**Thomas Reagan**

**C964 - Capstone Project**

**Heart Disease Prediction Application**

**A1. LETTER OF TRANSMITTAL**

January 5, 20201

John Smith

EasyMed Insurance Co.

1234 Easy Street

Houston, TX 78987

Dear Mr. Smith,

The medical industry has been evolving drastically over the past half-century. With modern advances in medicine, medical determinations have become more complicated. By extension, this complicates the health insurance industry. As the determining factors for diagnoses become more prevalent, it becomes more likely that potential insurance recipients are at risk for health complications, thereby creating financial liabilities within the company.

We at MLD (Machine Learning Diagnostics Co.) are proposing a web application that can help handle the logistics of customer medical data in terms of determining whether a particular customer is at risk of a medical condition. Given the data we have access to, we can currently put together an application specifically to predict if someone will be at risk for developing heart disease within ten years. The application takes a basic input of a variety of medical readings and attributes for a person and outputs if they are at risk.

This type of efficient feedback can help your company in its financial decisions. Premiums can be adjusted for more at-risk clients in order to maintain the financial integrity of the company. As medical costs skyrocket, it’s more important than ever to be able to efficiently and accurately make calculated business decisions.

While we don’t have direct experience in the medical industry, we have a team of seasoned developers specialized in the field of data analysis and machine learning, and we expect the development process and implementation methods to be of similar scope to other industry projects of a similar nature. We anticipate the total funding required for this project to not exceed $36,000. Given that we already have access to the tools and resources necessary, there are no additional costs to be incurred outside of the man-hours needed to develop the application.

We hope to hear from you in hopes of building a mutually beneficial business relationship between our two companies.

Sincerely,

Thomas Reagan

CEO - ML Diagnostics Co.

**A2. PROJECT RECOMMENDATION**

**Problem Summary**

The problem is that traditional methods for analyzing medical data are becoming more cumbersome as medical science progresses. As we learn more about what can contribute to the development of heart disease, more attributes on a stat sheet are created, meaning a larger logistical burden in determining accurate trends on which to base decisions. In order to properly price premiums for clientele, insurance companies need to be able to accurately analyze the health-risk potential of their customer base.

Efficiency and accuracy are prudent in alleviating this problem. Utilizing modern advances in machine learning, the application will be able to determine with >80% accuracy whether a particular person is at risk for developing heart disease within a ten year time period.

**Application Benefits**

The application will be able to present a variety of relation charts and other visualizations using a relatively large dataset in order to present some of the trends and methodology behind the prediction model. The prediction tool itself will consist of a handful of input boxes and will present an accurate prediction of heart disease development within ten years for the input individual. The application format allows for speedy analysis of any given patient relative to a larger data sampling as well as providing the direct prediction without any manual, time-consuming, analysis. The direct benefit is the apparent ease of analytical burden on active data analysts within the insurance company. By trivializing the determination of the heart disease riskiness of a client to a few seconds, employees are able to direct their time and resources to other relevant tasks, whatever they may be.

**Application Description**

The Python programming language is standard in machine learning and data analysis science and as such, this application is written in Python as well. Data analysis, as well as model training and testing, will be done in Jupyter Notebooks. The application itself will be developed using a Dash framework and deployed on the Heroku web platform. The app opens to a user dashboard that consists of a couple of options. A dropdown at the top offers a few graph options for visual data analysis by the user. Below the graphical representation is the prediction module. There is a series of input options, some text, and some radio buttons. With every change made to the input values, the model recalculates and displays its prediction in the text box at the bottom. This all happens in real-time, meaning employees can input client data back to back without necessarily having to change all of the input variables every time. If only a couple of attributes change from one client to the next, the user only needs to change those two and the application automatically adjusts to the new values. This increase in efficiency is important when analyzing potentially hundreds or thousands of clients daily.

**Data Description**

The data that will be used to build a prediction model is a CSV (comma-separated values) file containing medical stats from thousands of subjects. It’s from the Framingham heart study data set which is public domain. The Framingham Heart Study is a renowned study into cardiovascular disease, and they have been critical to research in determining what causes heart disease. Their study’s status in the medical community is important in our determination that the data is valuable for use in developing our prediction model.

Each row in the data set represents a test subject and contains sixteen attributes: sex, age, education, smoker status, cigarettes per day, hypertension, whether they’re on blood pressure medicine, stroke history, diabetes status, total cholesterol levels, blood pressure readings (both systolic and diastolic), body mass index, heart rate, glucose levels, and whether or not a subject developed heart disease or not. The data is all represented numerically, which makes its implementation into the application easier. The heart disease attribute is what our model will seek to determine based on the input of the other variables.

We will also determine through data analysis whether certain attributes may or may not be relevant, such as the one for education level. There are also a few hundred samples that have incomplete data, and also through analysis, we will determine what should be done about them.

**Objective and Hypotheses**

The primary objective of the application is to deliver a quick and accurate heart disease prediction through an easy to use web interface. We will establish an intuitive dashboard with a series of input options and an output that’s clear in its prediction delivery. The primary output will be a simple ‘yes’ or ‘no’ to represent whether the client is considered at-risk for heart disease within a ten year period. The secondary objective is to provide some of the critical data analytics via visualizations that the user can select for inspection if they wish to compare their input stats to the dataset used in the application’s development.

Our hypothesis is that the model will be able to determine trends that align with the medically understood correlations between these stats and heart disease risk. And upon determining these trends, accurately predicting the heart disease target. Remember, this application is for the sake of efficiency in data analysis by people who aren’t necessarily medical experts and not for making medical breakthroughs. If the medical stats entered in alignment with what we’d expect a doctor to establish as being at-risk for heart disease, then we expect our model to make the same determination.

**Methodology**

The application will be developed using the agile methodology. Using Jupyter Notebooks lends itself to this methodology since we will be able to perform data analysis cell by cell and get immediate feedback to determine how we want to proceed. Developing a machine learning model takes an iterative approach, slowly testing different variations to establish what solution works best for our needs. An agile approach gives us the freedom to play around with the data and experiment before implementation, therefore saving time and resources by not having to rework sections of the application based on what could be minor changes.

**Funding Requirements**

All the tools used to create the application are freely available. The only costs incurred will be the labor cost for the man-hours needed for the development, testing, and implementation of the web app. We expect this to not exceed $36,000 as we anticipate the application requiring 120 total development hours split amongst three developers. Forty hours per developer at $100 per hour each comes out to $36,000.

**Stakeholders Impact**

The stakeholders for our company support this venture into a new industry. We are a data analytics and machine language application company, and diversifying our clientele only confers benefits to the stakeholders as it fosters lasting business relationships which in turn are lucrative to the stakeholders. Machine learning is becoming more ubiquitous in data analytics, and they understand the importance of its application across various fields of industry.

Similarly, EasyMed Insurance Co. should support this project for similar reasons. The benefits of the application will allow the company to successfully and more confidently navigate business decisions that will be very relevant in competing with other companies. Offering clients an attractive premium while properly mitigating the financial risk that comes with taking on those with sub-prime medical stats is critical for staying competitive in the insurance industry. This project allows the company to make the assertions more efficiently without needing medical expert middle-men to analyze the data, which can be both time-consuming and expensive.

**Data Precautions**

The Framingham data is anonymous and as such we are not required to navigate any legal implications regarding sensitive information. This medical information was voluntarily given to the researchers and is therefore not subject to HIPAA regulations.

**Developer’s Expertise**

Our company has developed and delivered similarly predictive applications for other industries, particularly sales forecasts for targeted products. We have a history dealing with machine learning algorithms and their benefits in handling various types of data. Given that we’ve been through a similar development process for companies in the retail industry, we have a base of knowledge and expertise within the team to build from. We have full confidence in our ability to meet our development milestones and to roll out the application based on the agreed timetables.

**B. PROJECT PROPOSAL**

**Problem Statement**

As healthcare becomes more expensive and medical research becomes more pronounced, insurance companies find themselves having to analyze more and more data while potentially exposing themselves to financial risk by taking on at-risk clients. Insurance companies become profitable by mitigating this risk by having to pay out for as few clients as possible while still collecting premiums from a large number of them. It’s therefore risky for a company to take on any client possible, as many of them can be riskier than others medically. To counter this, companies need to maintain personnel that deals with data analytics to help make these financial determinations.

Our application seeks to streamline much of this data analysis via machine learning. The application’s prediction model takes a hefty dataset of the medical stats of people, analyzes patterns between these stats from subject to subject, and predicts whether or not a series of the same stats for a single subject will be classified as ‘at risk’ or not. Our prediction is for heart disease development within a ten-year window with this proposed app. Heart disease is the leading cause of death in the United States, and as such, being able to determine if someone is at risk for developing it within ten years can be critical for insurance companies in deciding on premium pricing and/or client admission. Reducing the need for any manual medical data analytics is also instrumental in saving money and preserving resources. The company will be able to more appropriately structure itself and accurately navigate business decisions with such an increase in efficiency.

**Customer Summary**

The application is being developed specifically to be used by data analysts within a medical insurance company. The intent is that those who generally pore over spreadsheets of data to compile relationships between data, often doing so manually, will have a new tool to boost the efficiency of their work. Manually analyzing medical data can take time and resources, such as maybe needing consultation of medical professionals, but our tool is made to free up those resources and deliver the same results in a more timely manner.

All of the calculations and analytics are done in the application, so no additional skillsets are required to utilize it. All the end-user needs to be able to do is navigate a browser to get to the website that hosts the application, and from there it’s basic site navigation and data entry. The dropdown menu for the graphs and charts are clearly labeled, as are the text fields and radio buttons for the prediction tool. The input fields also handle invalid input options, helping to keep user error to a minimum.

**Existing System Analysis**

The application is a web app, meaning as long as the company allows access to the particular URL for it, it will not require any system overhaul or restructuring. All that will be needed are computers with access to any of the standard set of browsers: Edge, Firefox, Chrome, Safari, etc. Additionally, depending on the client’s security systems, the web app may need to be added to a whitelist to grant employees access to it.

**Data**

The source of the data to be used in the development of this application is from the Framingham heart study. The data is collected in a CSV file and is available online at multiple locations, but we obtained it ourselves at [www.kaggle.com](http://www.kaggle.com). The dataset represents over four thousand people, each with sixteen attributes. These attributes mostly contain medical information, but there are a couple of social elements as well. These outlier attributes will be tested for relevancy in determining the target and disposed of if they’re deemed unnecessary for the goals of the project.

There are also hundreds of missing values spread across multiple entries and attributes. Similar to attribute testing, determinations will be made based on testing on how to handle these entries. Our options are to either drop them entirely or fill them in using logical reasoning or with statistical averages depending on how much sway they could have on the model’s performance overall. Otherwise, nothing else needs to be done to the data. In its available state, it is already all numerical data, which is crucial for its implementation into the prediction model and the application.

There aren’t any additional data requirements beyond the Framingham data set. The sample size is large enough and there are a sufficient number of attributes to effectively train a prediction model.

**Project Methodology**

We will be applying the Agile methodology for the development of this application. We believe that the flexibility and adaptive nature of this methodology perfectly suit our team’s abilities and experience. Our team, having used this methodology in past projects, knows that continuous and iterative changes will need to be made during the development process. This process is more precisely broken down as follows:

Requirements:

* The requirements for the software will be established and finalized in accordance with the agreements on expectations among all relevant stakeholders to the project.
* While the agile development method is all about flexibility and adaptation, the fundamental requirements need to be established as solidly as possible, as doing so better gives the development team the ability to meet their milestones.

Development / Testing:

* After the requirements are established, the development process can be broken down into tasks.
* Using a scrum system, these tasks will be grouped into development sprints, creating iterative milestone goals to be accomplished along accompanied timetables.
* These tasks will be aligned in a way to maximize flexibility and allow for the retooling of previously met milestones if necessary.
* The code will be continuously tested throughout the development cycle, giving the need for the aforementioned flexibility.
* Early testing will involve running the code block by block to ensure the fundamental systems are functioning correctly as they are being developed. All of this testing will be done in Jupyter Notebooks on the data analysis side and within PyCharm, a Python IDE, on the application side.
* Later testing will be a continuation of this as well as black-box testing (running and using the application with the dashboard, as an end-user would.) This will continue after the application is pushed to Heroku to verify functionality remains intact in accordance with the requirements.

Delivery / Feedback:

* At this stage, the application will be delivered to the client for live testing. Client personnel training will be assisted with the provided overview of the app and instructions on how to use it.
* User feedback will be collected during the live test period to address any issues or suggestions for changes as declared by the client or determined by the development team.
* After any changes are made, another period of live testing occurs and the feedback loop continues until the requirements are met and the client agrees as such.

**Project Outcomes**

The deliverables will be a development schedule as well as the application itself. A detailed schedule for the development process will be established following the finalization of the requirements as agreed on by all stakeholders involved. This schedule will then be made accessible to said stakeholders.

The application itself is the final deliverable. Once it is ready for live testing, the web app will be made available to the client for use within its organization. Should it be agreed upon as a requirement, the data analysis process and testing done in a Jupyter notebook, as well as the raw dataset itself, can be provided in addition to the application rollout.

**Implementation Plan**

Implementation will occur across three phases. The first phase consists of testing the fully-featured application locally in a web browser. In this phase, the development team will test the usability to verify that the functionality is complete and the requirements are met.

The second phase of implementation is verifying compatibility within the Heroku web app platform. Internal testing will continue with the web app post-Heroku deployment to verify nothing broke with the application’s transition to the platform. Given that Heroku’s Python libraries aren’t necessarily as up to date as those used during the coding process, it’s important to ensure that all of the functionality is intact as previously tested in a local environment. During this phase, the team can also test the web app in a variety of browsers and create a compatibility list for the client to ensure staff won’t encounter any issues.

The final phase of implementation is the web application being made fully available to the client for acceptance testing. At this point, the development team can make iterative changes based on user feedback until an end result is reached that the client is satisfied with. This live test environment can be based on a full rollout to the client’s entire staff or to only a small testing division, as determined by the client.

**Evaluation Plan**

The first step of evaluation occurs during the internal testing phase of implementation. At this point, the application is in working order, and the team will verify not only the functionality of the app’s mechanics but also the accurate representation of the data being used. The data used in developing this application is not sensitive and is publicly available, so we need not consider navigating any established regulatory practices. There is also the need to evaluate the responsiveness of the application and verify that it can output a prediction within a few seconds of a changed input variable.

A large part of the evaluation occurs in the third phase of implementation. Given that the client needs to be satisfied with the end result, iterative acceptance testing is the best way to determine if the requirements are being met and to what extent. These determinations will be made via user feedback in the form of surveys or questionnaires. We will determine what percentage of users find the application helpful, how easy it is to use, and to what degree it allows for them to more effectively analyze data compared to their previous methods. Additionally, we can determine what aspects of the application are most useful as well as suggested options to be added that could be useful.

Beyond its development process, further evaluation can occur on the impact of the application in a period of time after its rollout. This can occur via surveys or other correspondence with the client’s management/parties privy to the knowledge of the efficacy of the application. From this correspondence, we can evaluate how well the application addressed the problem of data analysis efficiency in the medical insurance industry with regards to this specific ailment, and further, determine how much of an impact this boost in efficiency had on business decisions.

**Resources and Costs**

Programming Environment:

* Our team uses PyCharm Professional Edition which incurs a cost of $19.90 per month for a license. There will be a total of three developers working on this project over the course of two months, so the total cost for the IDE license comes out to $119.40.
* Additional Python libraries will be used to develop the application: Pandas, Matplotlib, NumPy, Scikit-Learn, and Jupyter Notebooks. These are all available for free and won’t incur any additional costs.
* The Heroku platform will be used for live deployment of the application, and this hosting incurs a monthly cost of $25 for our development needs. This cost will only be needed during the second month for testing.
* Note: Depending on how many staffers the client wants to grant access to, this monthly cost could go up post-rollout as high-traffic plans on Heroku can cost up to $500 per month. This cost won’t apply to the development costs, however.

Total Cost for Programming Environment: $144.40

Environment Costs:

* Nothing proprietary is required to deploy the application into a live environment. All that is required is a computer installed with an industry-standard, up to date browser and access to the internet. The client is already equipped as such.
* Depending on the client’s security infrastructure, the application may need to be whitelisted to obtain access. The client has an internal IT department that will be able to handle this transition.

Total Cost for Environment: $0

Human Resource Requirements:

* We will have a team of three full-time developers working on this project. The development plan is over the course of two months.
* We estimate that the application will take 400 total hours to develop, test, and implement.
* Our company is charging $80 per hour for each developer assigned to the project. This comes out to just over 133 work hours per developer at a rate of $80 per hour which totals $32,000.
* Additional costs could be incurred depending on the client’s requests for changes that extend beyond our timetables.

Total Cost for Human Resources: $32,000

Total Cost: $32,144.40

**Timeline and Milestones**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Phase** | **Event** | **Start** | **End** | **Duration (Days)** | **Dependencies** | **Resources** |
| 1 | Establishing Requirements | 1-25-21 | 1-28-21 | 4 | N/A | Developers and Managers |
| 1 | Meeting with Stakeholders | 1-29-21 | 1-29-21 | 1 | Requirements | Developers, Managers, Stakeholders |
| 2 | Sprint 1 (Analyze Data and Test Models) | 2-1-21 | 2-5-21 | 5 | Phase 1 | Developers |
| 2 | Sprint 2 (GUI and Data Implementation) | 2-8-21 | 2-12-21 | 5 | Sprint 1 | Developers |
| 2 | Sprint 3 (Internal Testing and Heroku Deployment) | 2-15-21 | 2-19-21 | 5 | Sprint 2 | Developers |
| 2 | Meeting with Stakeholders | 2-22-21 | 2-22-21 | 1 | Sprint 3 | Developers, Managers, Stakeholders |
| 3 | Delivery and Live Testing / Feedback Gathering | 2-23-21 | 3-5-21 | 9 | Phase 2 | Developers, Managers |
| 3 | Alterations | 3-8-21 | 3-10-21 | 3 | Feedback | Developers |
| 3 | Additional Live Testing and Feedback Gathering | 3-11-21 | 3-12-21 | 2 | Changes Made | Developers, Managers |
| 3 | Alterations | 3-15-21 | 3-17-21 | 3 | Feedback (2nd Round) | Developers |
| 3 | Meeting with Stakeholders (Project Finished) | 3-18-21 | 3-18-21 | 1 | Changes Made | Developers, Managers, Stakeholders |

**D. PHASE ONE POST-IMPLEMENTATION REPORT**

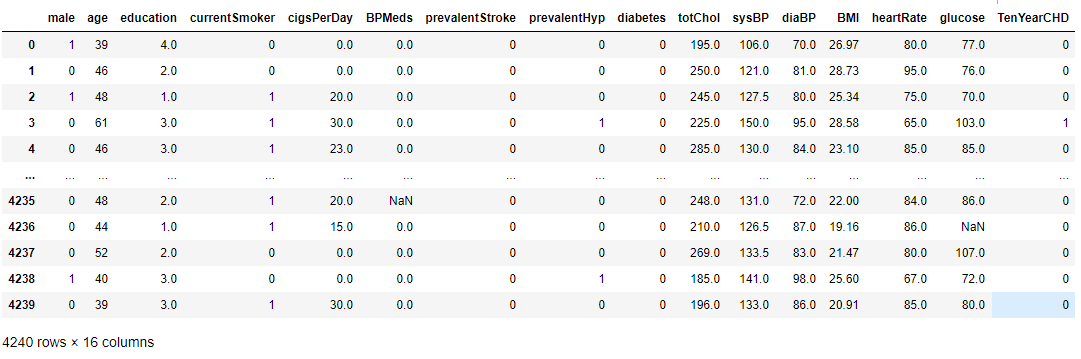
**Project Purpose**

As data analysis becomes ever more important in the medical industry, the need for greater efficiency arises. Insurance companies in particular need to be able to base their business decisions on medical knowledge since their entire business depends on whether or not people get sick, then requiring the company to pay out. As medical research further determines what is medically relevant for a particular affliction, data analysts then need to be able to cross-reference that research with their clientele data. But as more and more data becomes available through technological progress, it becomes unwieldy and inefficient to deal with in a manual capacity. Even further, issues arise with the medical training needed to properly assess the data, creating the need for either third party consulting or maintaining high-cost in-house personnel.

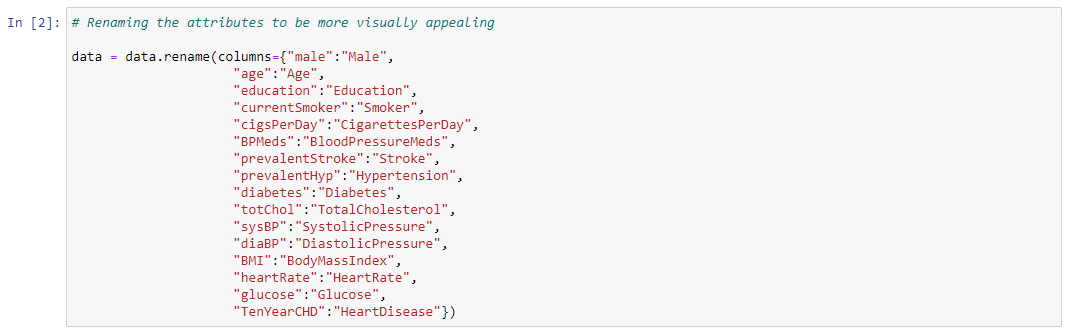
The purpose of this project was to solve this problem of inefficiency. By using a machine learning prediction model, we were able to create an application that takes a series of inputs from a user and outputs whether or not those inputs represent a candidate for heart disease development. Since the model was trained using a large dataset, it can accurately make this prediction that would otherwise require the assessment of a medical expert. Even better, the prediction occurs in real-time at the change of any input variable. This makes the application very accessible since it only requires the user to be able to use a keyboard and mouse to set the inputs. Furthermore, should they desire, users can inspect the data used in training the model by generating visual representations of different trends. Given these options and the application’s ease of use, the workload of a data analyst using this tool has greatly decreased, at least for this particular ailment. Eventually, this tool will be a launching point for additional tools of similar scope, but for other diseases and ailments.

**Datasets**

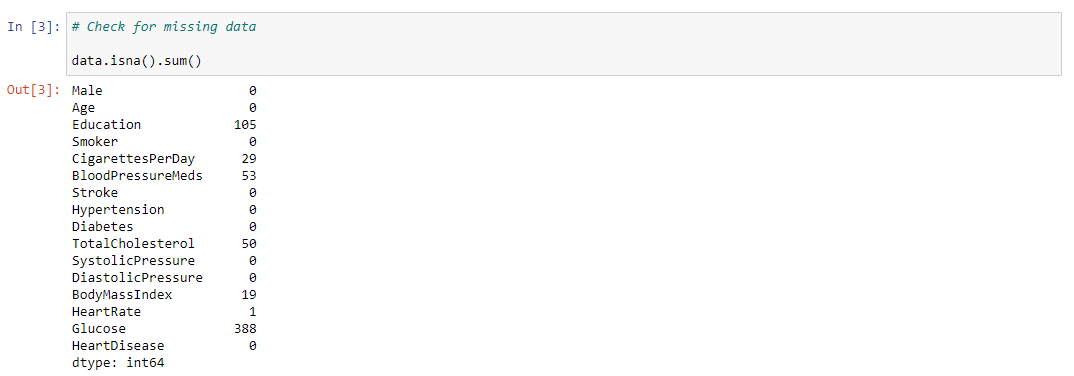
The raw data set is a comma-separated value (CSV) file. Searching “Framingham Heart Disease” on [www.kaggle.com](http://www.kaggle.com) will populate various sources for the same set. Below is a screenshot of the raw dataset displayed in a Jupyter notebook:



The data was modified in a few ways to fit the project needs. Firstly, the attributes were renamed for better readability. Beyond that, the data was missing various attribute values amongst a few hundred rows. This was handled depending on the situation for each attribute and its relevance for accomplishing the project’s goals. The process for this is shown in screenshots below:



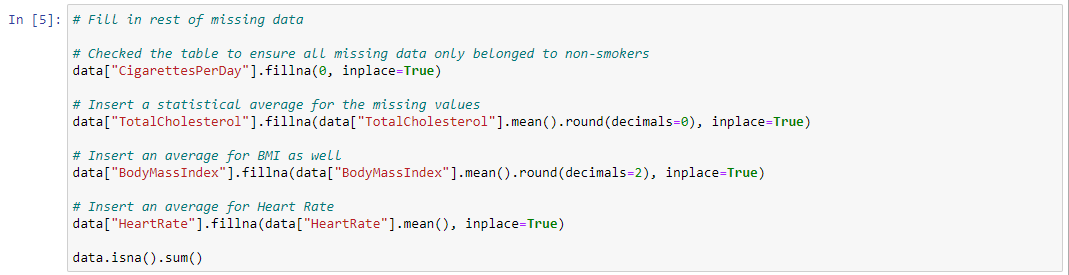
First, we renamed the attributes just for aesthetic purposes. Since there are over 4,000 rows in this dataset, we next needed to check for where there are missing values and what their distribution looks like across the different attributes. The method for checking these numbers is shown in the next screenshot:

****

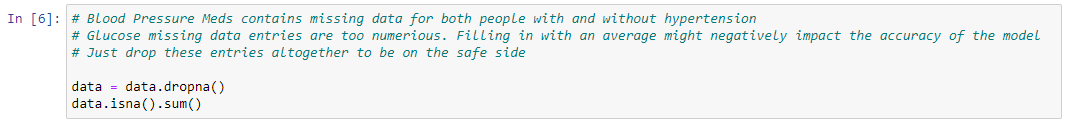
Here we can see that there are a significant number of missing values in the “Education” column. Instead of dropping these samples entirely, it’s better to just drop the entire column from the dataset as the level of education someone has isn’t relevant to whether or not they are medically at risk for a condition. It’s merely added for social research along with medical research.



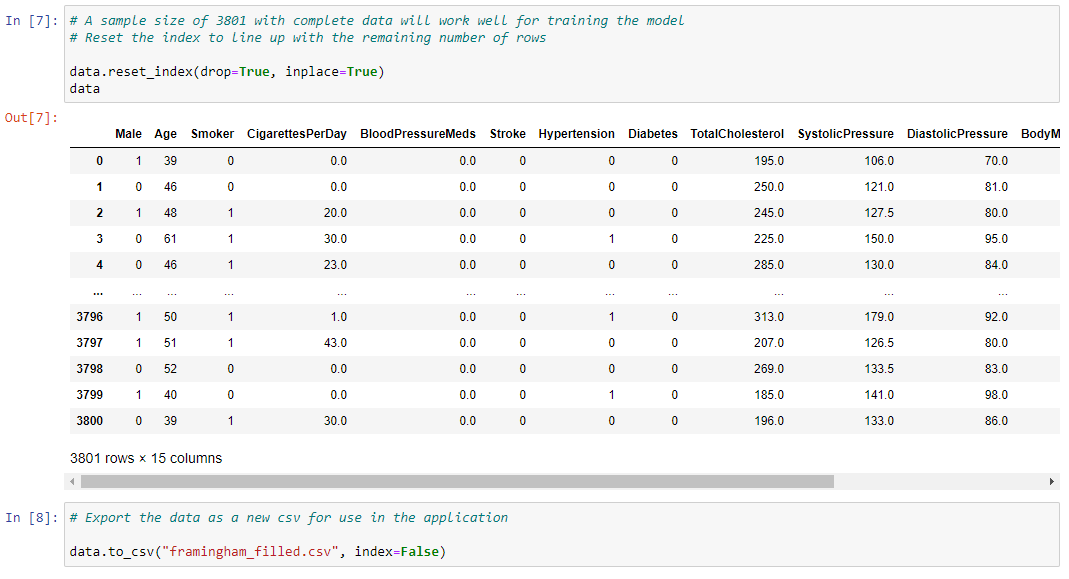
Next, it was necessary to determine how to fill in the remaining values. These range from logical assumptions to using statistical averages in the study. The idea is that the total number of missing values for each of these attributes is small enough to not have any significant sway over the performance of the prediction model. We tested the model with various methods, but overall, it’s better to maintain the larger overall sample size.

****

Using the same reasoning, we determined that it’s best to drop the rows with missing glucose values since it’s a statistically significant total. At 388 missing entries, if we averaged them, it could create a statistical anomaly that could throw off the prediction model. Additionally, the blood pressure meds attribute contained people that both had and did not have hypertension, making any logical assumption, like for the smoker attribute, impossible.

****

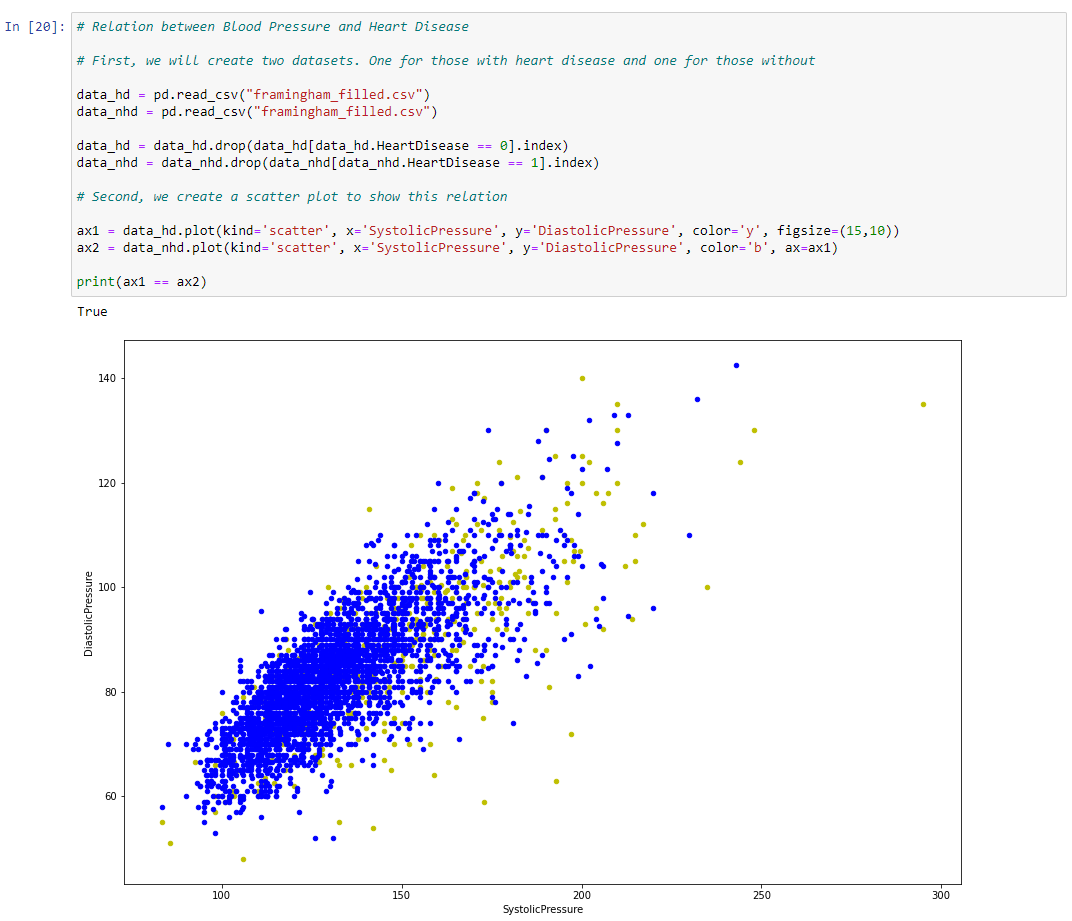
We ended up with a more complete dataset with no missing values while maintaining a sample size of 3,801.



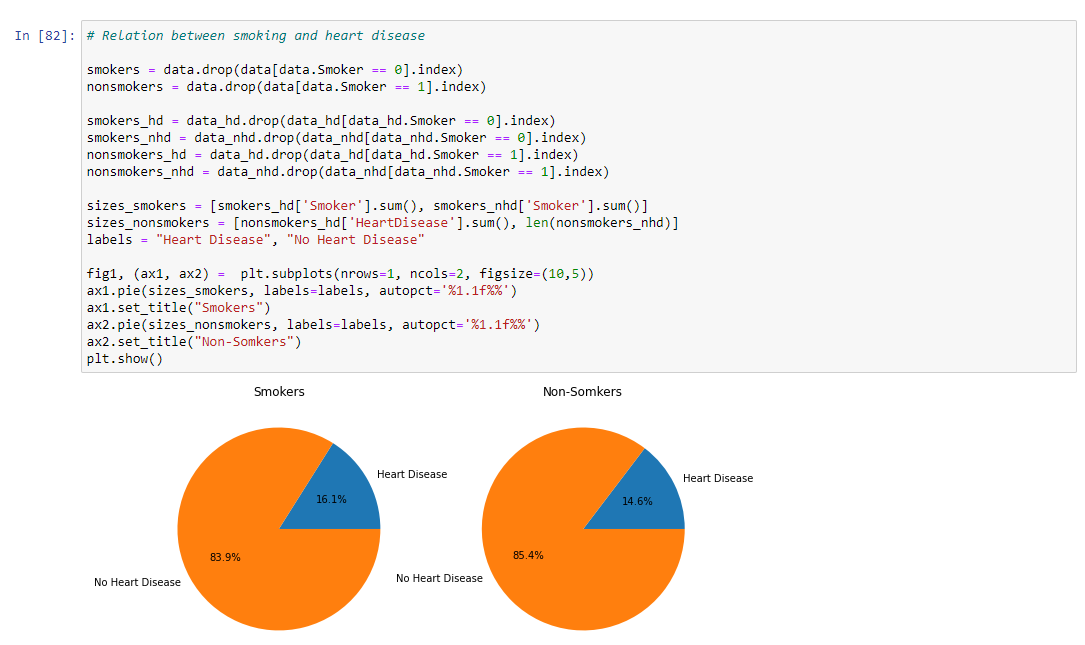
Finally, we exported the modified data as a new CSV for use in the application.

**Data Product Code**

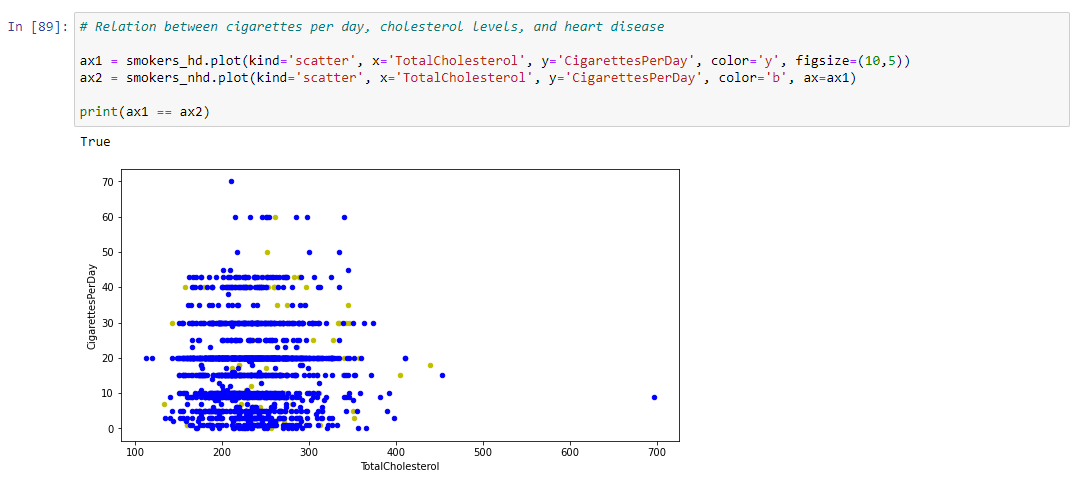
We first analyzed the data in a Jupyter notebook to determine the expected behavior for the prediction model by generating graphs showing different relations between certain data and the target heart disease identifier. First, we determined that there was an obvious relationship between blood pressure readings and those who developed heart disease within ten years.

****

We also observed whether there was a connection between smoking and heart disease. While not significantly so, there is a larger percentage of smokers that developed heart disease than non-smokers. Furthermore, we observed how the number of cigarettes smoked per day related to the development of heart disease.



Observations of the relationship between the number of cigarettes per day and cholesterol and how they affect the heart disease rate were also analyzed. There are visually minor trends in the plot, but these trends are how the model makes its determinations.



Further similar analyses were done to consider relations between other data as well, such as sex and age, glucose, and BMI. All of the observed trends were used to set expectations for what we expected to see from the model’s predictions. The source code for this analysis is available with an attached Jupyter notebook file, and the application source file containing similar methods to dynamically generate graphs of this nature is also available.

**Hypothesis Verification**

We hypothesized that the prediction model would mimic conventional medical thought with regards to how this kind of medical information relates to heart disease. In other words, generally higher stats across all fields correlates directly with an increased risk of heart disease. And the model acts as such. When we test it by inputting much higher than average values for traditional indicators for heart disease, such as blood pressure and age, we predictably get an accurate result that this person is at risk for developing heart disease within ten years.

An anomaly we didn’t anticipate was the prediction model not knowing how to react to an input that’s wildly out of any kind of normal range. We suspect that since its predictions are based on a series of humanly grounded values, which it weighs independently depending on their relation to the target, that it depends also on those values being input in concert with others to determine an accurate prediction. For example, setting the cholesterol to an obscene amount may not budge the model’s ‘no’ prediction despite it being ridiculous to assume that such a human wouldn’t have heart issues. So while our hypothesis was, in essence, correct, it’s only so given inputs within a reasonable, expected range.

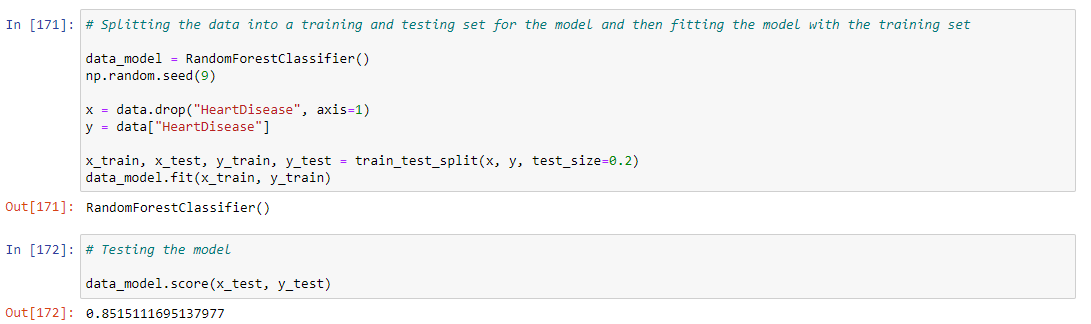
**Effective Visualizations and Reporting**

Within the application, the user has the ability to generate five different visual representations that show the trends the model is working with when it interprets the dataset. First is the blood pressure scatter plot that shows how higher blood pressures tend to trend along with developing heart disease. Second is the option to compare the rates of heart disease between sexes via a pie chart, which shows that males tend to be more prone to heart disease. The third option is a bar graph comparing smokers vs nonsmokers which shows a slight trend that smoking does mean a higher risk of heart disease. The fourth graph option offers an extended look at smoking data, comparing how many cigarettes a person smokes to their cholesterol level with different plot colors representing who developed heart disease. Lastly, the user can take a look at the distribution of heart disease based on age.

These visualizations don’t show drastic trends in any capacity, but minor trends compounded with one another establish a story of how all of these attributes relate to the development of heart disease, which is what the model does when it’s trained. It finds these same trends when it interprets the raw numerical dataset, and uses those trends to make its predictions.

**Accuracy Analysis**

Using the model’s built-in method to score itself, it returned an accuracy rate of slightly over 85%. The idea here was to take the original dataset and split it into two groups. The first set is the training set; the set that the model analyzes for patterns in order to make its predictions. The second set, consisting of 20% of the total set, is the test set; the one the model uses makes predictions on. This means that on that 20% of data rows, the model successfully determined the target “Heart Disease” attribute’s value correctly 85% of the time. This training and testing took place in a Jupyter notebook as shown here:



Furthermore, when testing the accuracy within the application itself, we first needed to verify the authenticity of the visual representation that we’re using to draw our comparisons. The data was confirmed to be accurately displayed, and when we input a range of variables in the predictor that mirrored what the trends indicated, we would most often get the expected result. For example, if we set the age in the 60s and boost the blood pressures to well above average, tick the smoker button, and put in 20 cigarettes per day, we’d expect a ‘yes’ output, and the model succeeds in this more often than not. Conversely, we expected the model to output ‘no’ for a vast majority of inputs since the vast majority of them would naturally fall within a normal range of healthy stats, and it does. Therefore, the predictor functions as we expected, affirming its accuracy score of about 85%.

**Application Testing**

Testing occurred across all phases of development. The initial bulk was in the form of unit testing. This took place primarily in Jupyter notebooks for cleaning our data, analyzing it, and applying it to our prediction model. During this testing, we were able to determine how to represent our data visualizations and finalize the model to be implemented into the application itself. Additional unit testing also took place during the application’s development. This occurred when each task was accomplished to verify proper functionality (for example testing the GUI first, then implementing a graph for one of the dropdown options and subsequently testing it, etc.)

Next, internal black box testing took place in a local environment. Each team member would run and use the application as if they were the intended user to gauge not only functionality but usability as well. After this was another round of internal testing, but this time after migrating the application for use on the web platform Heroku. This additional round of testing was to determine that the application made the transition without any compatibility issues, both with the Heroku platform’s implementation of Python libraries and with various web browsers.

Finally, we had two rounds of acceptance testing in a live environment. Testing in this stage was user-guided since all updates that came as a result of this phase of testing derived from user feedback. Acceptance testing is the most critical part of the refining process since there is a much wider presence of experiences and ideas to draw from. An example of a change happening as a result of this stage of testing is the addition of a probability indicator in the output box for the predictor. Initially, there was no visual indication that the model was doing anything as any variables changed in small ways since it could persist on a constant yes or no output. While the development team knew what was happening under the hood, the users were unsure if it was working at all since there are no visual cues. With an update, however, you can now see a probability indication for the predictor change as you alter variables bit by bit; an important update for usability.

**Application Files**

All of the project files can be accessed at the following GitHub link: <https://github.com/ribeye9/HeartDiseasePredictor>. In this repository is the Jupyter notebook file where data cleanup, analysis, and model training and testing was done. Also included is the CSV file containing the dataset used to train the prediction model and to draw the data visualizations in the application. The file with the extension .pk1 is the prediction model itself. This file needs to exist in the directory with the main python file as well since it’s directly imported into the application for use. The requirements txt file is necessary for Heroku when pushing the app to the web platform as it needs to know what libraries are needed by the application to function. Finally, there are the Python files: main, which is the actual application file that contains the dashboard layout and the callback implementations, and ‘users’ which is a separate file declaring a dictionary of acceptable usernames and passwords necessary to login to the app when it’s launched. Ideally, this users.py file wouldn’t be included in a public repository like this since it contains sensitive security information, but it’s necessary to demonstrate how this project operates.

**User’s Guide**

Using the Application:

* Visit this link: <https://heartdiseasecapstone.herokuapp.com/> (Ensure you are using a common browser such as Chrome, Edge, Firefox, or Safari and that it is up to date.)
* Sign in using any of the following credentials (Username / Password):

Admin / Admin

Hello / World

Capstone / WGU

* Once logged in, you are met with a simple dashboard. The top half of the page consists of the data visualization options. There is a dropdown menu with five distinct relations to analyze. These options are clearly named and when selected, a new graph or chart will generate in the graph field. The user has the option to hover over aspects of the graph to see additional information and has the ability to filter sets by clicking on the legend labels. Additionally, for the scatter plots, the user can zoom in by clicking and dragging a box to direct their zoom.
* The bottom half of the dashboard contains the predictor. Here the user will find various input options with default values set. Below these input fields and buttons, there is the output box labeled ‘Heart Disease.’ As changes are made to the input values, the output will change accordingly (since the app is hosted on a web platform, there is some minor latency, so these updates could take up to a few seconds.) The input fields are restricted to the correct data type, so you can’t input a letter into a numerical field for example. The output field is not editable and contains a probability prediction for additional model information as well as a visual cue to know with certainty that the model is indeed outputting a prediction with each change to a variable.

**Summation of Learning Experience**

Working on this project really encapsulates what software development is all about. It encompasses the entire process, from conceptualization to implementation and even to documentation. It’s also an important lesson in the necessities of research and then trial and error. Persistence is key in software development, as things are ever-evolving, and my experiences in pursuit of a computer science degree reflect this. A degree in this field represents not only an ascertained level of knowledge but also the ability to seek and utilize the tools and resources necessary to solve whatever problems you might come across.

I came into this project with only minor supplemental programming knowledge beyond what was already covered in previous courses. And I expected something that was basically an amalgamation of covered concepts, but it wasn’t that at all. Instead, I felt almost lost from the outset. Past classes helped me understand how all of the pieces work, but putting the puzzle together can always be something new despite these understandings. I understood how modules work in a program, but never had to actually use any third-party libraries and create an environment using an IDE. I understood networking protocols and cloud services, but never actually had to host an application myself and learn how to push updates to it. I hardly ever found myself using a basic Command Line interface before, but now I feel like I’ve learned a great deal about it due to this project. I can keep going on with more examples in this project alone. There are so many options available to use in software development, and it’s ever-expanding, that the final lesson learned is that it will always be necessary to consult documentation and outside resources to guide you. This is an impossible field to learn everything, so learning how to teach yourself anything is the most important skill for a developer.