Cardiovascular Disease Prediction and Risk Analysis Using Supervised Machine Learning

Albina Cako
Faculty of Engineering and Architectural Science
Biomedical Engineering
Ryerson University

Jeffrey Rezazada
Faculty of Engineering and Architectural Science
Biomedical Engineering
Ryerson University

I. INTRODUCTION

Cardiovascular disease (CVD) is the leading cause of illness and deaths globally. In 2013, CVD caused 17.3 million deaths, which made up 31.5% of total deaths worldwide [2]. It is projected that 91.2 million people (43.9 %) of the total population of the United States will have some sort of CVD by 2030 [1]. The risk of developing CVD is dependent on many risk factors, including modifiable and unmodifiable ones. The most common modifiable risk factors are tobacco use, high blood pressure, high glucose levels, obesity and physical inactivity. Unmodifiable risk factors include age, ethnicity and gender [3] [4]. As CVD is increasing globally, primary prevention is very important. Using prediction rules, risk scores of modifiable risk factors is a primary prevention strategy that has been used for many years. However, this is usually done through laboratory work and other medical procedures that can usually predict each risk factor separately and can take days to obtain results [3].

CVD is a worldwide health crisis. New and faster prediction tools are needed to help medical doctors detect the risk of CVD in patients before it becomes a chronic condition. In this study we created a CVD prediction application using machine learning that will allow medical professionals to be able to quickly assess the risk of developing CVD for a patient. This application can help medical professionals make better decisions in referrals for cardiologists, resource distribution and disease prevention. In addition, in low-income areas or areas where there are shortages of doctors, the application can help make quick predictions from the medical office. This would allow for physicians to swiftly assess and provide the patient with some recommendations for the modifiable risk factors, as changes to these risk factors can have a large impact on reducing their chance to develop CVD or disease progression [6].

There has been a similar research study that created a machine learning classification model for prediction of CVD probability by using data from 423,604 participants, as well as 473 features [5]. However, having many features would lead to long evaluation times for the medical doctor and would not be very user friendly if deployed in an app. It would take the doctor hours to input the patient data in order to make a prediction. In this project we mainly used the major modifiable and unmodifiable CVD risk factors to create our CVD prediction model and application. It is important to note that we used the CRISP-DM methodology to do our project, taking the project from problem understanding to deployment.

II. PROBLEM STATEMENT AND DATASET

The problem targeted in this project is CVD prediction using the major modifiable risk factors of CVD. An application was built to be used as a quick detection guide for CVD, by giving the risk for CVD, in percentage. The purpose of the app is to be used by medical doctors to detect those in danger and the probability of a patient developing or having CVD. If the risk can be detected beforehand, then patients can be advised to change those modifiable risk factors and doctors can inform their patients how much those changes would reduce the chance of developing CVD. Changing the modifiable risk factors can have a large impact on reducing CVD development. For example, a research study from the United States showed that reduction on modifiable risk factors accounted for a 44% of the decline in CVD related deaths [6].

The dataset for this problem was selected from Kaggle. It is a dataset gathered from 70,000 patients during their medical examination [7]. The original dataset contained 13 columns and 70,000 rows. The

12 features in the dataset were ID, Age, Gender, Weight, Systolic Blood Pressure, Diastolic Blood Pressure, Cholesterol, Glucose, Smoking, Active (physical activity). The last column (Cardio) noted whether the patient had cardiovascular disease or not. The Cardio column represents the target variable.

The ID column indicated the patients' ID numbers, and the age column indicated the patients' age in days. The columns 'Weight' in kg, 'Height' in cm, 'Systolic Blood Pressure' in mmHg and "Diastolic Blood Pressure' in mmHg had defined values. 'Gender' was listed in binary (1 = female, 2 = male). 'Smoking', 'Active' and 'Cardio' were also listed in binary (0 = no, 1 = yes), while 'Cholesterol' and 'Glucose' had 3 categorical values (1 = normal, 2 = high, 3 = very high). When analysing the target variable, it was seen that the dataset was balanced.

III. METHODS AND MODELS

Data exploration was performed to understand the data and perform the necessary data cleaning, using the Data_exploration.py and clean_data_analysis.py files. The 'data.describe()' function of 'pandas' library [8] was used to explore the general trend of the data including the min, max, mean, and standard deviation. The describe feature revealed that there were outliers in the data, including negative values for both blood pressure columns, which is erroneous. We plotted a boxplot for the weight, height, systolic blood pressure and diastolic blood pressure features, since they had defined values. Below is an example of the boxplot for systolic blood pressure. As seen in the boxplot, there were multiple

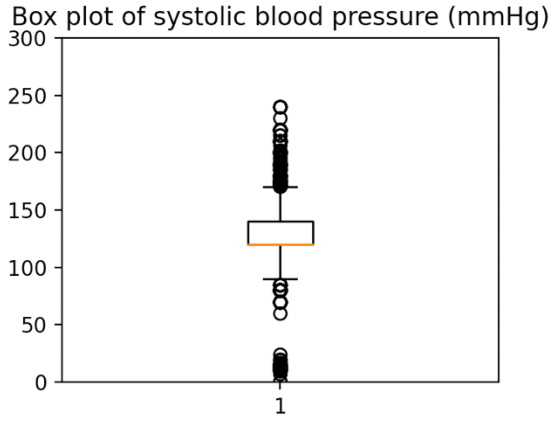

Fig. 1 Boxplot showing outliers for systolic blood pressure.

outliers that could affect our modelling. We decided to remove all outliers from our dataset by removing any patients outside of the ranges set by the whiskers of the box plot. These were the following ranges chosen per each categorical variable containing outliers: height (140 – 180 cm), weight (40 – 110 kg), systolic blood pressure (90 – 170 mmHg) and diastolic blood pressure (70 – 100 mmHg). After this change, the overall data sample was reduced to 62103 records, a loss of only 11%. Medical research was consulted for the blood pressure ranges to ensure that out of normal ranges that were included in the data were chosen correctly [9]. Finally, the data was cleaned by removing the ID column and changing a few columns to make them binary and consistent with the rest of the data. For the 'Age' feature, the age of the patient was changed from days to year. This was done to make the prediction app more user friendly in the future, as most patients are assessed by age in years (cardio_train_updated.csv). For the 'Gender' feature, female was kept as 1 and male was given a value of 0. In the cholesterol and glucose columns 1 (normal) was turned into 0, while 2 and 3 (high and very high) were turned into 1, representing higher than normal range values. The data was saved into a new file (cardio_train_clean_featureselection.csv) and was ready for modelling.

Prior to modelling, the data was assessed whether it was balanced by checking whether the target variable column (Cardio) had equal amounts of both 1 (yes) and 0 (no) values. As seen in Figure 2, the data was very balanced, having 31181 patients with no CVD and 30922 with CVD. The data was

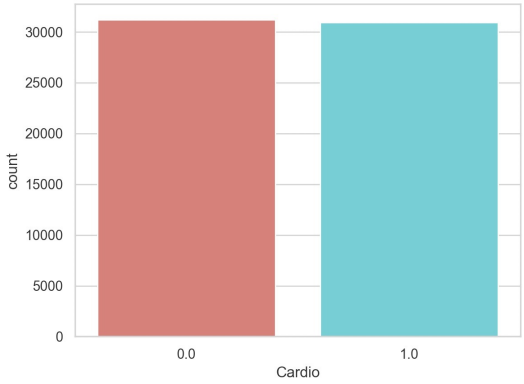

Fig. 2 Target variable Cardio distribution of yes (1.0) and no (0.0) values, representing the amount of population who have or do not have CVD.

checked for distribution of features. We noticed that it was not equal, especially for the 'Alcohol' and 'Smoking' features, where only a small portion (~6%) of the population smoked and/or drank alcohol. However, we decided to keep all features for initial models in order to observe how it affected our model performance in the application. Figures 3 and 4 show a visual representation of how the proportions of each feature correlate to the patient being diagnosed with CVD or not.

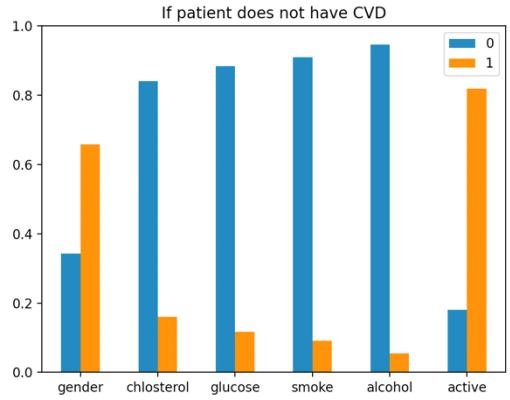

Fig. 3 The distribution of no (blue) and yes (orange) for the individual features, for people who do not have CVD.

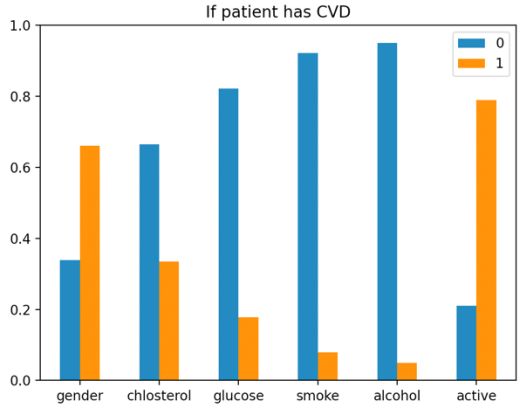

Fig. 4 The distribution of no (blue) and yes (orange) for the individual features, for people who have CVD.

It is most noted in Figures 3 and 4 that cholesterol and glucose levels increase in patients with CVD.

In this project, seven different classification models were used to train the data, using both parametric and non-parametric algorithms. These included Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Logistic Regression, Gaussian Naïve Bayes, Multinomial Naïve Bayes, Support Vector Machine (SVM) and Random Forest. Differing models were used to test the dataset to observe which model would perform the best. We wanted to include a non-parametric model such as random forest to see how decision trees perform in a classification problem. The data was divided into 75 % train and 25 % test split.

We ran a correlation matrix to determine if any of the features were correlated. R^2 metric was used to determine if the features were correlated. The highest correlated features were Systolic and Diastolic Blood Pressure with an R^2 value of 0.51. Usually, a value of 0.51 is a small correlation, however, since the two blood pressure values are linearly correlated, we decided to only use systolic blood pressure in our modelling. We removed the diastolic blood pressure feature from our data. The rest of the features showed no significant correlation.

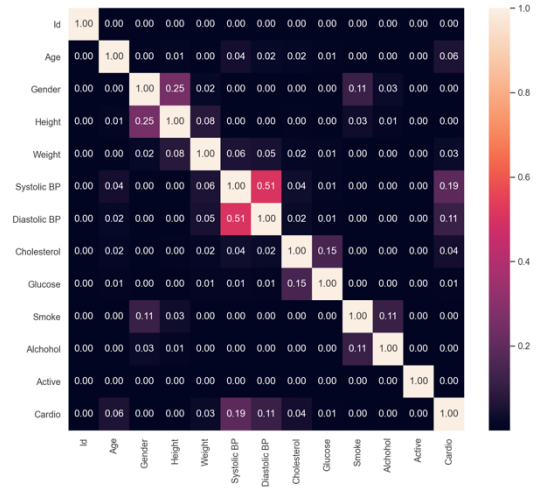

Fig. 5. The correlation matrix for the features of the dataset the highest correlation is seen between systolic and diastolic blood pressure features.

Ten features were then selected for initial modelling: age, gender, height, weight, systolic blood pressure, cholesterol, glucose, smoke, alcohol, active. The initial models were run prior to tuning and a confusion matrix was used as the performance evaluation metric. By running a confusion matrix, we obtained the accuracy of each model, but most importantly, the precision and recall. The recall, which represents the false negatives, was chosen as the most important metric for our problem, as false negatives in medical diagnostics should be minimized. If a patient has the disease but our model predicts they do not, it can put their lives in danger. Thus, lower false negative rates result in a better model.

The top 4 models, ranked using the above metrics, were logistic regression, LDA, random forest and SVM. These models were tuned to achieve higher accuracy and recall score, in order to optimize the model to the dataset. To tune the models, we used k-fold Cross Validation and hyperparameter tuning. The final parameters used by each tuned model were saved and will be displayed in the results section. The final models were saved using the 'joblib' library and the best model was deployed in the application.

After running and tuning all the models, we wanted to try to incorporate the high and very high distinguishing values of cholesterol and glucose into our models, using one-hot encoding. We first created a new dataset, where we split the glucose and cholesterol columns in 3 binary columns each: 'Glucose Normal', 'Glucose High', 'Glucose VeryHigh' and 'Cholesterol Normal', 'Cholesterol High' and 'Cholesterol VeryHigh', saving the data as 'cardio_train_clean_1hot.csv'. All data exploration and model metric analysis were done again to the one-hot dataset to compare the metrics to our binary initial dataset.

The top four models were again tuned using the same method stated earlier to optimize the each of the top models. The final models were saved using 'joblib'.

IV. RESULTS AND DISCUSSIONS

The initial section will discuss the results from the initial cleaned data without using one-hot encoding. The untuned models that performed best were logistic regression and random forest. We chose these two based on highest accuracy and recall, respectively. The top 4 performing models are shown below:

Table 1. Initial Model Performance

Model	LR	LDA	RF	SVM
Accuracy	0.717	0.715	0.713	0.706
Precision	0.745	0.745	0.755	0.755
Recall	0.655	0.644	0.665	0.600

Top 4 performing models (LR = logistic regression, LDA = Linear Discriminant Analysis, RF = random forest, SVM = Support Vector Machines)

As seen above, the logistic regression performed with highest accuracy, while random forest performed with the best recall score. We decided to tune the top 4 models using 3-fold cross validation and hyperparameter tuning.

After tuning, two models outperformed the rest, random forest and logistic regression. As it is observed in table 2, the random forest was the highest performer in the accuracy and in the recall values. The logistic regression was tuned by altering the C parameter as it is the only parameter tuned with hyperparameter tuning, while random forest was tuned by random searching the best parameters that improve recall score. The final logistic regression model used the parameter C = 0.1, while random forest used the following parameters: n_estimators = 1000, min_samples_split = 10, min_samples_leaf = 2, max_features = sqrt, max_depth = 10, boostrap = True.

Table 2. Tuned Model Performance

Model	RF	LR
Accuracy	0.734	0.717
Precision	0.762	0.744
Recall	0.675	0.656

Top 2 performing models (RF= random forest, LR = logistic regression)

The same procedure was performed with the one-hot dataset. Models tended to perform slightly better in the one-hot dataset. Table 3 lists the results of the initial models. After hyperparameter tuning using the one hot dataset, the random forest was chosen as the best performer due to its high accuracy and high recall value.

Table 3. One-Hot Model Performance

Model	LR	RF	LDA	SVM
Accuracy	0.719	0.718	0.718	0.706
Precision	0.745	0.738	0.750	0.756
Recall	0.655	0.668	0.642	0.598

Top 4 performing models (LR = logistic regression, RF = random forest) using one hot encoding dataset

For deployment, we constructed an application with all of our features. We initially chose to use the random forest model, using the one-hot implementation, as it had the highest accuracy and recall value. When deploying the one-hot random forest model, we realized that the implementation of one-hot-encoding was done improperly as the single feature was split into three separate features (one-hot encode for 1,2,3,), instead of one feature containing a 3-unit vector. Mapping the single feature to three separate features leads to bias in the data towards this feature, in this example, 'Cholesterol' and 'Glucose', leading to a biased model.

We decided to do our deployment using the initial dataset that did not include one-hot encoding, which groups the high and very high together. This was done to simplify the grouping, as the data does not specify what the difference between the 'high' and 'very high' categories is. Once we deployed the random forest in the initial dataset, we realized that the random forest model did not perform well in the application. It would have quite erroneous predictions, as when no risk factors were present, and the numerical data was set to 0 (the default setting for the app), the risk of CVD was seen to be approximately 20%. Therefore, we decided to test the performance of the logistic regression model, since it also had a high accuracy and recall value. The logistic regression performed well on the application, having near 0% probability when the application was in the default setting.

After further testing, we noticed some issues with some of the features working poorly when deployed in the application. We decided to go back and look at our data distribution of the features. This data can be found at the bottom of the 'Data_exploration.py' file. First, the 'Cholesterol', 'Glucose', and 'Active' features were analysed, and it was seen that they had an acceptable data distribution, with at least 20/80 data distribution between the binary features. Although a 50/50 distribution would be ideal, rebalancing the data further from 20/80 would result in a large loss in the data size. A larger dataset would have been ideal to overcome the issue of distribution, however this was not possible for this project.

When using the alcohol and smoke features on the application, it was seen that two features were not performing well on the application. An increase in alcohol consumption and smoking feature would lower the CVD risk. Based on our research, these values do not match the latest scientific evidence as both these are considered important risk factors in CVD development [5]. By further looking into the data distribution, in the alcohol and smoke columns we noticed that only 5 % and 8% of the total patients were alcohol consumers or smokers in the dataset. This could be causing bias in our data and overfitting, which is why the application was giving inaccurate results. Therefore, for our final deployment model we decided to remove the features 'Smoke' and 'Alcohol'. We reran and tuned a final logistic regression model using 8 total features: 'Gender', 'Age (years)', 'Height (cm)', 'Weight (kg)', 'Cholesterol levels (normal, high)', 'Glucose Levels (normal, high)', 'Active (yes, no)', 'Systolic BP (mmHg)'. The final model had an accuracy of 72% and a recall score of 66%. This is the model that has been deployed in our application.

As the implementation of the one-hot encode was improper, the next step would be to redeploy the model using the proper implementation of one-hot encoding and evaluate the metrics again. Also, the grouping of the features 'Cholesterol' and 'Glucose' mixed the two groups 'High' and 'Very High', could have led to some inaccuracy, as these two could have differing effects on the model's probabilities. When looking at the overall data, it is also unclear what the deciding thresholds for the categories are; what is the difference in the normal, high, and very high values. This should also be explored so the user for the application can have a better understanding of how to input the patient's data. Finally, the next steps for the application would be to implement a way for the user to check how changing a certain criterion modifies the risk for CVD. For example, if the patient reduces their weight, how does the probability for CVD change. This would allow the doctor to explain to the patient how to reduce their overall risk, all within the app.

V. DEPLOYMENT

We designed a prediction application using 'Dash App' platform to deploy our machine learning model. The app can be found by running the app.py file and clicking on the link provided. The application will yield a percentage of CVD risk

based on the final logistic model selected based on the 8 features. The application is colour coded, showing the risk based on the risk percentage (green, yellow, red). Green represents CVD risk of less than 50%, yellow risk between 50% and 75 %, while red risk above 75%. For further instructions on the user guide for the application refer to the 'ReadME' file is available on Github [1]. It is important to note that this app is only 72 % accurate and it can only be used as a guide. It does not replace medical advice.

VI. IMPLEMENTATION AND CODE

The full code for this dataset can be found in the GitHub repository listed as the 1st reference [1]. Several python packages were used to do our exploratory analysis and data cleaning including 'NumPy', 'seaborn', 'matplotlib' and 'pandas'. For model training and tuning we used the 'scikit-learn' package and for saving the models we used the 'joblib' package. Finally, for deployment we used the dash package, which allows for deployment of a python machine learning model into an application. For most of the coding, we used prior knowledge of using the python libraries. However, we did use some resources and source code regarding our project. For data exploration and analysis, we used a few resources to understand the exploration process [10] and we used a source code from Kaggle to plot the correlation matrix [11]. For one-hot encoding we obtained the information regarding on how to do one hot encoding [12], however, we used our coding knowledge to build the code. For the modelling we used the scikit-learn library [13]. Finally, for deployment we used a tutorial series as a guide on YouTube on how to use the Dash App. A few source codes were taken from the tutorial [14].

VII. REFERENCES

[1]https://github.com/bioalgorithm/CVD_prediction

[2] Benjamin, E. J., Blaha, M. J., Chiuve, S. E., Cushman, M., Das, S. R., Deo, R., ... & Isasi, C. R. (2017). Heart disease and stroke statistics—2017 update.

[3] Gaziano, T. A., Bitton, A., Anand, S., Abrahams-Gessel, S., & Murphy, A. (2010). Growing epidemic of coronary heart disease in low- and middle-income countries. Current problems in cardiology, 35(2), 72–115. https://doi.org/10.1016/j.cpcardiol.2009.10.002

[4] Pandya, A., Gaziano, T. A., Weinstein, M. C., & Cutler, D. (2013). More Americans living longer with cardiovascular disease will increase costs while lowering quality of life. Health Affairs, 32(10), 1706-1714.

[5] Alaa, A. M., Bolton, T., Di Angelantonio, E., Rudd, J. H., & van der Schaar, M. (2019). Cardiovascular disease risk prediction using automated machine learning: A prospective study of 423,604 UK Biobank participants. PloS one, 14(5), e0213653.

[6] Mensah, G. A., Wei, G. S., Sorlie, P. D., Fine, L. J., Rosenberg, Y., Kaufmann, P. G., Mussolino, M. E., Hsu, L. L., Addou, E., Engelgau, M. M., & Gordon, D. (2017). Decline in Cardiovascular Mortality: Possible Causes and Implications. *Circulation research*, *120*(2), 366–380

[7] Ulianova, S. (2019, January 20). Cardiovascular Disease dataset. Retrieved October 17, 2020, from https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

[8] User Guide. (n.d) Retrieved from https://pandas.pydata.org/docs/user_guide/index.html

[9] InformedHealth.org [Internet]. Cologne, Germany: Institute for Quality and Efficiency in Health Care (IQWiG); 2006-. What is blood pressure and how is it measured? 2010 Jun 24 [Updated 2019 May 23]. Available from: https://www.ncbi.nlm.nih.gov/books/NBK279251/

[10] Satyam, K. (2020). Countplot using seaborn in Python. Retrieved from https://www.geeksforgeeks.org/countplot-using-seaborn-in-python/

[11] Marcelino, P. (2019, August 13). Comprehensive data exploration with Python. Retrieved from https://www.kaggle.com/pmarcelino/comprehensive-data-exploration-with-python

[12] Yadav, D. (2019, December 09). Categorical encoding using Label-Encoding and One-Hot-Encoder. Retrieved October 17, 2020, from https://www.kaggle.com/sulianova/cardiovascular-disease-dataset

[13] User Guide¶. (n.d.). Retrieved December 15, 2020, from https://scikit learn.org/stable/user_guide.html

[14] Gharn1989. (2018, April 10). Plotly Dash Tutorial - Creating your first app (Video 01). Retrieved from https://www.youtube.com/watch?v=yPSbJSblrvw