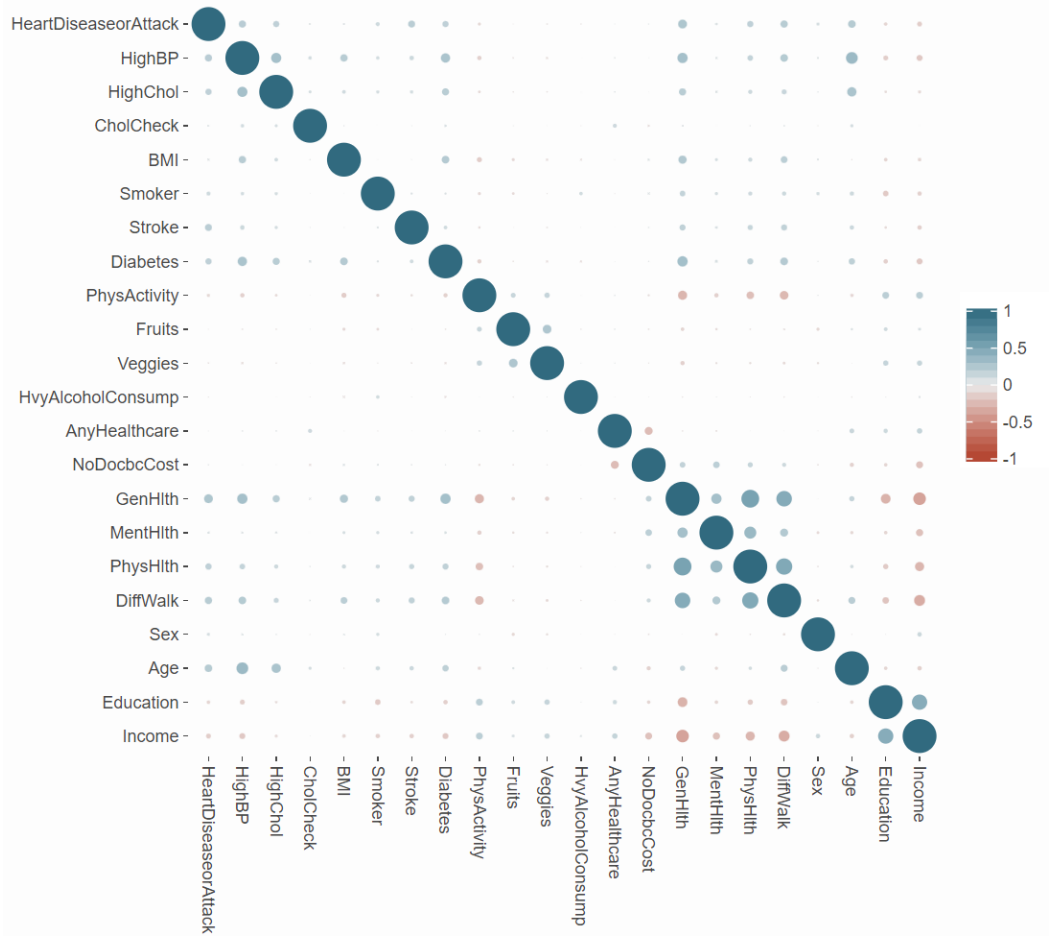

Figure 1: Correlation Plot

A correlation plot shows us the associations between each variable broken down. The diagonal is equal to 1 because each variable is equally correlated to itself. The size of the dot represents the strength of association where blue is positive and red is negative. We see that the strongest positive associations are between general health and physical health and between general health and difficulty walking, which is unsurprising. The strongest negative associations are between income and general health and income and education. In terms of the variable of interest, heart disease, we see that the strongest associations are relatively weak at around +0.2 and very weak negative associations. The strongest positively associated features align with previous research, those being general health, age, previous stroke, difficulty walking, high blood pressure and high cholesterol.

3.2 Histograms

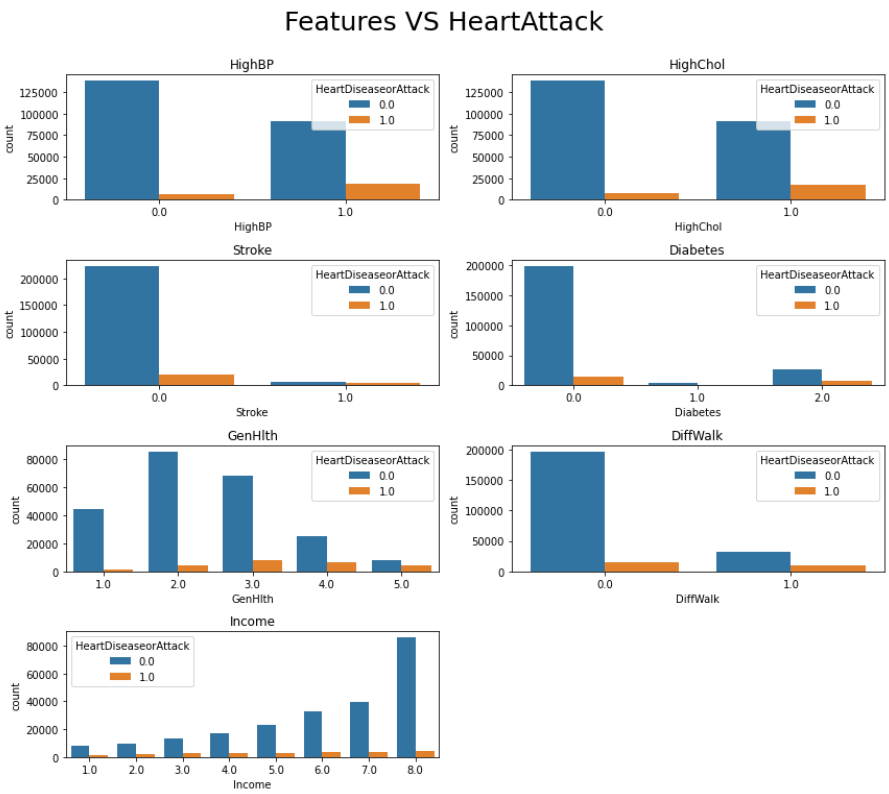

Figure 2: Histogram

The following histogram is the features vs. HeartAttack. Variables such as 'HighBP', 'Stroke' have been grouped by the 'HeartDiseaseorAttack'(0 or 1). The X-axis represents the feature and Y-Axis is the total count of the observations of each variable colored by blue(HeartDiseaseorAttack = 0) and orange(HeartDiseaseorAttack = 1). From the first and second plot, we are likely to see the trend that the count of people who have HighBP and HighChol are more likely to have HeartAttack. Also in the plot GenHlth vs. Heartattck, people with higher GenHlth are more likely to have HeartAttack.

3.3 Heatmaps

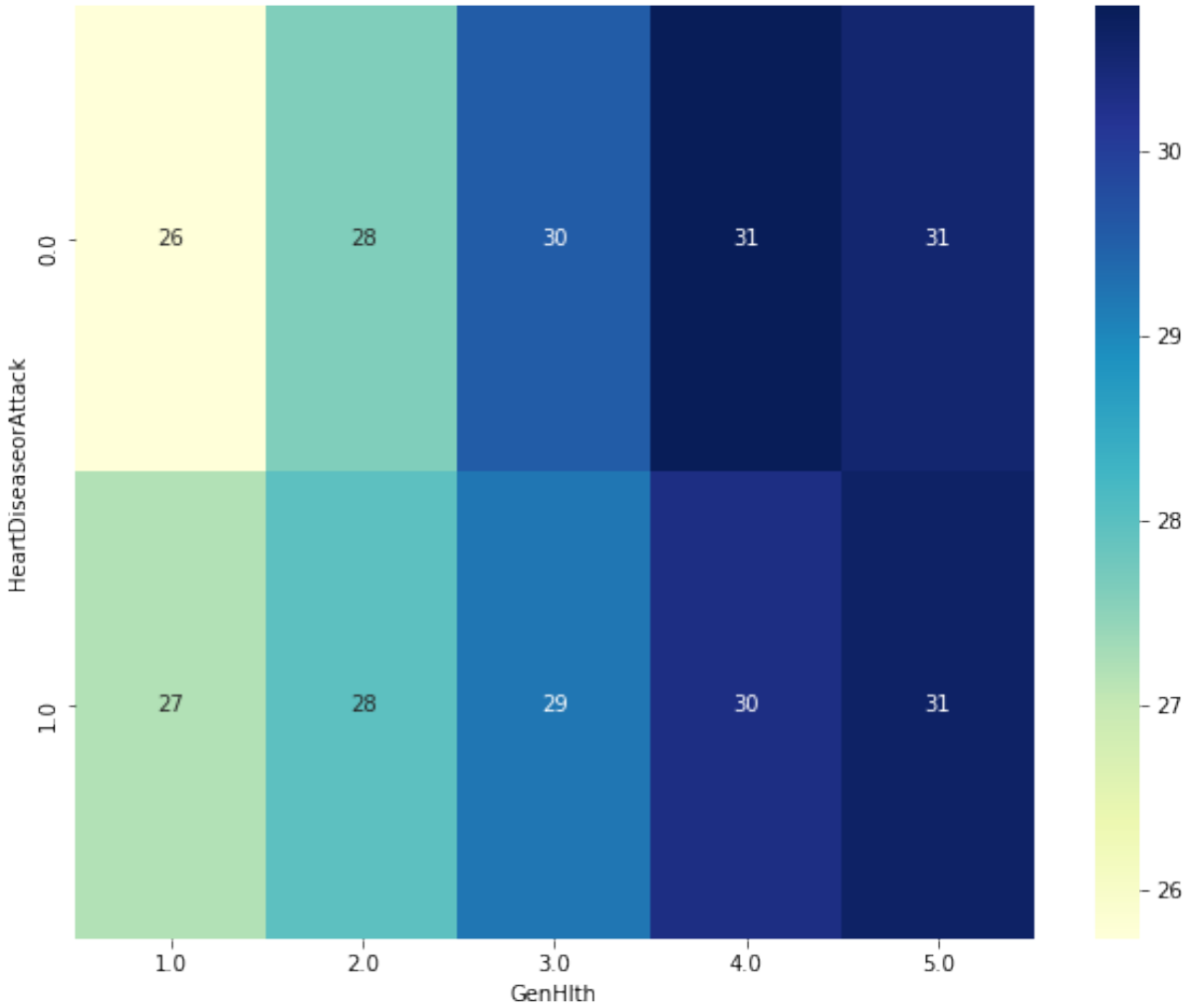

Figure 3: A Heatmap of HeartDiseaseorAttack, GenHlth, and BMI

The heatmap below shows us the magnitude of certain BMI values with the General Health values and whether or not they had a Heart Attack or Disease. The General Health values range from 1 - 5 with 1 meaning Excellent and 5 meaning Poor.

Figure 1 shows us that for patients with heart disease, the rightmost column is the most frequent at a BMI of 31 and at a General Health level of 5. It can be inferred that heart disease is more frequent in individuals with poor General Health and a higher BMI score.

The heatmap also implies that lower occurring frequencies of heart disease are at lower BMI scores and lower General Health scores meaning that the better the general health and the lower the BMI, the lower the frequency of heart disease.

3.4 Pairplots

3.4.1 High Cholestrol

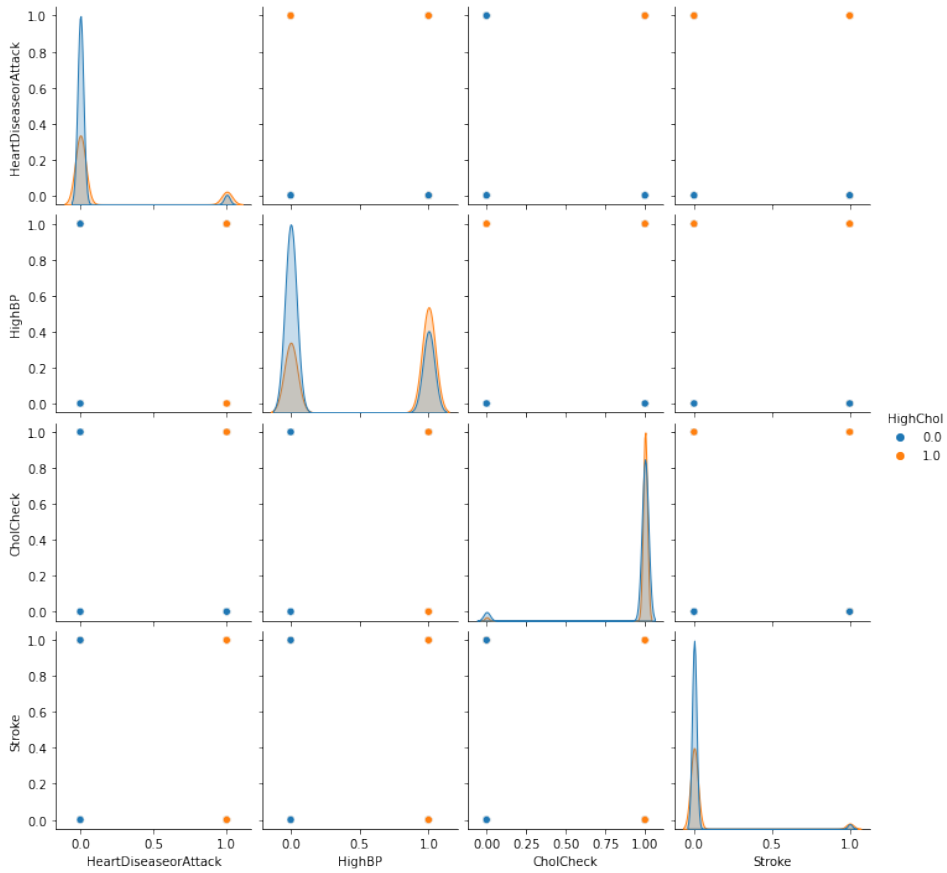

Figure 4: High Cholesterol Pairplot

High cholesterol is a prevalent cause for heart diseases and strokes. Thus, we decided to study this variable using a pair plot to study its effects on these diseases. We chose a pair plot to study its effects since it would let us see the distributions of high cholesterol and also see its relationship with other variables.

The plot shows that people with higher cholesterol have a high risk of having a heart disease. They have a very significantly high chance of getting a heart attack when compared to people that do not have the problem of high cholesterol.

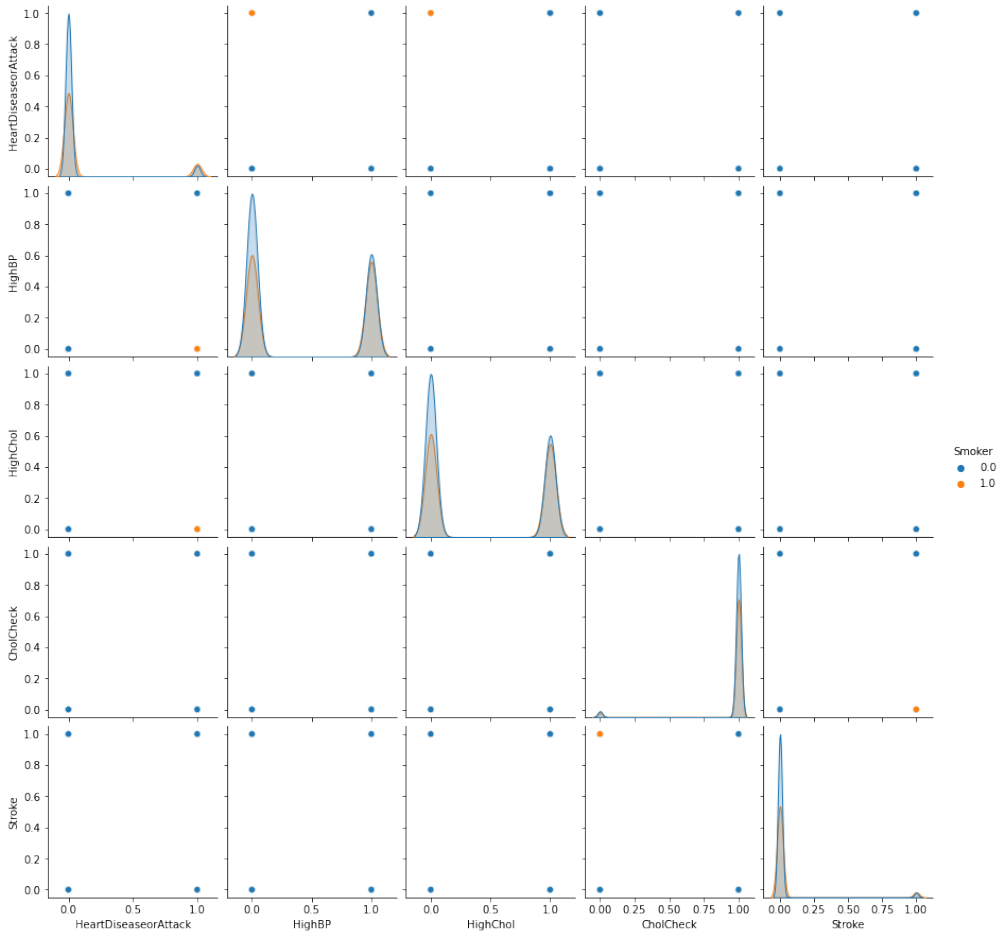

Figure 5: Smoker Pairplot

3.4.2 Smoker

Smoking is a common habit which is also a major cause for heart diseases and strokes. Thus, we decided to study that variable using a pair plot to study its effects on these diseases.

The data represents 0 as smokers and 1 as non-smokers. From the plot we can clearly see that smokers have a high risk and also have a higher frequency of heart attacks and strokes. This could also be interacting with high blood pressure.

3.4.3 High Blood Pressure

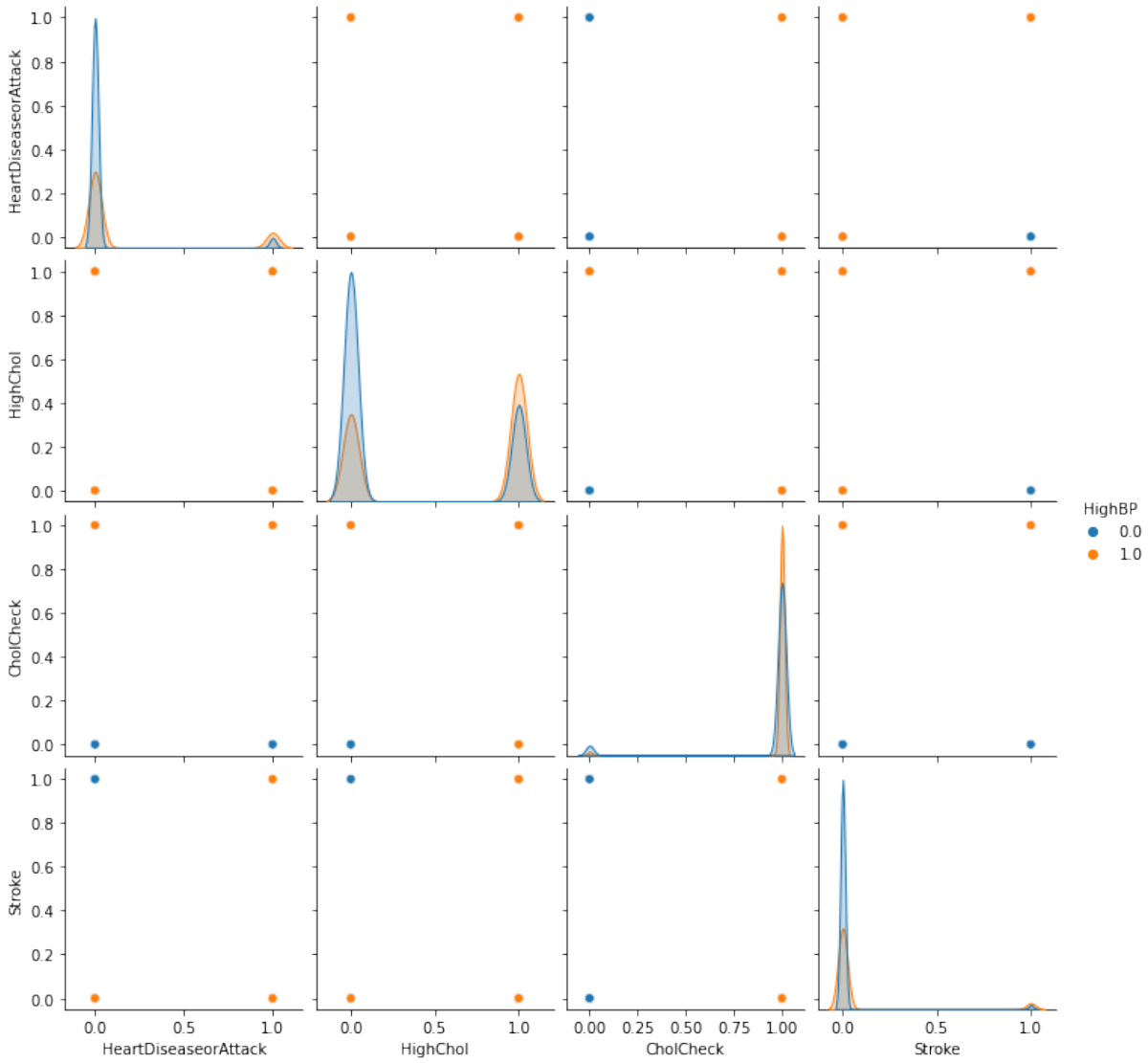

Figure 6: High Blood Pressure Pairplot

High blood pressure has been a very common cause for heart diseases. It is medical condition which can lead to heart disease. We plotted a pair plot to check its relationship with other variables all together.

We can clearly in the third pair plot see that having a high blood pressure can lead to several diseases. This can lead to a stroke as well. This can be helpful for patients to keep in their mind. They should be aware what high blood pressure can lead to.

All the pair plots show that, smoking,high blood pressure and high cholesterol can lead to other diseases.

3.5 Contingency Tables

HighBP	GenHlth	Smoker	No	Yes	Totals
No	Excellent = 1	No	24784	228	25012
		Yes	10932	206	11138
	Fair = 4	No	4561	465	5026
		Yes	5122	879	6001
	Good = 3	No	19283	782	20065
		Yes	15396	1273	16669
	Poor = 5	No	1030	199	1229
		Yes	1935	543	2478
	Very Good = 2	No	34735	640	35375
		Yes	21108	750	21858
Yes	Excellent = 1	No	5069	268	5337
		Yes	3498	314	3812
	Fair = 4	No	7096	1980	9076
		Yes	8063	3404	11467
	Good = 3	No	17333	2301	19634
		Yes	15720	3558	19278
	Poor = 5	No	1972	1038	3010
		Yes	3037	2327	5364
	Very Good = 2	No	16302	1191	17493
		Yes	12811	1547	14358
Totals			229787	23893	253680

HighBP	GenHlth	Smoker	No	Yes	Totals
No	Excellent = 1	No	0.107856	0.009543	0.098597
		Yes	0.047574	0.008622	0.043906
	Fair = 4	No	0.019849	0.019462	0.019812
		Yes	0.022290	0.036789	0.023656
	Good = 3	No	0.083917	0.032729	0.079096
		Yes	0.067001	0.053279	0.065709
	Poor = 5	No	0.004482	0.008329	0.004845
		Yes	0.008421	0.022726	0.009768
	Very Good = 2	No	0.151162	0.026786	0.139447
		Yes	0.091859	0.031390	0.086164
Yes	Excellent = 1	No	0.022060	0.011217	0.021038
		Yes	0.015223	0.013142	0.015027
	Fair = 4	No	0.030881	0.082869	0.035777
		Yes	0.035089	0.142469	0.045203
	Good = 3	No	0.075431	0.096304	0.077397
		Yes	0.068411	0.148914	0.075993
	Poor = 5	No	0.008582	0.043444	0.011865
		Yes	0.013217	0.097393	0.021145
	Very Good = 2	No	0.070944	0.049847	0.068957
		Yes	0.055752	0.064747	0.056599

Figure 7: Contingency table of Heartdiseaseorattack vs. HighBP grouped by general health rating, and smoking status by count and conditional probability

3.5.1 Shannon Entropy

	HighBP	GenHlth	Smoker	HeartDiseaseorAttack
0	0.000000	1.000000	0.000000	0.051897
1	0.000000	1.000000	1.000000	0.092124
2	0.000000	2.000000	0.000000	0.090517
3	0.000000	2.000000	1.000000	0.149427
4	0.000000	3.000000	0.000000	0.164668
5	0.000000	3.000000	1.000000	0.269811
6	0.000000	4.000000	0.000000	0.308327
7	0.000000	4.000000	1.000000	0.416547
8	0.000000	5.000000	0.000000	0.442840
9	0.000000	5.000000	1.000000	0.525803
10	1.000000	1.000000	0.000000	0.199149
11	1.000000	1.000000	1.000000	0.284523
12	1.000000	2.000000	0.000000	0.248655
13	1.000000	2.000000	1.000000	0.341774
14	1.000000	3.000000	0.000000	0.361298
15	1.000000	3.000000	1.000000	0.478242
16	1.000000	4.000000	0.000000	0.524566
17	1.000000	4.000000	1.000000	0.608173
18	1.000000	5.000000	0.000000	0.644200
19	1.000000	5.000000	1.000000	0.684361

Figure 8: Shannon's Entropy of the contingency table

The contingency tables allow us to see nonlinear associations. For example, the above figure shows the frequency of heart attack or disease by groups of those with or without high blood pressure, then among those the groups of general health rating further grouped into those who smoke. One of the main goals in data exploration is to expose the stochastic and deterministic structures[1]. Real life data is often messy and hard to predict as we practically never know the true underlying distribution of our data. The stochasticity or randomness is constrained by how deterministic the data is. Therefore, it may be more authentic to rely on methods that do not apply specific assumptions such as equal variance to expose these structures. A more explicit way of showing these associations is through conditional entropy, using Shannon's entropy which shows us how much information there is in each variable.

Converting to conditional probability, we can see which groups have higher probabilities and more likely to have heart disease. Using Shannon's entropy (natural log), we can see which variables have more information and contribute the most towards the aforementioned stochastic and deterministic structures. The groups that contain the most information are those with poorer perceived general health. Those with high blood pressure have a consistently higher entropy than those with regular/low blood pressure that do not. This is true for smoking, but blood pressure seems to have more information than smoking.

4 Methodology

4.1 Survivorship Bias

Lets establish some general baseline statistics on heart disease within the scope of the United States. Around 25% of deaths are from heart disease or a heart attack. Along with general knowledge on heart disease we know that most people are not alive for long after their first heart complication. Indeed, 62% of stroke patients die within a year. Hospitalization is also extremely expensive, and the vast majority of people who are currently hospitalized are so by necessity. Heart disease is extremely lethal for those who have already had a complication, by the time a person medically knows they have heart disease, hospitals often can only postpone death, but not the cause. In 2015, Given this, What can we infer firstly about the data?

Given in our 2015 data-set only 9.42% of people have had heart disease, around 25% of American deaths are from heart disease, and a low estimate of 48.2% of Americans had some form of heart disease in 2015 will die of heart disease. We can conclude that the dataset's survivorship bias is very extreme.

4.2 What Makes a Model Good?

Statistical modeling requires context. Different models and different distributions are created and used to model real world phenomena, and take the gathered data to predict outcomes. Context determines the purpose of statistical work. Given that the response variable is whether a person has had a heart disease or a heart attack plus 21 health predictors, one of many utilities the data can provide is preventative intervention for people who are at risk of heart disease. Choosing that as the context, helping hospitals and biopharmaceutical companies monitor those at risk or provide preventative intervention, model criterion can be determined.

In the field of healthcare, the sensitivity and specificity of a test are key factors in determining the efficacy of a test or treatment. Realistically, these two metrics are conditional on an arbituary cutoff value. In the context of a A.I. model that predicts whether or how likely a person will have a heart disease, a higher sensitivity indicates people who will actually have heart disease are less likely to be labeled as those who will not. A higher specificity indicates that people who will actually never have heart disease are less likely to be labeled as someone in danger of heart disease. What the model will label someone as, once again, depends on the cutoff, which depends on the hospital or company interested. If doctors can afford to see more people at risk, then the percentage of the population that is identified as under the threat of heart disease can be increased by increasing the cutoff value.

4.3 Choosing a Model Metric: Sensitivity

Sensitivity will be our model evaluation metric. For one, only 9.42% of the population within the dataset has heart disease. Furthermore, a more sensitive model indicates a higher percentage of patients identified are at risk, which increases the efficacy a hospital's doctors can deliver preventative measures. It should be noted that the sensitivity and specificity for a given cutoff and model are generally a trade off of each other. In this case, if the model evaluation metric was chosen to be specificity instead, then a good model would make less people worry about being at risk of having a heart attack.

4.4 Assumptions

There are some relatively simple heuristics one can apply to figure out that the dataset is most likely not a hollistically random sample. For one, people who had a heart attack and died would not able to take the survey. This leads to survivorship bias in the data, especially given that there is no time component to apply some sort of correction for forward modelling. Furthermore, the data has had all the N/A's removed.

So the assumptions we operate under for the methodology section is as follows.

1. The data represents a random sample of the United States population
2. The time frame required to acquire the data did not change the likelihoods, joint-likelihoods, or distribution for the variables

3. The correspondents provided honest responses to the best of their ability

4.5 Feature Selection: All Features

The lack of correlation in a variable does not mean that a combination of variables will not reveal what is actually going on. The dataset shows that heavy alcohol consumption has little to no correlation with the dataset's three health general metrics and no correlation with high cholesterol. So should heavy drinking be dropped during feature selection? No. Perhaps people with a higher income are more likely to go drink at a bar, and can afford better healthcare. Perhaps the respondents who are heavy drinkers are also less likely to have body dysmorphia or negative self-images. Or perhaps heavy drinking really doesn't have much of an affect on people. But correlation does not imply causation, and the reasons behind observed human phenomena is not for individual statisticians to speculate. This is something to consider especially when working on data on people.

In practicality however, feature selection and feature engineering are often a necessity when gathering the data itself and determining the scope of a model that uses real world data. Statisticians must often use heuristics to make tradeoffs in areas such as model complexity, training time, and the model itself. For example, finding the optimal number of layers and layer sizes for a neural network is generally not a tracktable problem.

Since the variables in the Kaggle dataset have already been preselected and preengineered (continuous variables such as BMI and Age have been ordinally encoded into categories), dropping variables or further smoothing of previously continuous preengineered data to reduce entropy within the variables themselves will almost certainly increase the entropy of the response variable conditional on the information we have.

Take for example, the Shannon's Entropy of Y given the 87 values of BMI, the $4th$ predictor

$$H[Y|X_4 = x_4], x_4 \in \{1, 2, ..., 87\}$$

and a smoothed vector X_4^* for BMI that puts BMI into 5 categories

$$H[Y|X_4^* = x_4^*], x_4^* \in \{1, 2, ..., 5\}$$

The latter embraces more information.

4.6 Machine Learning Classification: Trees

Of course, statisticians can still drop features and smooth data, especially when there is a small sample size. Given that we have 253,680 observations with 0 's NA's, and binary categorical (boolean) predictors, it is clear that a nonparametric model is more suited for modeling Heart Disease than a parametric model such as linear regression. A binary decision tree is typically great for modeling categorical real world phenomena, and by creating a cutoff for variables with more than one category, predictors such as Age and BMI in our dataset can be used to create binary splits as well.

A binary decision tree recursively partitions observations in the response variable based off values in the predictor variables. Each split aims to increase the purity of the subsegments in the roots of the tree. For this dataset, the maximum depth any tree can have is the number of predictor variables, p, and the maximum number of roots (given unlimited data points) is 2^p.

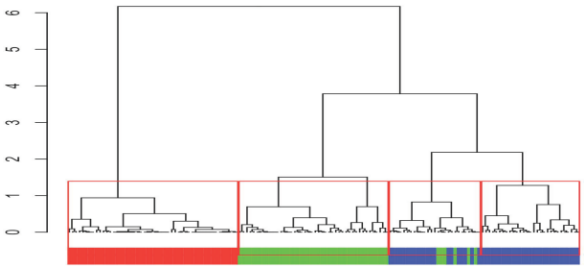

Figure 9: Example of a Top-Down Tree

Therefore, under the context of our given data and goal, we define statistical modeling with a generalized decision tree on categorical variables as, the nonparametric categorization of subsegments s in the response variable Y, conditional on the observed binary (boolean $\{0|1\}$) value c in the observation's predictor $X_j, j = 1, 2, ..., p$, or the boolean outcome c of meeting an arbitrary cuttoff value for predictors with more than one value.

With n observations with p predictors for each observation,

$$\begin{aligned} Y &= (Y_1, Y_2, ..., Y_n)' \\ X &= (X_1, X_2, ..., X_p) \\ X_j &= (c_{1j}, c_{2j}, ..., c_{nj})' \\ & c \in \{0, 1\} \end{aligned}$$

The information gain is then the difference in entropy before and after the split. With a base-2 log in this case for our binary splits, the Shannon's Entropy statistic is an appropriate measure of entropy. Decision tree implementations such as a greedy decision tree, which sequentially picks the split with the most Information Gain each time, have a tendency to "overfit" by picking suboptimal splits in the beginning that stunt them later on.

An assumption inherent to using a single sequential decision tree as per our definition, is that the predictors affect each other sequentially. If a split for some predictor has already been made, the predictor can not be used again. The standard decision tree lacks the degrees of freedom and backpropagation correction capabilities that neural networks have, but neural networks come with their own concerns.

Better results can be obtained using a decision tree obtained by random forest (where multiple decision trees start with randomized splits), or a bag of random forest trees (or random forest ensemble with more trees, each as weak predictors). These implementations are more robust to overfitting, especially with the large sample size. Which is why the model we determine to be most optimal for the data is a gradient boosted random forest ensemble, to eliminate relying on any individual sequence of the predictors.

4.7 Gradient Boosted Regressor

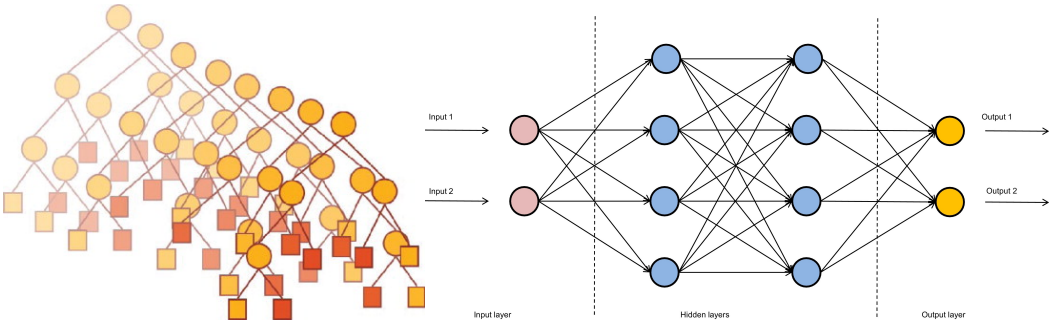

Figure 10: Visual Representation of a Random Forest[4] and a NN[5]

The above implementations, whether it be a bag of randomized decision trees or a neural network, rely on a similar Bayesian concept of using the theoretical information gain from conditional probabilities (or hyperparameters) to create subsegments, or lattices, or tensors. They create arbitrary bounds to fit an activation function such as the logistic activation function popular in hierarchical clustering algorithms.

While linear models aim to predict,

$$Y|(X_1, X_2, ..., X_p)$$

The models of our interest are more akin to,

$$(((Y|X_1)|X_2)|X_3)...|X_p)$$

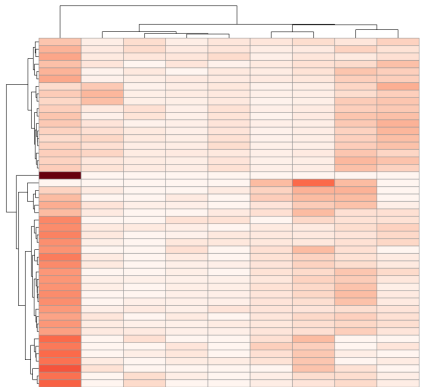

Figure 11: A Heatmap (Fushing)

Gradient boosting is an optimization method that prevents overfitting by bootstrapping data and repeatedly testing weak learners on the data so that extreme errors in population segments are less likely. An ensemble of gradient boosted random forest models will allow for multiple different smaller forests to vote on an outcome. These conferred advantages are why we believe that the highest performing model will be a Gradient Boosted Random Forest, used as a regressor for cutoff flexibility.

Ultimately, these models are not so much that different from a heatmap with an output layer, such as the one illustrated in Figure 11 above. The lattices created in Figure 11 categorize observations an a similar manner as two fitted trees from a random forest ensemble would.

5 Results

5.1 Comparing the Performance Between Various Models

All of the models used on our dataset were run on python3 and imported from SkicitLearn, which uses Scipy and NumPy, which use solvers from BLAS and LAPACK for linear algebra. Each model used was gridsearched, and would be further gridsearched according to their theoretical potential to improve with better hyperparameters, observed improvement after initial manual hyperparameter tuning, the degrees of freedom in their hyperparameters, and the training time required to retrain them.

Preliminary models such as Linear Regression, Ridge Regression, Lasso Regression, and Support Vector Machine were used to establish a baseline. Lasso with ordinally encoded data performed the best with alpha=0.000001 and decreasing alpha further no longer yields greater performance. It does continue to improve slightly, but trains exponentially slower (whilst exponentially decreasing alpha) using one hot encoded X. Popular penalty statistics were tested during their initial grid searches.

Models that did categorical predictions, such as Logistic Regression would have a high accuracy, but very low specificity. We obtained the raw probability predictions for models and converted them to binary again for different cutoffs, with Logistic Regression performing very well. The logistic activation functions on neural networks and decision trees also outperforms the relu function. The Multi-Layer Perceptron (MLP) regressor was used in lieu of converting the predicted probabilities of a categorical MLP back to the cutoffs. A total of around 20 neural networks with 3 to 4 hidden layers, each of sizes up to 21^2, were trained before finding one that we could not further optimize. The highest performing neural network is displayed. The optimized MLP's took around 3 hours each to train with adaptive learning rates, so there may be room for more optimization (but they seem to be overfitting). A total of 4 gradient boosted random forest ensembles were trained before finding an Optimized model, each taking around 30 minutes.

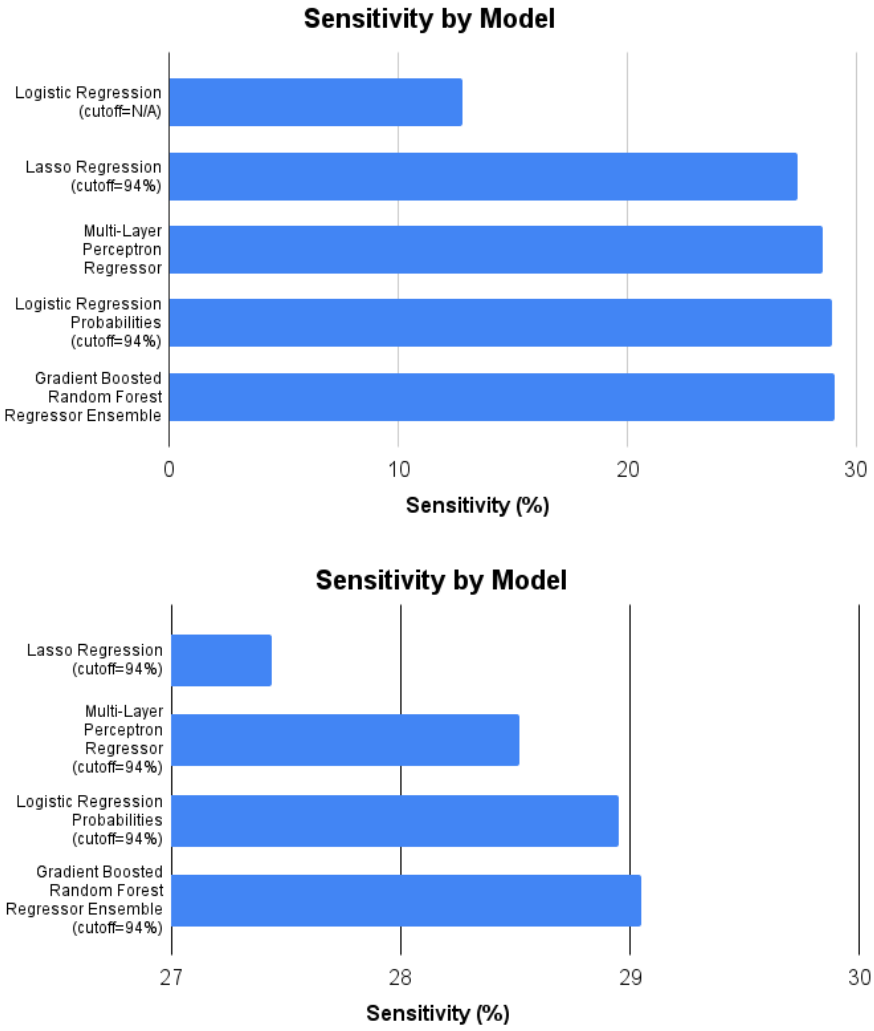

Figure 12: Model Metrics

The model sensitivities for high performing models are below, with categorical Logistic Regression (no cutoff) as a control for the low specificity observed when model predictions are not set to a cutoff in population. Model performance at cutoffs where 94%, 90%, and 85% of the population is assumed to be healthy are show above.

5.2 The effect of Cutoff

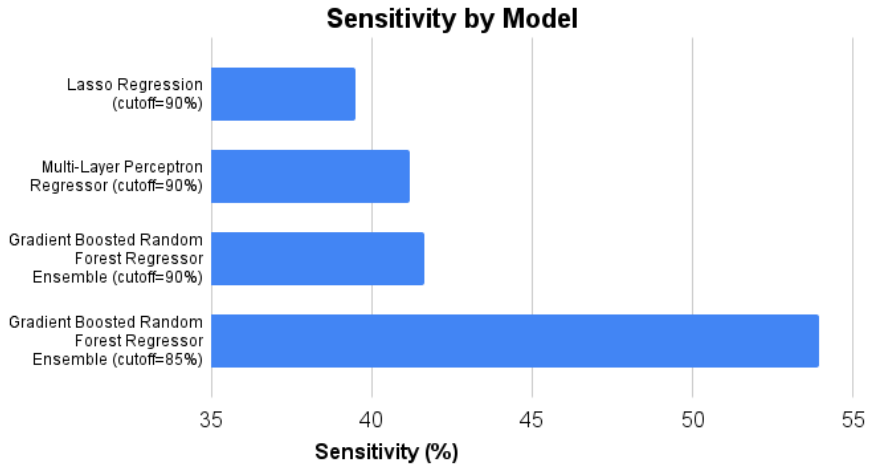

Figure 13: Model Metrics

All of the unique models in Figure 12 are the highest performing gridsearched models we found, the same Gradient Boosted Random Forest Regressor Ensemble is used for all of the cutoff levels. The third bar chart in Figure 12 includes the Gradient Boosted Random Forest Regressor Ensemble (GBRFR_N460_MD3), with 460 initial estimators and a maximum depth of 3 predicting at a cutoff of 85% to show how large of a role cutoff plays. This particular model is the only one in the entire Figure that has more True Positives than False Negatives, with their values being 3865 and 3303 respectively. At 85% cutoff, the accuracy is 85.74% and at higher cutoff values around 94%, the accuracy increases to over 90% at the cost of sensitivity.

5.3 Heart Disease and GBRFR Model Cutoffs

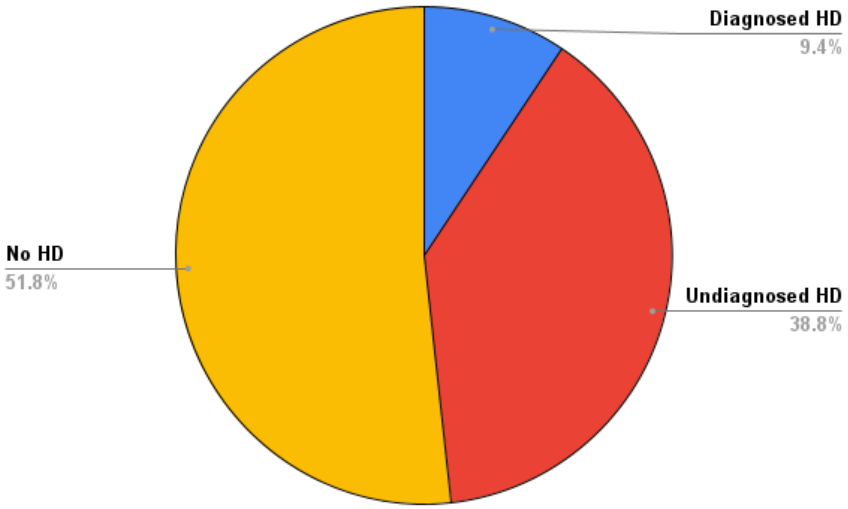

Figure 14: American Adults and Heart Disease (HD) [7]

Around 25% of deaths in 2015 were from heart disease or a heart attack. Along with general knowledge on heart disease we know that most people are not alive for long after their first heart complication. Indeed, 62% of stroke patients die within a year. Hospitalization is also extremely expensive, and the vast majority of people who are currently hospitalized are so by necessity. Heart disease is extremely lethal for those who have already had a complication, by the time a person medically knows they have heart disease, hospitals often can only postpone death, but not the cause.

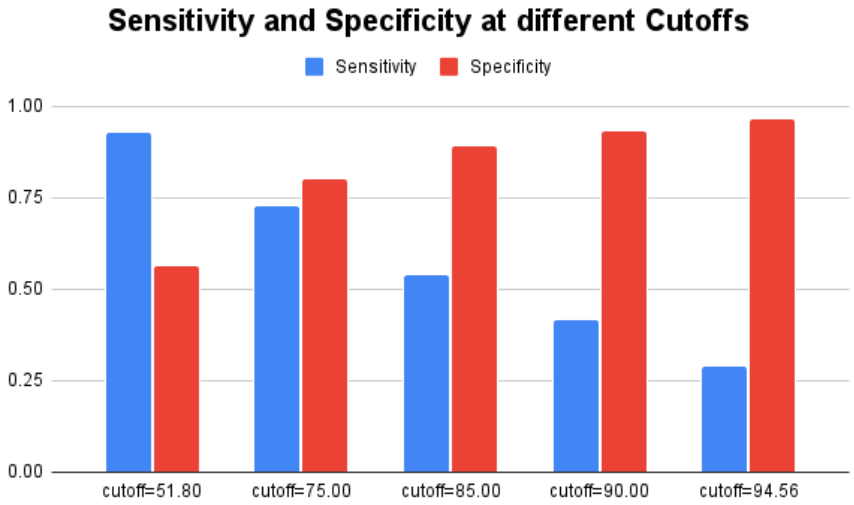

Figure 15: Comparison of Cutoffs (% of data assumed to have no heart disease)

It is certain that the GBRFR_N460_MD3's False Positives at all the cutoffs in figure 15 contain people who will have heart disease one day. It is possible that the majority of them will have heart disease. Most Americans diagnosed with heart disease, have had a heart complication [7]. The first cutoff (51.80) in figure 15 corresponds to a low estimate of 48.2% of Americans in 2015 having some form of heart disease at a given time. Since 9.42% of people in the our data set (Americans in 2015) had a been diagnosed with heart disease, with 10% as high estimate, the corresponding cuttoff is 90 (The fourth cutoff).

We know that at around an 85% cutoff the TP eclipsed FN counts balance out when the model is predicting 15% of people have had heart disease, the model is correct around half the time. The other half the time predicts someone will have heart disease. While only $2/5$ths of the people in the data has experienced heart disease, when the model has a cutoff is $3/5$ths of (the expected quantity of) people who have had or will have heart disease, then the TP is already much higher than FN. This indicates that the model is sensitive to people who already have had heart disease, but even more crucially, people who might have heart disease in the future.

6 Conclusion

Studying real world data does not allow for many assumptions of traditional parametric statistical theory. The question is seldom performing a linear regression and obtain a straightforward answer via hypothesis testing and p-values. Exploring the structure of the data shows us that it would be inauthentic to drop features but a practical necessity for scenarios with large amounts of data and minimal information loss from feature selection. Our EDA findings agree with previously established research that features such as high blood pressure, BMI, and cholesterol have the strongest associations and provide the largest information gain. Tradeoffs must be made, and models must be adapted to the data. Models such as logistic regression fit the data very well but have extremely low specificity, so they raw probabilities must be used. The model with the highest specificity, the Gradient Boosted Random Forest Ensemble was very robust to overfitting and time efficient. Amongst the stronger models, it was the most hyperparameter consistent and predictively versatile, while also requiring fewer new assumptions for our data. The survey data is a very useful tool, but it

is important to acknowledge that such data is limited in how much information it can determine for the fate of people. Hospitals can use this data to identify people who are potentially at risk of a heart attack, and identify geographical regions that require more medical facilitation in the future.

7 References

[1] Fushing, H. (2022). CEDA with mimicking for Data Analysis. *Lecture-0: More is Multiscale Different.* [PowerPoint slides]. Statistics Department, University of California, Davis. Link

[2] Pandas Development Team (2008). Table Visualization. Link

[3] Zinno, G. (2020). How to compute Shannon entropy of Information from a Pandas Dataframe? Link

[4] Science Direct. (2018). Random Forest. Link

[5] Ahmad, M. W., Mourshed, M., Rezgui, Y. (2017). Trees vs Neurons: Comparison between random forest and ANN for high-resolution prediction of building energy consumption, Energy and Buildings. Link

[6] FARID RIZQI S. (2021). HeartDisease EDA+Prediction. Link

[7] Emelia J. Benjamin, Paul Muntner, Alvaro Alonso, + 49 More. (2019, January 31). Heart disease and stroke statistics—2019 update: . American Heart Association. Link

8 Appendix

Listing 1: Raw Code

```
import math
import random
import seaborn as sns
import sklearn.metrics
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

from timeit import default_timer
from numpy import unique
from numpy import sqrt, log, dot, array, diagonal, mean, transpose, eye, diag, ones
from numpy import transpose, diag, dot
from numpy.linalg import svd, inv, qr

from scipy.stats import entropy
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split

from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression, LinearRegression
from sklearn.linear_model import Lasso, Ridge
from sklearn.svm import LinearSVC, LinearSVR
from sklearn.preprocessing import OneHotEncoder
from sklearn.neural_network import MLPRegressor, MLPClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingRegressor
from sklearn.ensemble import GradientBoostingRegressor, RandomForestRegressor

random.seed(1)

df = pd.read_csv("heart_disease_health_indicators_BRFSS2015.csv")

colNames = df.columns

dfX = df.copy().drop(columns=["HeartDiseaseorAttack"])

dfY = df["HeartDiseaseorAttack"]

X = array(dfX)
```

```
Y = array(dfY).reshape(len(dfY), 1)

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=1)

encoder = OneHotEncoder()

dfX_enc = refactorColsEncode()
X_enc = array(dfX_enc)

X_enc1 = encoder.fit(X).transform(X).toarray()
X_enc2 = encoder.fit(X_enc).transform(X_enc).toarray()

def check(vec):
    if type(vec[0]) not in (int, float, np.float64):
        return np.ravel(vec)
    return vec

def accuracy(Y_pr, Y_ts):
    Y_pr, Y_ts = check(Y_pr), check(Y_ts)
    return sum([1 if pr == ts else 0 for pr,ts in zip(Y_pr, Y_ts)]) / len(Y_pr)

def confusion(Y_pr, Y_ts):
    Y_pr, Y_ts = check(Y_pr), check(Y_ts)
    TP, FP, TN, FN = 0, 0, 0, 0
    dd = {(1,1):TP, (1,0):FP, (0,1):FN, (0,0):TN}
    for pr, ts in zip(Y_pr, Y_ts):
        dd[(pr,ts)] += 1
    return {"TP": dd[(1,1)], "FP": dd[(1,0)], "FN": dd[(0,1)], "TN": dd[(0,0)]}

def convertToBinary(Y_pr, cuttoffPercentile=85):
    cutoff = np.percentile(Y_pr, cuttoffPercentile)
    return [1.0 if pr > cutoff else 0.0 for pr in Y_pr]

def runModel(model, __X, __Y, cuttoffPercentile=85):
    __X_train, __X_test, __Y_train, __Y_test = train_test_split(__X, __Y,
                                                test_size=0.30, random_state=1)
    model.fit(__X_train, np.ravel(__Y_train))
    modelPred = model.predict(__X_test)
    if any([0.000001 < pr < 0.99999 or pr < -0.000001 for pr in modelPred]):
        modelPred = convertToBinary(modelPred, cuttoffPercentile)
    print("Accuracy:", accuracy(modelPred, __Y_test))
    print("MSE:", mean_squared_error(modelPred, __Y_test))
    print("Confusion:", confusion(modelPred, __Y_test))
    return model

def getCategories(X):
    return {j: list(unique(X[:, j])) for j in range(len(X[0]))}

def shannonsEntropy0(column):
    if column[0] != int:
        column = [int(obs) for obs in column]
    HX = 0.0
    Categories = list(unique(column))
    for cat in Categories:
        px = column.count(cat) / len(column)
        HX += px * log(px, 2)
    return -HX

def shannonsEntropy(col):
    if type(col) != int:
        return shannonsEntropy0(col)
    HX = 0
    column = [int(obs) for obs in list(X[:, col])]
    for cat in categories[col]:
```

```
            nobs = column.count(cat)
            px = nobs / len(column)
            HX += px * log(px, 2)
        return -HX

    def shannons(categories, X):
        scores = {}
        for col in categories:
            categories[col] = [int(obs) for obs in categories[col]]
        for col in categories:
            scores[col] = shannonsEntropy(col)
        return scores

    cell_hover = {
        'selector': 'td:hover',
        'props': [('background-color', '#ffffb3')]
    }
    index_names = {
        'selector': '.index_name',
        'props': 'font-style: bold; color: black; font-weight: normal;'
    }
    headers = {
        'selector': 'th:not(.index_name)',
        'props': 'background-color: #000066; color: white;'
    }

    a = {0.0: 'No', 1.0:'Yes'}
    b = {1.0:"Excellent = 1", 2.0:"Very Good = 2", 3.0:"Good = 3", 4.0:"Fair = 4", 5.0:"Poor = 5"}

    dat2_crosstab = pd.crosstab([df.HighBP.replace(a), df.GenHlth.replace(b), df.Smoker.replace(a)

    cm = sns.light_palette("green", as_cmap=True)
    mytable_dat2 = (dat2_crosstab.style
      .background_gradient(cmap=cm)
      .highlight_max(axis=1, props='color:black; font-weight:bold;')
      .set_caption('Contingency Table')
      .set_table_styles([cell_hover, index_names, headers]))

    dat4_crosstab = pd.crosstab([df.HighBP.replace(a), df.GenHlth.replace(b), df.Smoker.replace(a)

    mytable4 = (dat4_crosstab.style
      .background_gradient(cmap=cm)
      .highlight_max(axis=1, props='color:black; font-weight:bold;')
      .set_table_styles([cell_hover, index_names, headers]))

    E = df.groupby(['HighBP','GenHlth','Smoker'])['HeartDiseaseorAttack'].apply(lambda x : entropy

    mytable5 = (E.style
      .background_gradient(cmap=cm, subset = 'HeartDiseaseorAttack')
      .set_table_styles([cell_hover, index_names, headers]))

    x_col = ['', 'HighBP', 'HighChol','Stroke', 'Diabetes', 'GenHlth',
           'DiffWalk',
           'Income']
    plt.figure(figsize=(12,20))
    for i,column in enumerate(x_col[1:]):
        plt.subplot(len(x_col),  2, i+1)
        plt.suptitle("Features VS HeartAttack", fontsize=25, x=0.5, y=1)
        sns.countplot(data=df, x=column, hue='HeartDiseaseorAttack')
        plt.title(f"{column}")
        plt.tight_layout()

    df = pd.read_csv("heart_disease_health_indicators_BRFSS2015.csv")

    dfIncEdu = df[["Education", "Income", "BMI"]]
```

```
dfIncEdu = dfIncEdu.pivot_table("BMI", "Education", "Income")
dfIncEdu

plt.figure(figsize=(10, 8))
ax = sns.heatmap(dfIncEdu, cmap="YlGnBu", annot=True)

dfHG = df[["HeartDiseaseorAttack", "GenHlth", "BMI"]]
dfHG = dfHG.pivot_table("BMI", "HeartDiseaseorAttack", "GenHlth")
dfHG

plt.figure(figsize=(10, 8))
ax = sns.heatmap(dfHG, cmap="YlGnBu", annot=True)

dfMentPhys = df[["MentHlth", "PhysHlth", "BMI"]]
dfMentPhys = dfMentPhys.pivot_table("BMI", "MentHlth", "PhysHlth")
dfMentPhys

plt.figure(figsize=(10, 8))
ax = sns.heatmap(dfMentPhys, cmap="YlGnBu", annot=True)

ax = sns.boxplot(x = "HeartDiseaseorAttack", y = "BMI", data = df)

ax = sns.boxplot(x = "HighBP", y = "BMI", data = df)

ax = sns.boxplot(x = "HighChol", y = "BMI", data = df)

data=pd.read_csv("heart_disease_health_indicators_BRFSS2015.csv")
data.dropna()

sns.pairplot(data)

age_30 = data[data["Age"] > 30]

sns.pairplot(age_30)

data.describe()

data_crosstab = pd.crosstab(data['HeartDiseaseorAttack'],
                            data['Age'],
                               margins = False)

data_crosstab1 = pd.crosstab(data['HeartDiseaseorAttack'],
                            data['Sex'],
                               margins = False)

data_crosstab2 = pd.crosstab(data['HeartDiseaseorAttack'],
                            data['MentHlth'],
                               margins = False)

data_crosstab4 = pd.crosstab(data['HeartDiseaseorAttack'],
                            data['Smoker'],
                               margins = False)

df=data[["HeartDiseaseorAttack","HighBP","HighChol","CholCheck","Stroke","Smoker"]]

sns.pairplot(df,hue='Smoker')

df1=data[["HeartDiseaseorAttack","HighChol","HighBP","CholCheck","Stroke"]]

sns.pairplot(df1,hue="HighChol")

df2=data[["HeartDiseaseorAttack","HighChol","HighBP","CholCheck","Stroke"]]

sns.pairplot(df2,hue="HighBP")
```

Listing 2: Raw Code

```
# RAW CODE

skLogiReg = LogisticRegression(max_iter = 800)
runModel(skLogiReg, X, Y)

skLogiReg = LogisticRegression(max_iter = 2000)
runModel(skLogiReg, X, Y)

runModel(Lasso(alpha = 0.0001, max_iter = 1000), X, Y)

runModel(Ridge(alpha = 0.001, max_iter = 1000), X, Y)

runModel(Ridge(alpha = 0.001, max_iter = 1200), X_enc1, Y)

GBR = GradientBoostingRegressor(n_estimators=150, random_state=1)

models_gen1 = [GradientBoostingRegressor(random_state=1),
               RandomForestRegressor(random_state=1),
               LinearSVR(max_iter=2500, random_state=1)]

for __model in models_gen1:
    print(f"{__model}:")
    runModel(__model, X_enc1, Y)

"""
GradientBoostingRegressor(random_state=1):
Accuracy: 0.8568012193840009
MSE: 0.14319878061599917
Confusion: {'TP': 3843, 'FP': 7573, 'FN': 3325, 'TN': 61363}
RandomForestRegressor(random_state=1):
Accuracy: 0.8453563544623147
MSE: 0.15464364553768528
Confusion: {'TP': 3203, 'FP': 7804, 'FN': 3965, 'TN': 61132}
LinearSVR(max_iter=2500, random_state=1):
Accuracy: 0.0
MSE: 0.09418678360317678
"""
start = default_timer()

models_gen2 = [GradientBoostingRegressor(n_estimators=150, random_state=1),
               GradientBoostingRegressor(n_estimators=200, random_state=1),
               GradientBoostingRegressor(n_estimators=150, max_depth=4, random_state=1),
               MLPRegressor(max_iter=800, random_state=1)]

for __model2 in models_gen2:
    print(f"{__model2}:")
    runModel(__model2, X_enc1, Y)
    print("")

stop = default_timer()
print(f"{stop-start} seconds")

from sklearn.ensemble import AdaBoostRegressor, GradientBoostingClassifier

start= default_timer()

GBC = GradientBoostingClassifier(n_estimators=360, max_depth=4, random_state=1)
ABR = AdaBoostRegressor(n_estimators=250, random_state=1)

"""
The Best One:
"""
GBR_NE360_MD4 = GradientBoostingRegressor(n_estimators=360,
```

```
                                        max_depth=4, random_state=1)

print("Gradient␣Boosted␣Classifier,␣for␣accuracy:")
runModel(GBC, X_enc1, Y)

print("\nAdaBoostRegressor:")
runModel(ABR, X_enc1, Y)

print("\nGradient␣Boosted␣Regressor,␣to␣maximize␣TP/FN:")
runModel(GBR_NE360_MD4, X_enc1, Y)

stop = default_timer()
print(f"{stop-start}␣seconds")

def runModelGBR(model, __X, __Y, cuttoffPercentile=85):
    __X_train, __X_test, __Y_train, __Y_test = train_test_split(__X, __Y, test_size=0.30,
                                                                random_state=1)
    model.fit(__X_train, np.ravel(__Y_train))
    modelPredOG = model.predict(__X_test)
    for cutoff in (85, 90, 94):
        modelPred = convertToBinary(modelPredOG, cutoff)
        print("Accuracy:", accuracy(modelPred, __Y_test))
        print("MSE:", mean_squared_error(modelPred, __Y_test))
        print("Confusion:", confusion(modelPred, __Y_test))

start= default_timer()

GBR_NE460_MD2 = GradientBoostingRegressor(n_estimators=450, max_depth=2, random_state=1)
GBR_NE460_MD3 = GradientBoostingRegressor(n_estimators=450, max_depth=3, random_state=1)
GBR_NE460_MD4 = GradientBoostingRegressor(n_estimators=450, max_depth=4, random_state=1)

print("GBR_NE460_MD2:")
runModelGBR(GBR_NE460_MD2, X_enc1, Y)

print("\nGBR_NE460_MD3:")
runModelGBR(GBR_NE460_MD3, X_enc1, Y)

print("\nGBR_NE460_MD4:")
runModelGBR(GBR_NE460_MD4, X_enc1, Y)

stop = default_timer()
print(f"{stop-start}␣seconds")


def runModelGBRTT(model, __X, __Y, cuttoffPercentile=85):
    __X_train, __X_test, __Y_train, __Y_test = train_test_split(__X, __Y,
                                                    test_size=0.30, random_state=1)
    modelPredOG = model.predict(__X_test)
    for cutoff in (85, 90, 94):
        modelPred = convertToBinary(modelPredOG, cutoff)
        print("Accuracy:", accuracy(modelPred, __Y_test))
        print("MSE:", mean_squared_error(modelPred, __Y_test))
        print("Confusion:", confusion(modelPred, __Y_test))

runModelGBRTT(GBR_NE460_MD2,X_enc1,Y)

for cutoff in (85, 90, 94):
    modelPred = convertToBinary(modelPredOG, cutoff)
    print("Accuracy:", accuracy(modelPred, __Y_test))
    print("MSE:", mean_squared_error(modelPred, __Y_test))
    print("Confusion:", confusion(modelPred, __Y_test))

runModel(RandomForestClassifier(), X, Y)

runModel(Ridge(alpha = 0.001, max_iter = 2000), X, Y)
```

```
runModel(Ridge(alpha = 0.001, max_iter = 2000), X_enc1, Y)

m13 = MLPClassifier(max_iter=400, random_state=1)
runModel(m13, X_enc2, Y)
#Accuracy: 0.905813097866078
#MSE: 0.094186902133922
#Confusion: {'TP': 641, 'FP': 641, 'FN': 6527, 'TN': 68295}

m11 = MLPClassifier(max_iter=400, random_state=1)
runModel(m11, X, Y)
#Accuracy: 0.9087301587301587
#MSE: 0.09126984126984126
#Confusion: {'TP': 1008, 'FP': 786, 'FN': 6160, 'TN': 68150}

m12 = MLPClassifier(max_iter=400, random_state=1)
runModel(m12, X_enc1, Y)
#Accuracy: 0.8895327446651949
#MSE: 0.110467255334805
#Confusion: {'TP': 1450, 'FP': 2689, 'FN': 5718, 'TN': 66247}

#### Compare Winners of m11-m13 with different activation functions
#Accuracy: 0.9076264059707768
#MSE: 0.09237359402922317
#Confusion: {'TP': 673, 'FP': 535, 'FN': 6495, 'TN': 68401}

m12_800 = MLPClassifier(max_iter=800, random_state=1)
runModel(m12_800, X_enc1, Y)

#### m21 = MLPRegressor(max_iter=800) beats everything so far by FAR

m21 = MLPRegressor(max_iter=800)
runModel(m21, X_enc1, Y)
#Accuracy: 0.8740802060338484
#MSE: 0.12591979396615158
#Confusion: {'TP': 2598, 'FP': 5013, 'FN': 4570, 'TN': 63923}

m2_1600 = MLPRegressor(max_iter=1600)
runModel(m2_1600, X_enc1, Y)

m2_3200 = MLPRegressor(max_iter=3200)
runModel(m2_3200, X_enc1, Y)

m2_10000 = MLPRegressor(max_iter=10000)
runModel(m2_10000, X_enc1, Y)

#m2_6400 = MLPRegressor(hidden_layer_sizes=(4000,800,200,40), max_iter=6400)
#runModel(m2_6400, X_enc1, Y)
#Accuracy: 0.8635157153369074
#Accuracy: 0.8635157153369074
#MSE: 0.13648428466309262
#Confusion: {'TP': 2196, 'FP': 5415, 'FN': 4972, 'TN': 63521}

m21 = MLPRegressor(max_iter=400)
runModel(m21, X, Y)

#Accuracy: 0.87912593293388
#MSE: 0.12087406706612004
#Confusion: {'TP': 2789, 'FP': 4820, 'FN': 4379, 'TN': 64116}

m32 = MLPRegressor(max_iter=800, random_state=1)
runModel(m32, X, Y)

#Accuracy: 0.8833569851781772
#MSE: 0.11664301482182277
```

```
#Confusion: {'TP': 2951, 'FP': 4660, 'FN': 4217, 'TN': 64276}

#runModel(MLPClassifier(hidden_layer_sizes=(189,63,21), max_iter=300, activation="relu", solve
#Accuracy: 0.9085724797645327
#MSE: 0.09142752023546726
#Confusion: {'TP': 877, 'FP': 667, 'FN': 6291, 'TN': 68269}

#runModel(MLPClassifier(hidden_layer_sizes=(189,63,21), max_iter=300, activation="logistic", s
#Accuracy: 0.9089535372647956
#MSE: 0.09104646273520446
#Confusion: {'TP': 638, 'FP': 399, 'FN': 6530, 'TN': 68537}

#runModel(MLPClassifier(hidden_layer_sizes=(189,63,21), max_iter=800, activation="logistic", s
#Accuracy: 0.9087301587301587
#MSE: 0.09126984126984126
#Confusion: {'TP': 671, 'FP': 449, 'FN': 6497, 'TN': 68487}
```