

Capstone Project

Machine Learning Engineer Nanodegree

Heart_Disease_Prediction

M.N.Shivapriya
February 26th, 2019

I. Definition

Project Overview

Nowadays, diseases are increasing day by day due to lifestyle, hereditary. Especially, heart disease has become more common these days, i.e. life of people is at risk. Among these are poor diet, lack of regular exercise, tobacco smoking, alcohol or drug abuse, and high stress

Heart disease is the leading cause of death for both men and women all over the world, accounting for about one million deaths each year. Nearly half of all cardiac events (such as a heart attack) happen in people with no personal history of heart disease. About 85 percent of people who die as a result of heart disease are 65 or older.

Many factors can contribute to the onset of coronary artery disease, which usually develops over time. Some of these can't be changed, such as a family history or increasing age. However, there is a lot you can do to reduce your risk of developing coronary artery disease. The lifestyle choices you make can go a long way to preventing heart disease.

Heart diseases are the leading cause of death globally, resulted in 17.9 million deaths (32.1%) in 2015, up from 12.3 million (25.8%) in 1990. It is estimated that 90% of disease is preventable. There are many risk factors for heart diseases that we will take a closer look at. The prediction of heart disease requires a huge size of data which is too complex and massive to process and analyze by conventional techniques.

The main objective of this study is to find out and build the suitable machine learning technique that is computationally efficient as well as accurate for the prediction of heart disease occurrence, based on a combination of features (risk factors) describing the disease. Different machine learning classification techniques will be implemented and compared upon standard performance metric such as accuracy.

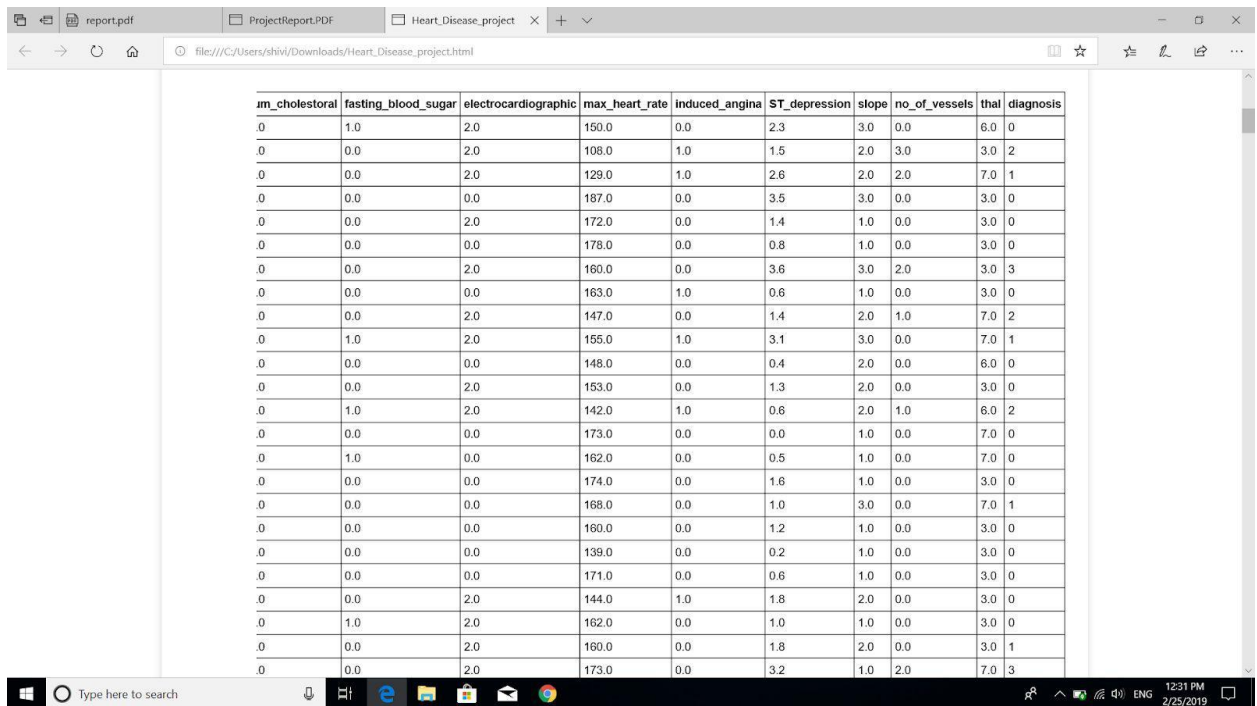
Reference Link:

https://www.researchgate.net/publication/328031918_Machine_Learning_Classification_Techniques_for_Heart_Disease_Prediction_A_Review#pf7

Problem Statement:

The main aim of my project is to predict the rate of heart disease. For this I selected the data set compiled from a wide range of sources and made publicly available by the United States Department of Agriculture Economic Research Service (USDA ERS). So, My goal is to predict the rate of heart diseases in U.S. Here I am using the classification models to find the accuracy of each model and select the best model which will have high accuracy to predict the rate of heart diseases.

1. Loading the data from dataset with `read_csv` command.
 2. Information about the data has been made with `info()`.
 3. Check whether there are any missing values in the dataset, if any replaced with 0.
 4. Numerical and Categorical Attributes has been differentiated.
 5. Visualization of Numerical Attributes has been made.
 6. Algorithms has been implemented.
 7. Best model among the models has been selected.
- Here the input parameters are the training data and the output will either 0 or 1 i.e. having heart heart disease or not.



am_cholesterol	fasting_blood_sugar	electrocardiographic	max_heart_rate	induced_angina	ST_depression	slope	no_of_vessels	thal	diagnosis
0	1.0	2.0	150.0	0.0	2.3	3.0	0.0	6.0	0
0	0.0	2.0	108.0	1.0	1.5	2.0	3.0	3.0	2
0	0.0	2.0	129.0	1.0	2.6	2.0	2.0	7.0	1
0	0.0	0.0	187.0	0.0	3.5	3.0	0.0	3.0	0
0	0.0	2.0	172.0	0.0	1.4	1.0	0.0	3.0	0
0	0.0	0.0	178.0	0.0	0.8	1.0	0.0	3.0	0
0	0.0	2.0	160.0	0.0	3.6	3.0	2.0	3.0	3
0	0.0	0.0	163.0	1.0	0.6	1.0	0.0	3.0	0
0	0.0	2.0	147.0	0.0	1.4	2.0	1.0	7.0	2
0	1.0	2.0	155.0	1.0	3.1	3.0	0.0	7.0	1
0	0.0	0.0	148.0	0.0	0.4	2.0	0.0	6.0	0
0	0.0	2.0	153.0	0.0	1.3	2.0	0.0	3.0	0
0	1.0	2.0	142.0	1.0	0.6	2.0	1.0	6.0	2
0	0.0	0.0	173.0	0.0	0.0	1.0	0.0	7.0	0
0	1.0	0.0	162.0	0.0	0.5	1.0	0.0	7.0	0
0	0.0	0.0	174.0	0.0	1.6	1.0	0.0	3.0	0
0	0.0	0.0	168.0	0.0	1.0	3.0	0.0	7.0	1
0	0.0	0.0	160.0	0.0	1.2	1.0	0.0	3.0	0
0	0.0	0.0	139.0	0.0	0.2	1.0	0.0	3.0	0
0	0.0	0.0	171.0	0.0	0.6	1.0	0.0	3.0	0
0	0.0	2.0	144.0	1.0	1.8	2.0	0.0	3.0	0
0	1.0	2.0	162.0	0.0	1.0	1.0	0.0	3.0	0
0	0.0	2.0	160.0	0.0	1.8	2.0	0.0	3.0	1
0	0.0	2.0	173.0	0.0	3.2	1.0	2.0	7.0	3

Metrics:

Accuracy:

It is the number of correct predictions made by the model over all kinds of predictions made

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{FP} + \text{FN} + \text{TN})$$

Accuracy is a good measure when the target variable classes in the data are nearly balanced

The following posts will provide some methods to evaluate the performance of a machine learning problem

<https://towardsdatascience.com/choosing-the-right-metric-for-evaluating-machine-learning-models-part-2-86d5649a5428>

<https://towardsdatascience.com/choosing-the-right-metric-for-machine-learning-models-part-1-a99d7d7414e4>

II. Analysis:

Data Exploration:

In this data set, I have used 14 attributes and around 503 trained and test data to evaluate accuracy.

Experiments with the Cleveland database have concentrated on endeavors to distinguish disease presence (values 1, 2, 3, 4) from absence (value 0). There are several missing attribute values, distinguished with symbol '?'. The header row is missing in this dataset, so the column names have to be inserted manually as shown in the below table

Number of records: 503 Number of variables: 14										
	age	sex	chest_pain	blood_pressure	serum_cholesterol	fasting_blood_sugar	electrocardiographic	max_heart_rate	induced_angina	ST_depression
0	63.0	1.0	1.0	145.0	233.0	1.0	2.0	150.0	0.0	2.3
1	67.0	1.0	4.0	160.0	286.0	0.0	2.0	108.0	1.0	1.5
2	67.0	1.0	4.0	120.0	229.0	0.0	2.0	129.0	1.0	2.6
3	37.0	1.0	3.0	130.0	250.0	0.0	0.0	187.0	0.0	3.5
4	41.0	0.0	2.0	130.0	204.0	0.0	2.0	172.0	0.0	1.4
5	56.0	1.0	2.0	120.0	236.0	0.0	0.0	178.0	0.0	0.8
6	62.0	0.0	4.0	140.0	268.0	0.0	2.0	160.0	0.0	3.6
7	57.0	0.0	4.0	120.0	354.0	0.0	0.0	163.0	1.0	0.6
8	63.0	1.0	4.0	130.0	254.0	0.0	2.0	147.0	0.0	1.4
9	53.0	1.0	4.0	140.0	203.0	1.0	2.0	155.0	1.0	3.1
10	57.0	1.0	4.0	140.0	192.0	0.0	0.0	148.0	0.0	0.4
11	56.0	0.0	2.0	140.0	294.0	0.0	2.0	153.0	0.0	1.3
12	56.0	1.0	3.0	130.0	256.0	1.0	2.0	142.0	1.0	0.6
13	44.0	1.0	2.0	120.0	263.0	0.0	0.0	173.0	0.0	0.0
14	52.0	1.0	3.0	172.0	199.0	1.0	0.0	162.0	0.0	0.5
15	57.0	1.0	3.0	150.0	168.0	0.0	0.0	174.0	0.0	1.6

Features information:

age- age in years

Sex - sex(1 = male; 0 = female)

chest_pain - chest pain type (1 = typical angina; 2 = atypical angina; 3 = non-anginal pain; 4 = asymptomatic)

blood_pressure - resting blood pressure (in mm Hg on admission to the hospital)
serum_cholesterol - serum cholesterol in mg/dl

fasting_blood_sugar - fasting blood sugar > 120 mg/dl (1 = true; 0 = false)

electrocardiographic - resting electrocardiographic results (0 = normal; 1 = having ST-T; 2 = hypertrophy)

max_heart_rate - maximum heart rate achieved

induced_angina - exercise induced angina (1 = yes; 0 = no)

ST_depression - ST depression induced by exercise relative to rest

slope - the slope of the peak exercise ST segment (1 = upsloping; 2 = flat; 3 = downsloping)
no_of_vessels - number of major vessels (0-3) colored by flourosopy

thal - 3 = normal; 6 = fixed defect; 7 = reversable defect

diagnosis - the predicted attribute - diagnosis of heart disease (angiographic disease status)
(Value 0 = < 50% diameter narrowing; Value 1 = > 50% diameter narrowing)

Types of features:

1. **Categorical features** (Has two or more categories and each value in that feature can be categorised by them): sex, chest_pain
2. **Ordinal features** (Variable having relative ordering or sorting between the values):

fasting_blood_sugar, electrocardiographic, induced_angina, slope, no_of_vessels, thal, diagnosis

3. **Continuous features** (Variable taking values between any two points or between the minimum or maximum values in the feature column): age, blood_pressure, serum_cholesterol, max_heart_rate, ST_depression

Description of Dataset:

```
# Display a description of the total dataset
display(df.describe())
```

	age	sex	chest_pain	blood_pressure	serum_cholesterol	fasting_blood_sugar	electrocardiographic	max_heart_rate	induced_angina
count	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000	503.000000
mean	54.258449	0.675944	3.176938	131.262425	248.552684	0.147117	1.035785	149.648111	0.333996
std	9.029434	0.468487	0.948108	18.046937	51.780612	0.354575	0.995364	23.208166	0.472108
min	29.000000	0.000000	1.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000
25%	47.500000	0.000000	3.000000	120.000000	212.000000	0.000000	0.000000	133.000000	0.000000
50%	55.000000	1.000000	3.000000	130.000000	245.000000	0.000000	2.000000	153.000000	0.000000
75%	61.000000	1.000000	4.000000	140.000000	277.000000	0.000000	2.000000	166.000000	1.000000
max	77.000000	1.000000	4.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000

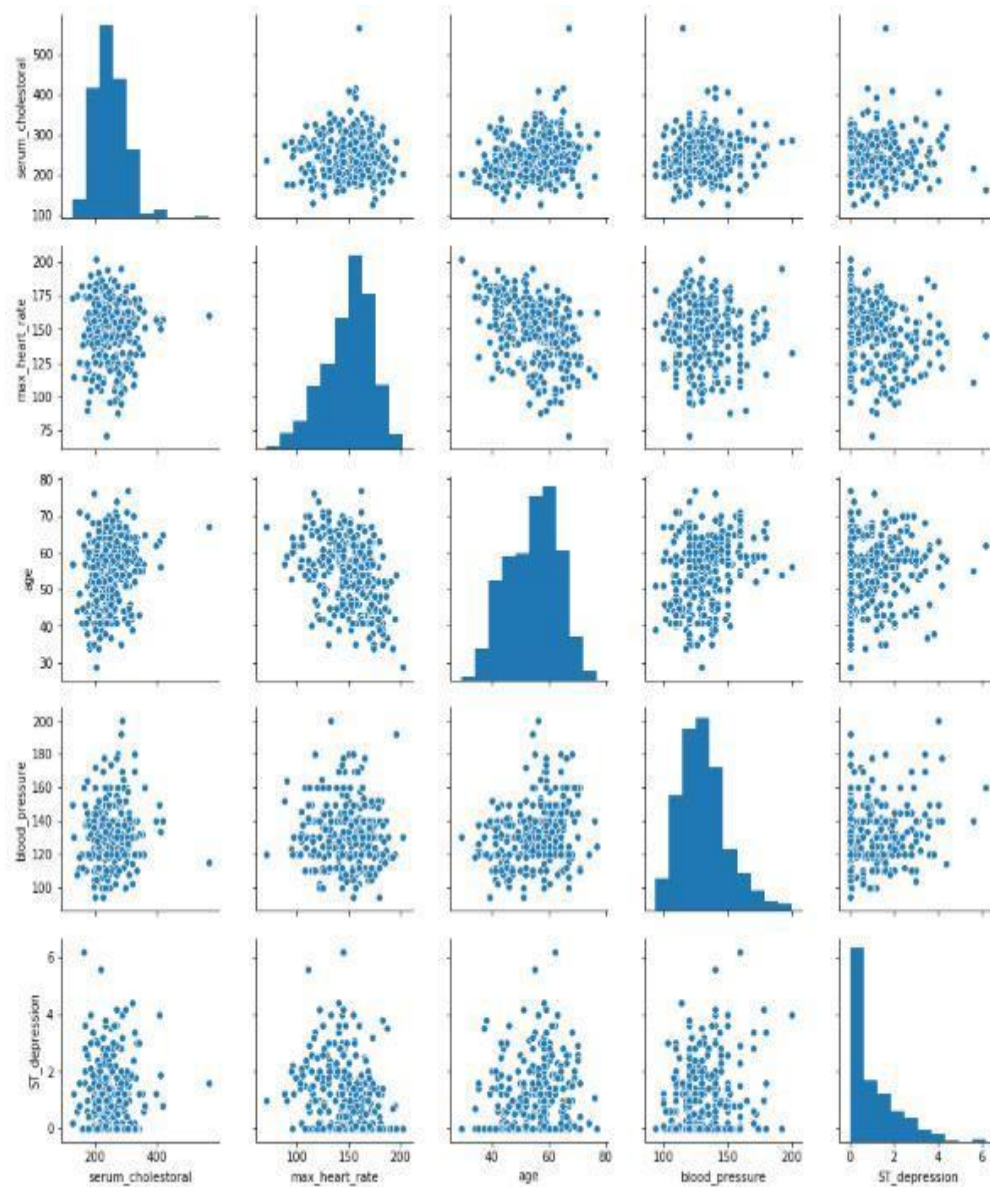
Information of Data set:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 503 entries, 0 to 502
Data columns (total 14 columns):
age                503 non-null float64
sex                503 non-null float64
chest_pain         503 non-null float64
blood_pressure     503 non-null float64
serum_cholesterol  503 non-null float64
fasting_blood_sugar 503 non-null float64
electrocardiographic 503 non-null float64
max_heart_rate     503 non-null float64
induced_angina     503 non-null float64
ST_depression      503 non-null float64
slope              503 non-null float64
no_of_vessels      497 non-null float64
thal               500 non-null float64
diagnosis           503 non-null int64
dtypes: float64(13), int64(1)
memory usage: 55.1 KB
```

There are only 9 missing values in this dataset and all variables are recognized as numeric. From dataset description, we know, however, that most of features are categorical and it's necessary to distinguish them.

Data Visualization



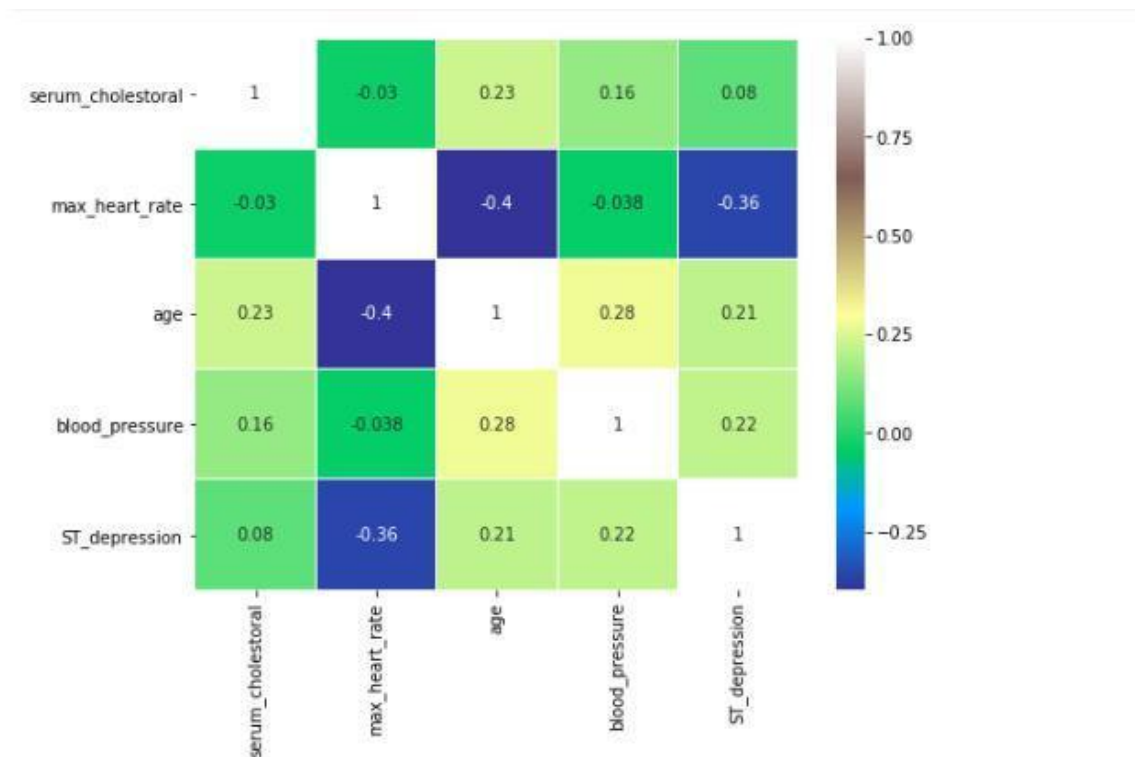
Let us visualize the absolute correlation coefficient of target variable with all the other variables. Higher absolute correlation coefficient means the variable can provide more information about how the target variable moves as shown in below figure. From the above plots, I infer that none of the displayed value pairs is having an explicitly high correlation, so there is no necessity to ditch any feature at this stage. I also notice a negative correlation between 'age' and 'max_heart_rate' and positive correlation between 'age' and 'blood_pressure', which is intuitive.

A correlation matrix will make us sure whether the above is correct.

Heat Map:

A heat map is a two-dimensional representation of information with the help of colors. Heat maps can help the user visualize simple or complex information. Heat maps are used in many areas such as

defense, marketing and understanding consumer behavior. More elaborate heat maps allow the user to understand complex data.



Apart from two aforementioned relations, there is one more important dependency: 'max_heart_rate' and 'ST_depression'. The conclusion comes up that both features, 'age' and 'max_heart_rate', will play an important role in predicting heart disease.

Observations:

Men are much more prone to get a heart disease than women.

The higher number of vessels detected through fluoroscopy, the higher risk of disease.

While soft chest pain may be a bad symptom of approaching problems with heart (especially in case of men), strong pain is a serious warning!

Risk of getting heart disease might be even 3x higher for someone who experienced exercise-induced angina. The

flat slope (value=2) and downslope (value=3) of the peak exercise indicates a high risk of getting disease

Modelling and Predicting with Machine Learning:

The main goal of the entire project is to predict heart disease occurrence with the highest accuracy. In order to achieve this, we will test several classification algorithms. This section includes all results obtained from the study and introduces the best performer according to accuracy metric. I have chosen several algorithms typical for solving supervised learning problems throughout classification methods.

```
def train_model(X_train, y_train, X_test, y_test, classifier, **kwargs):  
    """  
    Fit the chosen model and print out the score.  
    """  
  
    # instantiate model  
    model = classifier(**kwargs)  
  
    # train model  
    model.fit(X_train, y_train)  
  
    # check accuracy and print out the results  
    fit_accuracy = model.score(X_train, y_train)  
    test_accuracy = model.score(X_test, y_test)  
    predictions_train = model.predict(X_train)  
    predictions_test = model.predict(X_test)  
  
    print "Train accuracy: {:.2f}%".format(fit_accuracy*100)  
    print "Test accuracy: {:.2f}%".format(test_accuracy*100)  
    print("Train F-Score: {}".format(fbeta_score(y_train, predictions_train, 0.5)))  
    print("Test F-Score: {}".format(fbeta_score(y_test, predictions_test, 0.5)))  
  
    return model
```

First of all, let's equip ourselves with a handy tool that benefits from the cohesion of SciKit Learn library and formulate a general function for training our models. The reason for displaying accuracy on both, train and test sets, is to allow us to evaluate whether the model overfits or under fits the data (so-called bias/variance tradeoff)

Algorithms and Techniques:

1. K-Nearest Neighbours
2. Decision Trees
3. Logistic Regression
4. Gaussian Naïve Bayes
5. Support Vector Machines
6. Random Forests

1. K-Nearest Neighbours (KNN)

K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). KNN has been used in statistical estimation and pattern recognition already in the beginning of 1970's as a non-parametric technique.

K-Nearest Neighbours algorithm is a non-parametric method used for classification and regression. The principle behind nearest neighbour methods is to find a predefined number of training samples closest in distance to the new point and predict the label from these.

Advantages:

The K-Nearest Neighbor (KNN) Classifier is a very simple classifier that works well on basic recognition problems. Effective if training set is large.

Disadvantages:

The main disadvantage of the KNN algorithm is that it is a *lazy learner*, i.e. it does not learn anything

from the training data and simply uses the training data itself for classification.

To predict the label of a new instance the KNN algorithm will find the K closest neighbors to the new instance from the training data, the predicted class label will then be set as the most common label among

the K closest neighboring points.

The algorithm must compute the distance and sort all the training data at each prediction, which can be

slow if there are a large number of training examples.

Another disadvantage of this approach is that the algorithm does not learn anything from the training data, which can result in the algorithm not generalizing well and also not being robust to noisy data.

Decision Tree

A **decision tree** is a decision support tool that uses a tree-like model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm that only contains conditional control statements.

Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in machine learning.

Advantages:

Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.

Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.

Help determine worst, best and expected values for different scenarios.

Can be combined with other decision techniques.

Disadvantages:

They are unstable, meaning that a small change in the data can lead to a large change in the structure of the optimal decision tree.

They are often relatively inaccurate. Many other predictors perform better with similar data. This can be remedied by replacing a single decision tree with a random forest of decision trees, but a random forest is not as easy to interpret as a single decision tree.

For data including categorical variables with different number of levels, information gain in decision trees is biased in favor of those attributes with more levels.^[7]

Calculations can get very complex, particularly if many values are uncertain and/or if many outcomes are linked.

3. Logistic Regression

Logistic regression is a basic technique in statistical analysis that attempts to predict a data value based on prior observations. A logistic regression algorithm looks at the relationship between a dependent variable and one or more independent variables.

Advantages: Logistic regression is designed for this purpose (classification), and is most useful for understanding the influence of several independent variables on a single outcome variable.

Disadvantages: Works only when the predicted variable is binary, assumes all predictors are independent of each other, and assumes data is free of missing values.

4. Naïve Bayes

In machine learning, naive Bayes classifiers are a family of simple probabilistic classifiers based on applying Bayes' theorem with strong (naive) independence assumptions between the features.

Advantages:

Very simple, easy to implement and fast. If the NB conditional independence assumption holds, then it will converge quicker than discriminative models like logistic regression. Even if the NB assumption doesn't hold, it works great in practice. Need less training data. Highly scalable. It scales linearly with the number of predictors and data points. Can be used for both binary and multiclass classification problems. Can make probabilistic predictions. Handles continuous and discrete data. Not sensitive to irrelevant features.

Disadvantages:

Naive Bayes is known to be a bad estimator. The Naive Bayes classifier makes a very strong assumption on the shape of your data distribution, i.e. any two features are independent given the output class. Due to this, the result can be potentially very bad - hence, a "naive" classifier.

This is not as terrible as people generally think, because the NB classifier can be optimal even if the assumption is violated, and its results can be good even in the case of sub-optimality.

5.Support Vector Machine

Support Vector Machines are perhaps one of the most popular machine learning algorithms. They are the go-to method for a high-performing algorithm with a little tuning. At first, let's try it on default settings.

Advantages: Effective in high dimensional spaces and uses a subset of training points in the decision function so it is also memory efficient.

Disadvantages: The algorithm does not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation.

6.Random Forest

Random forest classifier is a meta-estimator that fits a number of decision trees on various sub-samples of datasets and uses average to improve the predictive accuracy of the model and controls over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement.

Advantages: Reduction in over-fitting and random forest classifier is more accurate than decision trees in most cases.

Disadvantages: Slow real time prediction, difficult to implement, and complex algorithm.

Benchmark Model:

Here we compare the final model with the remaining models to see if it got better or same or worse. The accuracy is compared among the models and the optimal model is selected. I choose Naive Bayes model as the benchmark model. Now we will try and achieve better accuracy than this model by using the above mentioned classification models.

III. Methodology:

Pre-Processing:

In this step we will pre-process the data. Data pre-processing is considered to be the first and foremost step that is to be done before starting any process. We will read the data by using `read_csv`. Then we will know the shape of the data. And by using the `info()` we will know the information of the attributes. Then we will check whether there are any null values by using `isnull()`. After that we will divide the whole data into training and testing data. We will assign 70% of the data to the training data and the remaining 30% of the data into testing data. We will do this by using `train_test_split` from `sklearn.model_selection`.

Implementation:

The implementation of the project is involved in two steps:

- 1.The classifier training stage
- 2.The model development stage

Classifier training stage:

The classifier was trained on the preprocessed data steps followed here is:

- Load the training and testing data into memory and preprocess. Here the data is preprocessed by finding the unique values and missing values. Missing values are replaced by 'nans'
- Using displot() function for defining the visualization of distribution of target labels and feature labels.
- Finding the correlation of the every feature with the target label to know which feature has the strongest with the target label.
- Now splitting the data into training and testig for further implementation.

Model Development Stage:

Out of the chosen algorithms we will start with KNN classification model. We will take a classifier and fit the training data. After that we will predict that by using predict (X_train). Now we will predict the accuracy of the testing data by using accuracy score (y_test, pred). By doing so for, the KNN will give us the accuracy of 0.834.We will continue the same procedure on Naïve Bayes, SVM, Decision tree ,Logistic Regression and Random Forest . By following the same procedure above that is fitting, predicting and finding the accuracy score we will get the accuracy score as bellow.

	Accuracy
KNN	0.8344
Decision Tree	0.8344
Logistic Regression	0.827
Naïve Bayes	0.8211
SVM	0.8476
Random Forests	0.9139

From the above reports Random Forest seems to be performing well.

Complications faced here:

- Cleaning and splitting the data for training and testing.
- Selecting the tuning parameters for the model to get the best accuracy score.

Refinement:

I found out random forest as the best classifier out of the chosen classifiers. Now we will perform tuning of random forest classifier in order to achieve the better accuracy. For doing refinement we will just tune the parameter. Here we will assign n_estimators=[100,] and max_depth=100. Now will find out the new accuracy. The new accuracy will be 0.9139.

max_depth : integer or None, optional (default=None)

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

n_estimators : integer, optional (default=10)

The number of trees in the forest. *Changed in version 0.20:* The default value of n_estimators will change from 10 in version 0.20 to 100 in version 0.22.

random_state : int, RandomState instance or None, optional (default=None)

If int, random_state is the seed used by the random number generator; If RandomState instance, random_state is the random number generator; If None, the random number generator is the RandomState instance used by np.random.

Before tuning the accuracy score of random forest is 89%

After tuning the accuracy score of the random forest is 91%

So, the best model for this classification problem is Random Forest

Complications:

- Chest pain (angina): When your coronary arteries narrow, your heart may not receive enough blood when demand is greatest — particularly during physical activity.
- Heart attack: If a cholesterol plaque ruptures and a blood clot forms, complete blockage of your heart artery may trigger a heart attack.
- Heart failure: If some areas of your heart are chronically deprived of oxygen and nutrients because of reduced blood flow, or if your heart has been damaged by a heart attack, your heart may become too weak to pump enough blood to meet your body's needs.
- Abnormal heart rhythm (arrhythmia): Inadequate blood supply to the heart or damage to heart tissue can interfere with your heart's electrical impulses, causing abnormal heart rhythms.

These four factors have higher issues to get heart disease and increase the complication level.

These attributes play a major role for diagnosis of heart disease.

IV. Result

Model evaluation and validation

The final model we have chosen is tuned random forest which gave us more accuracy that is 0.9139. In order to achieve this accuracy we assigned n_estimators=[110]. Here we can say that the solution is reasonable because we are getting much more less accuracy while using other models. The final model

that is tuned random forest has been tested with various inputs to evaluate whether the model generalises well. This model is also robust enough for the given problem. We can say this by testing it over different random states. From this we can say that Small changes in the training data will not affect the results greatly. So the results found from this model can be trusted.

The result can be trusted because here i am using train_test_split for splitting the data into training and testing.And the data with same features can work correctly because i am using the hyper parameter as random state,max_depth and n_estimators so the data with same changes will not affect the model. So,my model works correctly for this model

Justification:

My final model's solution is better than the benchmark model.

Tuned	Random Forest	Benchmark Model
Accuracy	0.9139	0.8344

From the above we can conclude that the results for the final model are stronger than the benchmark model. Hence we can say that tuned random forest provides the significant to solve the problem of predicting heart diseases.

V. Conclusion:

The goal of the project was to compare different machine learning algorithms and predict if a certain person, given various personal characteristics and symptoms, will get heart disease or not. Here are the final results.

```
# initialize an empty list
accuracy1 = []
accuracy2 = []
# List of algorithms names
classifiers = ['KNN', 'Decision Trees', 'Logistic Regression', 'Naive Bayes', 'SVM', 'Random Forests']

# List of algorithms with parameters
models = [KNeighborsClassifier(n_neighbors=5), DecisionTreeClassifier(max_depth=6, random_state=2606), LogisticRegression(),
          GaussianNB(), SVC(C=0.05, kernel='linear'), RandomForestClassifier(n_estimators=110, random_state=2606)]

# Loop through algorithms and append the score into the list
for i in models:
    model = i
    model.fit(X_train, y_train)
    score = model.score(X_test, y_test)
    score1=model.score(X_train,y_train)
    accuracy1.append(score1)
    accuracy2.append(score)
```

```
In [117]: # create a dataframe from accuracy results
summary = pd.DataFrame({'Train_Accuracy':accuracy1, 'Test_Accuracy':accuracy2}, index=classifiers)
summary
```

Out[117]:

	Test_Accuracy	Train_Accuracy
KNN	0.834437	0.906250
Decision Trees	0.834437	0.934659
Logistic Regression	0.827815	0.872159
Naive Bayes	0.821192	0.877841
SVM	0.847682	0.857955
Random Forests	0.913907	1.000000

It does not come as a surprise that the more complex algorithms like SVM and Random Forests generated better results compared to the basic ones. It is worth to emphasize that in most cases hyperparameter tuning is essential to achieve robust results out of these techniques. By producing decent results, simpler methods proved to be useful as well.

Machine learning has absolutely bright future in medical field. Just imagine a place where heart disease experts are not available. With just basic information about a certain patient's medical history, we may quite accurately predict whether a disease will occur or not.

Reflection:

1. I have learnt how to visualize and understand the data.
2. I have learnt that the data cleaning place a very vital role in data analytics.
3. Removing the data features which are not necessary in evaluating model is very important.
4. I got to know how to use the best technique for the data using appropriate ways
5. I got to know how to tune the parameters in order to achieve the best score.

On a whole I learnt how to graph a dataset and applying cleaning techniques on it and to fit the best techniques to get best score.

Improvement:

Artificial Neural Network (ANN) performed well in most models for predicting heart disease as well as Decision Tree (DT). Finally, the field of using machine learning for diagnosing heart disease is an important field, and it can help both healthcare professionals and patients. It is still a growing field, and despite the massive availability of patient data in hospitals or clinics, not much of it is published.

Since the quality of the dataset is an essential factor in the prediction accuracy, more hospitals should be encouraged to publish high-quality datasets (while protecting the privacy of patients) so that researchers can have a good source to help them develop their models and obtain good results.