This file contains 3 documents: a letter of transmittal, a project proposal and a post implementation report. This was a project for my undergrad degree in computer science (the full requirements for this project can be found in the README).

Contents

[Prompt A - Letter of Transmittal 2](#_Toc48304550)

[Prompt A - Project Recommendation 4](#_Toc48304551)

[Problem Summary 4](#_Toc48304552)

[Application Benefits 4](#_Toc48304553)

[Application Description 5](#_Toc48304554)

[Data Description 5](#_Toc48304555)

[Objectives and Hypotheses 6](#_Toc48304556)

[Methodology 6](#_Toc48304557)

[Funding Requirements 6](#_Toc48304558)

[Stakeholders Impact 7](#_Toc48304559)

[Data Precautions 7](#_Toc48304560)

[Developer’s Expertise 8](#_Toc48304561)

[Prompt B – Project Proposal 9](#_Toc48304562)

[Problem Statement 9](#_Toc48304563)

[Customer Summary 9](#_Toc48304564)

[Existing System Analysis 10](#_Toc48304565)

[Data 10](#_Toc48304566)

[Project Methodology 11](#_Toc48304567)

[Project Outcomes 12](#_Toc48304568)

[Implementation Plan 13](#_Toc48304569)

[Evaluation Plan 13](#_Toc48304570)

[Resources and Cost 14](#_Toc48304571)

[Timeline and Milestones 15](#_Toc48304572)

[Prompt D – Post Implementation Report 16](#_Toc48304573)

[Project Purpose 16](#_Toc48304574)

[Datasets 17](#_Toc48304575)

[Data Product Code 19](#_Toc48304576)

[Hypothesis Verification 21](#_Toc48304577)

[Effective Visualizations and Reporting 22](#_Toc48304578)

[Accuracy Analysis 22](#_Toc48304579)

[Application Testing 23](#_Toc48304580)

[Application Files 24](#_Toc48304581)

[Installation and User’s Guide 24](#_Toc48304582)

[Summation of Learning Experience 27](#_Toc48304583)

[Sources 29](#_Toc48304584)

# Prompt A - Letter of Transmittal

Office of the Executive Director

Natrona County Health Department

475 S. Spruce Street  
Casper, WY 82601

Subject: Creation of a data product to address growing concerns in the rise of heart attacks in the county.

Dear director,

As you well know, there is growing concern about the rise of heart disease in the county. It is believed that this spike in cases can be partially attributed to a lack of up to date research being made available to health professionals as well as a lack of modern information being made easily accessible to the public. Over the past several decades, the technological capabilities of human beings have grown exponentially. So has our understanding of the human body and its biological processes. However, research specifically in the area of heart disease has lagged behind.

Our proposal to meet both aspects of the problem is a web-based data product. This product will analyze a large file of patient data. This data will be split into two parts: a training set and a test set. The training set will consist of multiple features which provide information about the patient such as age, gender and the results of several tests. This set will also provide information about whether the patient has heart disease. This set of data can be utilized to analyze how these features contribute to heart disease. The test set of data will contain just the features and the product will need to take the information learned from the training set and make a prediction about whether the patient has heart disease. The accuracy of these predictions will demonstrate the effectiveness of the analysis on the training set and determine the plausibility of predicting heart disease.

There are two primary objectives of the proposed project. The first is to create a product that will analyze heart disease data and present it to the user in such a way that they can explore visualizations to answer the question “what causes heart disease?” The second objective is to add a predictive functionality to the product that can take the information learned by asking and answering that question to examine attributes of a test subject and, with reasonable accuracy, predict whether that subject has heart disease.

The benefit of hosting this application in a web-based environment is that it will be easily accessible to both medical professionals and average citizens. Medical professionals will be able to visually analyze data and precisely calculate how the attributes in the dataset contribute to heart disease. Patients will be able to view graphics depicting risk factors associated with specific traits as well as view the data to find a profile similar to their own.

My firm, DSI, specializes in bioinformatics research and is uniquely suited to handle this project. The team that will be dedicated to this project will consist of multiple developers, QA’s and UX designers and data scientists with several years of industry experience. The estimated funding for this project is $24,000. This number was reached by an estimation that the project will take 2 developers and 1 data scientist 75 hours, plus a UX designer working for approximately 15 hours. We charge a rate of $100 per hour for developer time. In the future, there may be an additional monthly charge for hosting services. The funding requirements will be discussed in the funding section of the project proposal.

Sincerely,

David Soule

David Soule

CEO

DSI LLC

# Prompt A - Project Recommendation

## Problem Summary

The problem at hand is that the incidence of heart disease in the area is rising dramatically. It is my belief that this can be partially attributed to a lack of sound, scientific data that is readily available to both healthcare professionals and the general public. Many doctors, nurses and other healthcare professionals operate primarily on information obtained during medical school. What we know about heart disease changes rapidly due to new research and expanding technology. The American Heart Association recently released an article stating that “Numerous research findings in 2019 yielded critical new information that builds on our already vast, existing knowledge of cardiovascular disease” (Kirkwood, 2019). While this information is available for healthcare professionals to research, the time commitment can be astronomical. There is a lack of relevant scientific data which can be obtained and analyzed quickly.

Healthcare professionals are not the only group sorely lacking in up to date information regarding heart disease. The general public, many of whom are at risk for the disease, operates primarily on anecdotal information. Many lack any sound understanding of the disease or its risk factors.

The proposed application will read and analyze a large collection of data pertaining to heart disease. It will use this data to evaluate which attributes correlate most strongly with a positive diagnosis of heart disease. The application will also be used to examine attributes to make a prediction of the heart disease diagnosis. This information will be presented to the user in interactive tables and charts with explanations of what the displays mean.

The application will not include any way for users to upload their own personal information, either through file uploads or prompts. Adding new data to the application will have to be handled by the development team to ensure that new data is processed the same way as the original data to prevent the introduction of ordering or formatting bias. The application is web-based, meaning that a browser and internet connection will be required. There will be no support for a desktop version.

## Application Benefits

The key benefit of this application is the ability to process large amounts of data and present it in an easily understandable way. First, data from numerous patients with a diagnosis of heart disease will be processed to depict which symptoms and characteristics contributed to the diagnosis. Additionally, the program will read data consisting only of attributes and make a prediction of whether the patient has heart disease. In the right hands, and with realistic expectations, this application can be a major asset in understanding heart disease and the factors that contribute to it.

## Application Description

The application will be written in the Python programming language. Python is popular in the fields of data science and machine learning. This application will make use of both. The tables and graphs will be created using the Dash library. Dash is a powerful tool for creating interactive displays in a functional dashboard. The graphs will display additional information with a mouseover and can zoom or pan. All tables can be filtered or sorted. Dash uses HTML to arrange the features, which will make the application easy to navigate for anyone familiar with navigating nearly any traditional website.

The source code will be stored in a GitHub repository and the application will be hosted on Heroku; a website dedicated to web app deployment. When changes or updates are required, new code can be pushed to GitHub and Heroku will automatically deploy and updated version of the application. Heroku also allows the creation of logs to monitor usage as well as the health of the application.

## Data Description

The data for this application will be stored in the format of a CSV (character separated values) file. This file will be downloaded from the GitHub repository. The original source of the data is the UCI Machine Learning Repository. The full dataset contains 76 attributes. Some attributes (name, social security number, etc.) are irrelevant and unnecessarily expose sensitive information while others (such as date of EKG reading) hold no correlation with heart disease. Due to this, the data used in the application will focus on these 14 attributes: age, sex, chest pain, resting blood pressure, serum cholesterol, fasting blood sugar, resting electrocardiographic results, maximum heart rate, exercise induced angina, ST depression induced by exercise relative to rest, slope of peak exercise ST segment, number of major vessels colored by fluoroscopy, results of thallium stress test, and diagnosis of heart disease. All attributes in this dataset are numeric values that either measure a variable or use an integer to indicate a level or category.

In addition to containing additional data that was determined to provide no meaningful value, the full dataset also uses different methodology for some variables. For example, the attribute for the diagnosis used the values 0,1,2,3 and 4 to measure the degree of heart disease. This application is more concerned with being able to distinguish the presence (value 1) of heart disease from the absence (value 0). The full dataset contains missing data; however, none of the missing data is found within the 14 attributes on which the application will focus.

## Objectives and Hypotheses

This application has two simple objectives: to analyze the factors that contribute to heart disease and to make a prediction about whether a patient has heart disease. This will lead to a conclusion that a small number of the 14 attributes used in the dataset have a strong correlation with heart disease. The hypothesis is that if a record in the dataset contains one specific combination of attributes, then the target data will be positive. It is highly improbable that the predictive model will be able to achieve near 100% overall accuracy. This is going to be due to how strongly certain attributes correlate with heart disease. For example, if 80% of records showing the presence of typical angina had a positive diagnosis, each record with that attribute would likely have a positive prediction, regardless of the values of the 13 other attributes. An outcome that is both beneficial and realistic would be the identification of combinations of attributes that results in a positive diagnosis 100% of the time.

## Methodology

DSI LLC typically employs the Agile methodology. Agile promotes developing software incrementally and iteratively. Each iteration utilizes every step in the development process. This means that small parts of the application will be designed, developed and tested before the full application is completed. Each iteration will add new features and functionality to the application. There are several benefits of developing software utilizing this methodology. First, software can be deployed rapidly with functional pieces of software available earlier. This facilitates implementing customer feedback and more easily allows changes. Additionally, because each iteration involves testing, bugs are caught earlier and are easier and less expensive to fix.

This application is a prime candidate to be developed in an agile environment because the application will consist of multiple features which can function independently of one another. For example, the first iteration will likely create the Dash app, create the necessary file structure and create the necessary files for the layout of the application. At this point, the application will be fully functional and will be viewable from a browser. All code created up to this point will be tested and verified. The next iteration will develop the part of the application which will download the CSV, clean the data and perform the train/test split for future use. Once again, when the second iteration is completed, the application will be functional and can be viewed from a browser. The third iteration will create the descriptive model and the fourth will create the predictive model. In order to maximize efficiency, we will also utilize paired programming. In paired programming, developers work in teams of two with one writing code while the other simultaneously performs code reviews. The result of this type of programming is developing high quality code more efficiently.

## Funding Requirements

The only cost associated with this project is developer time. There are no licensing fees because Python and all associated libraries are free and open source. The application will be developed using PyCharm, which has also been provided at no charge. The app will be hosted on a Heroku free account. The free account provides 1 Dyno (an instance of the application) and approximately 1GB of storage. This will meet current specifications, but if there is a future need for expansion, it will require approximately $25 per month for hosting services.

The total cost for developer time is based on the estimate of three developers (2 programmers and 1 data scientist) working for a total of 75 hours each. Additionally, there will be a UX designer working on the project for an estimated 15 hours. We charge a flat rate of $100 per hour for developer time. This brings the total cost for the project to $24000.

## Stakeholders Impact

We have identified two primary stakeholders who will be impacted directly by the completion of this project. The first group is healthcare professionals, who will obtain the ability to perform new research into trends and relationships between attributes and heart disease. The second group is members of the community who are at risk or who have been diagnosed with heart disease. This group will gain the ability to research the condition, determine their risk factors and analyze precisely which attributes contributed to that risk.

## Data Precautions

The dataset contains patient data, so it will fall under the purview of the Health Insurance Portability and Accountability Act of 1996 (HIPAA). HIPAA prevents patient health information from being disclosed without the express knowledge and consent of the patient (hhs.gov, 2015). The raw dataset contains identifiable information such as patient name and patient ID number, as well as sensitive personal information such as patient SSN. All of the attributes containing sensitive or identifiable information had been altered to generic values by the creators of the dataset and were also subsequently removed during the cleaning process.

According to the US Department of Health and Human Services, patient data may be used and distributed without the consent of the patient if it does not contain any readily identifiable information. There are two paths to ensuring compliance with deidentification. The one we have chosen is known as the safe harbor method. The safe harbor method requires that each of the following identifiers be removed: names, any geographic subdivision smaller than state, all elements of dates except for year, phone numbers, vehicle identifiers, fax numbers, device identifiers, email addresses, URLs, SSNs, IP addresses, medical record numbers, biometric identifiers, health plan numbers, full-face photographs, account numbers, certificate/license numbers or any other unique identifiers. All of these identifiers have been removed from the cleaned dataset and have not been stored by any means that would allow any patient to be re-identified. This leads to the conclusion that the application’s use and distribution of patient data is in compliance with HIPAA.

## Developer’s Expertise

All of the developers working on the project are college graduates and have been working in the computer science field for at least 2 years. The two programmers that will work on the project have experience using the Python language and are familiar with the libraries that will be utilized. Our data scientist has extensive experience in the libraries and tools that will be required. Additionally, the UX designer is experienced in web development using HTML and will be able to incorporate those skills into this project in the creation of the user interface. Each member of the team brings unique strengths that will be combined to create a product that will meet or exceed the client’s needs.

# Prompt B – Project Proposal

## Problem Statement

The problem that the community is facing is that there is a lack of readily available, easy to comprehend and up to date scientific data regarding heart disease. Community leadership is understandably concerned with the sudden increase in diagnoses of heart disease as well as deaths resulting from it. As a company that specializes in bioinformatics research, we believe these two issues have a causal relationship. Healthcare professionals are in need of a tool allowing rapid analysis of large amounts of data pertaining to heart disease. This tool needs to provide accurate analysis of attributes and exactly how various combinations of those attributes affect the diagnosis.

Healthcare professionals are not the only group affected by this problem. Members of the community who are at risk for the disease are also lacking in relevant information. At risk patients would also benefit from a tool that interprets and displays exactly how each attribute influences the target variable. An additional need facing the general public is the ability to take a patient profile, analyze the attributes and make a prediction about whether that patient likely has heart disease. While this prediction should never replace an actual diagnosis from a doctor, the patient will be given numerical data that shows risk and exactly which factors attributed to that risk.

## Customer Summary

As stated above, the users of this application will fall into two categories, healthcare professionals and patients. The primary need for healthcare professionals is data analysis and interpretation. This need will be met by the descriptive model of the application. The descriptive model will use logistic regression and provide an interpretation of the estimated coefficients. The result will be graphs which describe the magnitude and direction of the relationship between predictor and response variables. Patients will also benefit from this information. Although not everybody understands the exact meaning of logistic regression coefficients, displaying the information in charts will make it easier to read and comprehend. Additionally, patients can make use of the predictive model which will use a test set of data to make predictions about the presence of heart disease given a collection of features. This data will be presented in a table which can be filtered. This allows patients to search the data for a specific profile and see if the profile resulted in a positive or negative prediction.

This application will be developed using Dash. Dash is a Python framework for building web applications. It will be accessible through any web browser and will be easy to navigate for anyone familiar with the internet. All calculations are handled by the application itself and all data will be presented in visual formats making it easy to view and comprehend. This means that users of the application will not be required to possess any technical knowledge or advanced training of any kind.

## Existing System Analysis

This application is not being created in order to replace a system that is already in place. It is being created from the ground up to fill a need. The CSV file is hosted in a GitHub repository removing the need for a DBMS. Maintenance will be performed by the developers and the hosting service so there is nothing required from the client. The only requirement for users of the application is a computer with internet connection and a web browser.

## Data

The data is collected from a CSV file which has been downloaded from the UCI Machine Learning Repository at the following link: <https://archive.ics.uci.edu/ml/datasets/heart+Disease>. This data was collected by Andras Janosi, M.D. from the Hungarian Institute of Cardiology in Budapest, Willian Steinbrunn, M.D. from the University Hospital in Zurich, Switzerland, Matthias Pfisterer, M.D. form the University Hospital in Basel, Switzerland, and Robert Detrano, M.D., Ph.D. from the V.A. Medical Center in Long Beach. The raw dataset contains 76 attributes. Research has shown that some of these attributes hold no correlation with heart disease. The application will focus on the 14 that do:

1. Age
   1. Ratio scale
   2. The patient’s age in years.
2. Gender
   1. Nominal scale
   2. Binary data (0 = female, 1 = male)
3. CP
   1. Nominal scale
   2. Type of chest pain experienced by patient (0 = none, 1 = typical angina, 2 = atypical angina, 3 = non-anginal pain)
4. Trestbps
   1. Ratio scale
   2. Measurement of resting blood pressure on admission to hospital
5. Chol
   1. Ratio scale
   2. Measurement of serum cholesterol levels
6. Fbs
   1. Nominal scale
   2. Indication of whether fasting blood sugar was > 120 (0 = no, 1 = yes)
7. Restecg
   1. Nominal scale
   2. Electrocardiographic results (0 = normal, 1 = ST-T wave abnormalities, 2 = showing left ventricular hypertrophy)
8. Thalach
   1. Ratio scale
   2. Maximum heart rate achieved
9. Oldpeak
   1. Ratio scale
   2. ST depression induced by exercise
10. Slope
    1. Interval scale
    2. Slope of the peak exercise ST segment
11. Ca
    1. Interval scale
    2. Number of vessels colored by fluoroscopy
12. Thal
    1. Nominal scale
    2. Results of thallium stress test showing how well blood flows into heart (3 = normal, 6 = fixed defect, 7 = reversible defect)
13. Num
    1. Nominal scale
    2. Indication of whether patient has heart disease.

The raw file is delimited by whitespace, but there are extra newline characters. This makes it difficult for a program to read. The first step in data cleaning will be to remove the extra newlines and alter the file to change the delimiter from a whitespace to a comma. The second step is to drop the unused attributes. The remaining dataset, with only 14 attributes will not contain any missing data. The next step is to alter the data in the target column. The raw dataset expresses heart disease as an integer value from 0 to 4 to show degree. This is not of interest to this project, so the column will be altered by replacing values 1 through 4 with a 1. This will allow the application to focus on differentiating the presence of heart disease from its absence.

## Project Methodology

The project will be developed using the Agile methodology. This process involves the use of an iterative and incremental approach to development. This approach will allow us to gather continuous feedback and make incremental changes as necessary to ensure the application meets the client’s expectations. Each iteration through the process involves repeating several steps. The workflow process is outlined below.

* Plan
  + Step 1: requirements gathering. Meet with client and stakeholders to ensure needs and expectations are fully understood.
  + Step 2: planning meeting. Developers meet to discuss the day’s milestones and agendas and what will be required to meet those. Tasks are listed and organized, and an overall plan is formed.
* Build
  + The coding of the project utilizes paired programming. One developer writes code while another simultaneously reviews. Open communication is encouraged and the two switch places regularly to maximize efficiency.
* Test
  + Each completed unit undergoes rigorous unit testing. When multiple units have been completed the team will perform integration tests to make sure the newly completed unit has not caused any unexpected bugs in the application as a whole.
  + Testing will also consist of user and acceptance testing. A representative of the client and/or stakeholders will test the application from a usability standpoint as well as make sure that progress meets expectations. Any feedback gained from this step can be used in the next iteration to make changes, as necessary.
* Review
  + This step consists of another meeting to discuss what went right, what went wrong and what changes need to be made for the next iteration. A plan is made for any bugs found during unit or integration testing or any changes that need to be made as a result of user or acceptance testing.
* Launch
  + Each iteration through the process has a goal of creating a functional piece of software by adding additional features and functionality to the launch from the previous iteration. This means that at the end of each iteration the client and stakeholders have the chance to view the current deployment of the application. This allows for testing of servers and hosting services as well as provides additional opportunity for usability and acceptance testing.

## Project Outcomes

* Project deliverables
  + Schedule
    - Will give overall timeline of project and display details of how developer time is being utilized.
  + Project reports
    - Reports showing overall progress and discussing milestones and goals reached
  + Meeting minutes
    - Notes from meeting discussions and client feedback.
* Product deliverables
  + Application
    - Data product web application
  + Source code
    - All code files used to clean data and create application
  + Datasets
    - Dataset used for the application

## Implementation Plan

The application will be developed using an agile methodology. This means that even though we will not utilize a phased deployment, each iteration through the process has a launch step. At the end of each iteration, the application will be in a functional state and made available to the client and/or any stakeholders who wish to test its usability. The first iteration through the development process will create the file structure, all files and the Dash app. Due to the utilization of paired programming, the code will be given a code review at the same time it is being written. After the build phase, the application will undergo unit testing. At the end of this iteration, the application as it exists (even though at this point it will essentially just be an empty UI) will be made available to usability testers.

Additional iterations will follow the same process. The code will be written, reviewed, undergo unit and integration testing and then be made available for usability testing. This continuous testing method will allow for discussion of feedback in the following day’s meeting and necessary changes to be made with relative ease. The overall implementation strategy is for iteration 1 to create the dash app, files and file structure, iteration 2 to create the code to download the dataset, clean the data and perform a train/test split, iteration 3 will create the descriptive model and iteration 4 will create the predictive model.

After the full application has been launched, any updates or user driven changes can be implemented through alterations to the source code offline. This code can then be pushed to the GitHub repository where it will automatically redeploy on Heroku.

## Evaluation Plan

The application will be evaluated based on the results of the final acceptance testing. This will occur after the final iteration through the development process has been completed, but before the full product is launched and the project is officially closed. There are two steps to this process: a determination about whether exit criteria have been met and a “go/no go” meeting.

The first step requires the answer to a question many software companies struggle with: when should testing be stopped? The answer to this question for DSI LLC is that testing is concluded when the pre-defined exit criteria have been met. These criteria will be defined in the initial meeting but typically involves the following requirements:

* Desired defect rate has been reached
* All high priority defects have been fixed
* All medium priority defects are closed or have a workaround in place.
* Test case coverage over 95%
* Test case pass rate over 95%
* Failed test cases (under 5%) must be low priority
* Additional criteria defined at project launch meeting

The second step is a “go/no go” meeting. This is a final meeting with the client and representatives of relevant stakeholders. This meeting will confirm that testing exit criteria have been met, the application is fully functional, and all requirements have been met. This is the last chance for change requests from the client prior to launch. This step concludes with acceptance testing. In the final acceptance testing, the client will verify that the application meets all needs and performs as expected. At this point, the client will take ownership of the application, the source code and all related tools and data.

## Resources and Cost

As discussed, the project will be developed using software that is free and open source. The PyCharm IDE has been provided at no cost. Python and all associated libraries are open source. DSI LLC is already in possession of computers with Python and up to date browsers installed. This means there is no cost for software and there are no fees for licensing.

DSI LLC has possession of all hardware necessary to complete this project. Additionally, all hardware associated with hosting the application is the responsibility of Heroku. The only requirement for using the application is a web browser, so the client or any prospective users will not be required to obtain any new hardware.

The only costs associated with the project are human resource costs. DSI LLC charges a flat rate of $100 per hour for developer time. The total development time is estimated to be 240 hours (3 developers working for 75 hours each and 1 UX designer working for 15 hours). This brings the total cost of the project to $24000.

## Timeline and Milestones

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Event** | **Start** | **End** | **Dependencies** | **Deliverables/Results** |
| Launch Meeting | 9/1/2020 | 9/1/2020 | n/a | Schedule, requirements |
| Iteration 1 | 9/2/2020 | 9/4/2020 | Launch meeting | Phase 1 application, meeting minutes, project report |
| Iteration 2 | 9/7/2020 | 9/9/2020 | Iteration 1 | Phase 2 application, meeting minutes, project report |
| Iteration 3 | 9/10/2020 | 9/15/2020 | Iteration 2 | Phase 3 application, meeting minutes, project report |
| Iteration 4 | 9/17/2020 | 9/21/2020 | Iteration 3 | Full application, meeting minutes, project report |
| Go/no go | 9/22/2020 | 9/22/2020 | Iteration 3 | Application, source code, datasets |

# Prompt D – Post Implementation Report

## Project Purpose

The purpose of this project was to address a lack of readily available scientific data to analyze heart disease. This information is crucial for healthcare professionals to analyze various attributes and determine precisely how they relate to heart disease. Additionally, at risk patients often rely too much on anecdotal evidence rather than actual data.

The project addressed this issue by creating both a descriptive model and a predictive mode. The descriptive model utilized logistic regression with interpretations of coefficients. These coefficients were placed in a bar chart in order to facilitate easy comparisons of their magnitudes. Additionally, with a basic understanding of how to interpret these coefficients as well as some basic probabilities and statistics, a user can calculate the effect of incremental changes in predictor variables on the response variable by using the logit function:

logit(p) = b0 + b1 \* X1 – C \* A + … = ln(oddsp)

ln(OddsHeartDisease|var=A) = b0 + b1 \* X1 – C \* A + …

ln(OddsHeartDisease|var=A+D) = b0 + b1 \* X1 – C \* (A + D) + …

where: C = the coefficient, A = the current value and D = the number of units of change

Cancel common terms by subtracting:

ln(OddsHeartDisease|var=A) - ln(OddsHeartDisease|var=A+D) = C \* D

convert from log odds to odds ratio:

(OddsHeartDisease|var=A+D) / (OddsHeartDisease|var=A) = eC \* D

These steps allow a user to determine the effect of increasing a predictor variable by any number of units on the odds of having heart disease. For example, if one wanted to find the effect of increasing the maximum heart rate achieved during exercise (variable ‘thalach’) by 10, the above steps would be used.

C = .02927544

Odds ratio = eC\*D = e.02927544 \* 10 = 1.34

Increasing the maximum heart rate achieved during exercise by 10 bpm increased the odds of being diagnosed with heart disease by 34%.

The predictive model of the application used a random forest classifier to analyze the attributes of a patient and made a prediction about whether that patient had heart disease. This information was placed in a table, which can be filtered in order view profiles resulting in either positive or negative predictions. This table can also be sorted to determine at what values attributes become a concern.

This application is hosted online and available to anyone with a web browser. This makes it easily accessible and readily available for any interested parties. The charts have a functionality to display additional data upon scroll over, which allows for precise calculations. Additionally, because the data is downloaded from a GitHub repository, it will be easy to update in the future by simply replacing the outdated file with the new one.

## Datasets

The dataset used for the application was a file downloaded from the UCI Machine Learning Repository at the URL: <https://archive.ics.uci.edu/ml/datasets/heart+Disease>. The unprocessed dataset was delimited by whitespace. The file contained additional newlines within instances that made the data hard to convert to a pandas dataframe. The first step in preparing the data was to read the file, remove the extra newlines, change it from a whitespace delimited file to a comma separated characters file and write it into a new file using the code:

with io.open('C://users/dwsou/downloads/cleveland.data', "r", newline=None) as fd:

f = open('C://users/dwsou/downloads/cleveland\_edited.data', 'w')

for line in fd:

words = line.split()

for word in words:

word=word.strip()

line2 = ','.join(words)

if words[-1] != 'name':

line2 = line2.replace("\n", "")

f.write(line2)

f.close()

fd.close()

The raw dataset contained 76 attributes and 303 instances. According to the dataset documentation, all published research to date has focused on a subset of 14 attributes.

Cleaning the data required three steps. The first step was to drop the columns which were not going to be used. The columns that were dropped consisted of data that either did not have a strong correlation with heart disease (such as date of EKG reading) or contained identifiable or sensitive information (such as name or social security number). Dropping this data not only increased the efficiency of the application, it also ensured compliance with the safe harbor method of the deidentification requirement of HIPAA. The next step was to add names to all of the instances to make them easier to read and process.

The third step was to convert the integer data (values 0 through 4) of the target column to binary data (0 to indicate absence of heart disease and 1 to represent the presence). Attempting to associate attributes with the various stages of heart disease had a negative impact on accuracy and made the information much more difficult to understand and interpret.

The text below is a sample of the first three instances from the raw dataset.

1 0 63 1 -9 -9 -9

-9 1 145 1 233 -9 50 20

1 -9 1 2 2 3 81 0

0 0 0 0 1 10.5 6 13

150 60 190 90 145 85 0 0

2.3 3 -9 172 0 -9 -9 -9

-9 -9 -9 6 -9 -9 -9 2

16 81 0 1 1 1 -9 1

-9 1 -9 1 1 1 1 1

1 1 -9 -9 name

2 0 67 1 -9 -9 -9

-9 4 160 1 286 -9 40 40

0 -9 1 2 3 5 81 0

1 0 0 0 1 9.5 6 13

108 64 160 90 160 90 1 0

1.5 2 -9 185 3 -9 -9 -9

-9 -9 -9 3 -9 -9 -9 2

5 81 2 1 2 2 -9 2

-9 1 -9 1 1 1 1 1

1 1 -9 -9 name

3 0 67 1 -9 -9 -9

-9 4 120 1 229 -9 20 35

0 -9 1 2 2 19 81 0

1 0 0 0 1 8.5 6 10

129 78 140 80 120 80 1 0

2.6 2 -9 150 2 -9 -9 -9

-9 -9 -9 7 -9 -9 -9 2

20 81 1 1 1 1 -9 1

-9 1 -9 2 2 1 1 1

7 3 -9 -9 name

The following text is a sample of the first three instances from the cleaned dataset:

age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal,target

63,1,3,145,233,1,0,150,0,2.3,0,0,1,1

37,1,2,130,250,0,1,187,0,3.5,0,0,2,1

41,0,1,130,204,0,0,172,0,1.4,2,0,2,1

## Data Product Code

The entire source code has been submitted along with this document and can be found in the app.py file. All files can also be found online at the following GitHub repository: <https://github.com/dsaovuilde/HeartDisease>. The code used to clean the data as well as a script used to optimize the machine learning algorithm are included in separate files as these scripts were only needed once.

After the data had been cleaned, the next step was to upload it to a GitHub repository and write the code to allow the application to read the data from a URL rather than reading a file from the disk. The URL for this dataset is <https://raw.githubusercontent.com/dsaovuilde/heart_disease/master/Data/Heart.csv>.

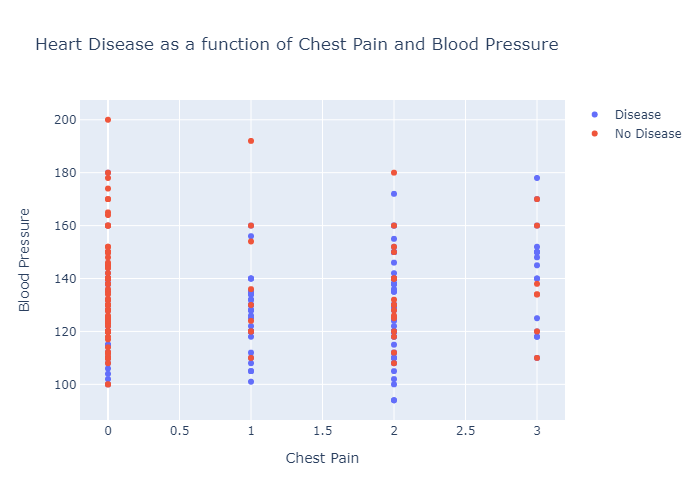
After the data was read into the application, it was converted into training and test data using the train\_test\_split function.

The next part of the code was the descriptive model. The first element in the descriptive model was a heatmap to color code the relationship between variables. The column of interest in this heatmap is the target data. The heatmap seems to indicate a strong correlation between chest pain and heart disease.

A picture containing drawing

Description automatically generated

The next element of the descriptive model consisted of two scatter diagrams depicting heart disease as a function of two variables which are widely known to contribute to heart disease. The first is heart disease as a function of age and maximum heart rate and the second is heart disease as a function of chest pain and blood pressure.

A screenshot of a cell phone

Description automatically generated

The next graph in the descriptive model was a bar graph of the logistic regression estimated coefficients. The coefficients are displayed in a graph to allow the user to visually interpret which features are most important.

A close up of a logo

Description automatically generated

These coefficients again indicate a strong correlation between chest pain and heart disease. The coefficients for age, blood pressure and maximum heart rate are relatively small, but these variables have large ranges of values. This means the maximum effect on the response variable can still be quite large.

The final element in the descriptive model was a tool allowing users to select one of three attributes and move a slider to see how incremental changes in that variable influence the response variable. For example, a user can select the radio button for blood pressure and move the slider up to see how much a small increase in blood pressure increases the odds of being diagnosed with heart disease.

The next part of the product code created was the predictive model. The first step was to choose a machine learning algorithm. We decided to select three models for testing and compare their effectiveness. The models chosen were logistic regression, convolutional neural network, and random forest classifier. Initial testing found the random forest classifier to be the most accurate, so this algorithm was chosen. The random forest classifier algorithm has multiple parameters and it is unlikely that simply choosing the defaults would have led to an optimal solution. The model was optimized using a random search algorithm. This code is not found in the application itself because it only needed to be used once (it could be needed in the future if the dataset is changed). The code can be found in the file called Optimization.py in the data folder. The script for hyperparameter optimization yielded the following results for best parameters:

{'n\_estimators': 733,

'min\_samples\_split': 5,

'min\_samples\_leaf': 4,

'max\_features': auto,

'max\_depth': 80,

'bootstrap': True}

After the algorithm was chosen and optimized, the next step in creating the predictive model was to fit the model to the training data and make predictions on the test data. In order to provide the user with as much information as possible, a column for the predicted variable as well as the actual result was added to the test data. This information was displayed in a table. This table can be sorted and filtered on any column. This means a user can search the table to analyze which sets of attributes led to positive or negative predictions and whether those predictions were accurate.

## Hypothesis Verification

This initial hypothesis for this project was that if a record contained a specific combination of attributes, then the result would be positive. The idea behind this is to find a specific subset of variables in this data that, when combined, will always lead to a diagnosis of heart disease. Searching through the test set of data showed that one such combination of variables was males with typical angina and a maximum heart rate greater than 120. The hypothesis was accepted.

## Effective Visualizations and Reporting

The visual elements in the application were arranged in such a way as to try to tell the story about what actually causes heart disease. The first element in this display was a heatmap showing the correlations between variables. This visual representation indicates stronger correlations between several variables beyond simply looking at the target. For instance, chest pain has a stronger correlation with blood pressure, which makes sense.

The second element in the visual display consisted of two scatterplots which analyzed heart disease as a function of two of the variables. The first was a function of age and maximum heart rate and the second was a function of type of chest pain and blood pressure. The first plot has an obvious negative trend. The older a patient is, the lower the maximum heart rate needed to be in order to indicate heart disease. A similar trend could be seen in the second graph. Patients experiencing no chest pain could have much higher blood pressures without having heart disease.

The next element was a bar graph showing the estimated logistic regression coefficients of each variable. These coefficients were displayed in a chart in order to allow users to easily discern the importance of features. Below the graph was a tool allowing users to choose a variable and make incremental changes with a slider and see how that change affected the odds of having heart disease.

## Accuracy Analysis

The effectiveness of the predictive model was displayed in a confusion matrix. There was a total of 61 records in the test set of data. Out of these 61 predictions there were 36 true positives, 19 true negatives, 1 false positive and 5 false negatives.

A screenshot of a cell phone

Description automatically generated

Using these numbers, as well as the accuracy score function, the predictive model was measured using three metrics: overall accuracy, sensitivity and specificity. The overall accuracy of the model was approximately 90%. The problem with accuracy being the sole metric for measuring the effectiveness of a predictive model is that it does not give a comprehensive report of how the model is performing. Accuracy is calculated by adding the total number of true positives and true negatives and dividing by the total number of predictions. This means that if only 10% of the patients in the dataset had heart disease, a completely worthless model that simply predicted negative every time would have an accuracy of 90% due to the true negative rate. This is why the model was also analyzed using the other two metrics.

Sensitivity measures a predictive model’s ability to accurately predict positive results. Specificity measures the model’s ability to correctly predict negative results. High sensitivity is especially important in medical diagnostics because false negative tests can have disastrous results for a patient. The model correctly predicted 36 out of the 37 positives making the sensitivity approximately 97%. The model correctly predicted 19 out of the 24 negatives, which made the specificity approximately 79%. These metrics not only allow analysis of the model’s performance, they also provide a starting point for future improvement and optimizations by analyzing incorrect predictions. The table below shows all of the false positives in the model. Two of them are older patients who have blood pressure over 130 and reached high maximum heart rates. Based on that information, a positive prediction makes sense, but the actual diagnosis was negative. It may be possible to analyze why the model predicted incorrectly in this instance, but that will have to be done in future updates.

A picture containing screenshot

Description automatically generated

False positives

## Application Testing

The application was developed using the agile methodology. This means that testing was done continuously as parts of the application were developed and new features were added. Early iterations through the development cycle created the user interface and Dash application. These features were then given unit tests, reviewed by the client’s onsite representative for acceptance testing and then launched in order to provide an opportunity for usability testing.

Additional iterations through the cycle added new features and functionality to the application. As these new units were created, they were given individual unit tests. After completion of unit testing, we performed integration testing to ensure that the newly added features did not introduce any bugs into the rest of the application. At the end of these iterations, the application underwent acceptance testing by the onsite representative and was launched to allow for user testing. The final iteration created the fully functional application and required an official acceptance in a “go/no go” meeting with the client.

## Application Files

All files can be found in a GitHub repository at the following link: <https://github.com/dsaovuilde/HeartDisease>. The main folder consists of the following files:

* Readme
  + A file outlining the project and providing information such as a link to the application.
* .gitignore
  + A file which tells GitHub which folders and files were not to be uploaded
* Procfile
  + This file contains information about how Heroku was going to launch the application. In this case it is a Gunicorn web server.
* requirements.txt
  + A text file which is required for Heroku to install the required dependencies
* runtime.txt
  + A file instructing Heroku to upgrade to python 3.7.3 (default is 3.6.x)
* app.py
  + python file containing all of the source code

There is also a subfolder named Data which contains the following files:

* Raw\_dataset.data
  + This is the completely unprocessed dataset downloaded from the UCI repository.
* Raw\_converted\_to\_comma\_separated.data
  + This file is the dataset after converting from a file with a whitespace delimiter to a file delimited by commas.
* Heart.csv
  + Dataset after all cleaning scripts have been executed
* cleaning\_scripts.py
  + code used to clean the dataset
* Optimization.py
  + code used to optimize the predictive model

## Installation and User’s Guide

Installation requirements are broken down into two sections: the guide for users who only want to see and interact with the information and the guide for administrators who will need to monitor application health and update the data. The only requirements for a user are an internet connection and a browser. The application has been tested on the following browsers:

* windows
  + Chrome
  + Firefox
  + Internet Explorer
* android
  + Chrome

The application and displays load quickly (depending on internet connection) and the response time for queries is nearly instantaneous. **Note:** if the application has been idle for an extended period of time, it will be placed in “sleep” mode and will take a longer time to load.

There are no login requirements due to lack of data sensitivity. Additionally, requiring users to create and account and login goes against the project goals of being readily available and easily accessible, and may discourage use of the application.

An administrator needing to update and monitor the application will need access to additional resources. To allow for application maintenance or updates, take the following steps:

* Ensure Python version 3.7.3 is installed on local machine
  + <https://www.python.org/downloads/>
* Install PyCharm IDE
  + <https://www.jetbrains.com/pycharm/download/#section=windows>
* Download Heroku CLI
  + <https://devcenter.heroku.com/articles/heroku-cli>
* GitHub account
  + Files can be forked from GitHub or manually uploaded from project files
* Heroku account
  + Deploying application and making changes will require login credentials

When logged into Heroku, navigate to the deploy tab and ensure the option to enable automatic deploys is currently selected. This option redeploys the application automatically when new changes are pushed to GitHub. The only requirement to update the application is to open it from PyCharm, make the necessary changes (or manually upload a file such as a new Heart.csv) and push the changes to GitHub. When this is done, the application will automatically redeploy.

It is recommended that when any changes are made to the csv file, the logs are checked to see if the predictive model needs to be re-optimized (detailed instructions for accessing logs can be found below). If optimization is required, use the code found in Optimiztion.py and input the parameters to the instance of RandomForestClassifier.

Although uptime and response time are primarily the responsibility of Heroku, the administrator can still gain useful information from monitoring logs. In the case of errors or crashing, the logs will give a stack trace to help find the location. These logs also show the program’s output (the program prints the accuracy metrics) which can be used to determine any detrimental effects on the application’s health or accuracy caused by altering the source code. Currently, the only way to obtain these logs is by using the Heroku CLI by taking the following steps:

* Open the command line interface
* Enter following command
  + heroku login
    - enter login credentials and return to CLI
* Enter following command
  + heroku logs --app=h3art
    - optional
      * add --tail to show only last 100 lines of log

If the application has crashed, the stack trace is printed at the bottom and can be used to find the cause. If changes have been made, the accuracy metrics will be printed at the bottom of the log and can be used to discern any negative effects resulting from these changes.

A screenshot of a computer

Description automatically generated

**Sample of log output**

Using the application is quite intuitive and the instructions can be found in the following user guide:

* Navigate to the application’s site: [www.h3art.herokuapp.com](http://www.h3art.herokuapp.com)
* If desired, graphs can be interacted with by zooming, panning, or obtaining more data via mouseover
  + Placing your pointer on a graph will display a toolbar
    - This will allow panning or zooming
    - There is also a button to download and save the graph
    - Click the home button to reset the graph back to original form
  + Placing the pointer on a point on a graph will display additional data
* To use the tool for showing effects of changing variables:
  + Select a variable and move the slider to the left or right as desired
    - The label above the handle will display current value
    - Output underneath the slider will show the effect
* To sort the values in the table:
  + Click the up or down arrow to sort by ascending or descending order
  + Click the same location again to reverse the order
  + Click a third time to reset
* To filter the values in the table:
  + Type number to search for in filter data boxes below column names
  + To reset filter, clear the number in the box and press enter

## Summation of Learning Experience

My formal education provided an excellent foundation for the knowledge and skills used in this project. My education included multiple classes which involved learning general concepts and best practices which can be applied across multiple programming languages. I also created a project using Python and PyCharm which gave me a solid understanding of Python syntax. Going into this project with a familiarity with Python was definitely a plus.

When I began working, I had no experience with data science, machine learning algorithms or nearly any of the libraries that I ended up using. For me, this project ended up being first and foremost a research project. I needed to prove that I could learn new concepts and tools on my own and expand on what I already knew. I began by reading the documentation and watching tutorials about Pandas. My next approach was to practice what I learned. I opened a Jupyter Notebook, downloaded a csv and created my first dataframe. I took the same approach with pyplot, plotly and several other data science libraries.

My next step was to research machine learning algorithms. Before I started any coding, I wanted to use a convolutional neural network as my predictive model. However, when I tested model accuracy, a random forest classifier yielded the best results. In the course of my research I learned two methods for hyperparameter optimization: grid search and random search. I used the random search algorithm for this project and was able to improve the overall accuracy of my predictive model by several points.

The last thing I needed to learn was how to take a python application and actually launch it. When you search a question of this nature, you will be pointed towards Flask, Jupyter and Voila. I initially planned to use these tools, but then I discovered Dash. I was intrigued by how simple yet powerful it was. I began to experiment with Dash and ultimately decided that it was the framework I wanted to use.

The largest struggle for me was deploying the app on Heroku. There is step-by-step documentation on how to do this, but it is outdated. Recent instructions about deploying on Heroku are written for Flask (the Dash framework is built on Flask). These can be difficult to follow since Dash uses a slightly different syntax. After multiple unsuccessful attempts to deploy, I noticed in a mistake in the logs. Heroku was installing Python version 3.6.x instead of s using version 3.7.3. I added the runtime.txt file to change the version to 3.7.3 and the application successfully deployed but would crash. By reading through the logs, I was able to trace the issue to the app callback for the data table. I fixed the callback and ensured the table still functioned correctly and redeployed successfully.

The unsuccessful launches provided other learning opportunities for me. I ended up creating several smaller applications and deploying them in order to figure out what code was causing the crashes. The quickest way to accomplish this was by using a git and a command line interface, which I had little to no experience with. While I still have much to learn in that area, the skills I gained by launching multiple applications in this manner will be beneficial moving forward.

# Sources

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository. Irvine, CA: University of California, School of Information and Computer Science. http://archive.ics.uci.edu/ml

Guidance Regarding Methods for De-identification of Protected Health Information in Accordance with the Health Insurance Portability and Accountability Act (HIPAA) Privacy Rule. (2015, November 6). <https://www.hhs.gov/hipaa/for-professionals/privacy/special-topics/de-identification/index.html#rationale>

Kirkwood, Michelle. (2019, December 31). *Top heart disease and stroke research advances in 2019*. <https://newsroom.heart.org/news/top-heart-disease-and-stroke-research-advances-in-2019>

Sklearn.ensemble.RandomForestClassifier. (2020). <https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html>