Cardiovascular Disease Prediction using Deep-Learning

A project submitted

In partial fulfillment of the requirements for the degree of

Bachelor of Technology in Computer Science & Engineering

SUBMITTED BY

KANISHKA SAHU

(11701173)

(3CE 6)

Department of Computer Engineering

PUNJABI UNIVERSITY, PATIALA

UNIVERSITY COLLEGE OF ENGINEERING (UCOE)

Content

Acknowledgement

The training opportunity I had with **iSMRITI at LPU Jalandhar** was a great chance for learning and enhancing my technical skills. I consider myself as lucky to have been provided an opportunity by the **Punjabi University, Patiala** to do training in a leading technology of today. I am also grateful to the **University's Computer Science Department** for guiding me and helping to find training as per my need. I express my deepest gratitude and thanks to **Mr. Vipul Arora (IIT Kanpur)**, Course Coordinator at Excellence Technology who in spite of being extraordinarily busy with his schedule took time out to solve my doubts, guide and keep me on the correct path and allowing to me carry out my project at their esteemed organization during the training. I perceive this opportunity as a big milestone in my career development. I will strive to use gained skills and knowledge in the best possible way, and I will continue to work on their improvement, in order to attain desired career objectives.
Sincerely,

Kanishka sahu

Student Declaration

I, KANISHKA SAHU, hereby declare that the presented report of the training titled "Teach Tech services" done in "AI ML and DL" is uniquely prepared by me after the completion of six weeks' summer training done at Teach Tech Services LPU ,Jalandhar . I also confirm that this report is only prepared for my academic requirement, not for any other purpose.

KANISHKA SAHU

Roll No. – 11701173

Department of Computer Science and Engineering

About Organization

iSMRITI is India's leading company in Artificial Intelligence, Internet of Things and its application for positive impact. Being the Brainchild of faculty and alumni from various IITs, iSMRITI does Consultancy, Product Development and Educational Workshops. Our company is based at IIT Kanpur with a group of passionate and experienced professionals carrying a rich intellectual background from world-class engineering institutions. We focus on providing technical expertise in various cutting-edge fields involving the Internet of Things (IoT), Artificial Intelligence (AI) and Robotics.

iSMRITI is on a technological mission to push the periphery of AI and IoT towards every section of our society. Providing skilful education to the students and professionals, consultations to the firms/enterprises, and innovative technological solutions to the industry is our way of making this world a better place.

Mission of the organization

The real value of intelligence and technology is when it simplifies things around us and also preserves rich human values. With the boost of 'Make in India', our philosophy is to develop a culture of makers, by educating the needful and helping the proficient. We desire not just a few, but everyone, to learn, imagine, dream and create a new future.

Specializations offered by organization

- IoT
- Machine learning
- Data Science
- Bigdata
- Full stack web development

- Android development
- Core java
- Artificial intelligence

Specialization Opted

In today's world Machine Learning and ai seems to take a nice curve in providing solution to many problems which were very difficult to approach earlier. So, learning this technology could actually prove to be a good idea in upcoming future.

The main purpose for opting in course of machine learning was my pre-training knowledge stack, which was oriented toward python and some basic machine learning and exploring this field more did proved as a good idea at the end.

Certificate

INTRODUCTION

This is a web-based simple to use cardiovascular diseases stage prediction application that can predict the health status of your heart with the help of Deep learning algorithm. This model is trained over 70000+ original data and statics of different humans and their health factor that we are using to train our model like blood pressure, diabetes or not, smoke or not, age, height etc.

I created this web-based app using flask that is a python-based web app development framework. It gives a simple cool interface so that anyone can simple create account and register and then he/she can fill the details of his/her and then can check the heart health status with the corresponding suggestion.

I'm using keras to train the model. Keras is an open-source neural-network library written in Python. It is capable of running on top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano, or PlaidML. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Also, I'm suing data representation to analysis the dataset and other features of the data set. In the following report I'll show, you the graphs and other stats of dataset.

PROBLEM

According to the American heart association cardiovascular disease is listed as the underlying cause of death, accounted for 840,678 deaths in the US in 2016, approximately 1 of every 3 deaths.

Following are the major point mentioned by the American health association –

• Between 2013 and 2016, 121.5 million American adults had some form of cardiovascular disease.

Between 2014 and 2015, direct and indirect costs of total cardiovascular diseases and stroke

were $351.2 billion ($213.8 billion in direct costs and $137.4 billion in lost productivity/mortality).

• In 2013-2016, 57.1% of non-Hispanic (NH) black females and 60.1% of NH black males had some

form of cardiovascular disease.

• In 2016, Coronary Heart Disease was the leading cause (43.2%) of deaths attributable to

cardiovascular disease in the US, followed by stroke (16.9%), High Blood Pressure (9.8%), Heart

Failure (9.3%), diseases of the arteries (3.0%), and other cardiovascular diseases (17.7%).

• Cardiovascular disease is the leading global cause of death, accounting for more than 17.6

million deaths per year in 2016, a number that is expected to grow to more than 23.6 million by

2030, according to a 2014 study.

SOLUTION THAT CAN HELP

From the above-mentioned points, we can understand the condition of how big this problem is but if we can get a simple warning before this disease attacks on our heart, we can safely prevent the condition. So, this project is all about how we can get the prediction stages of our hearth's health so. In this project we are taking the original data of 70000+ peoples. They are belonging from the different backgrounds and have different daily routine and habits. For training of this dataset, we are using Keras library using the TensorFlow backend.

A small warning gives us a chance to understand the health status of our heart and we can take care of our heart. Surely this model is not 100% accurate but it will give us small warning so we can save a life.

In the todays world artificial intelligence is doing really great job in the medical science field as well. So the well trained prediction models can play a great role.

The role of AI & Machine Learning in Medical Science

Machine learning works effectively in the presence of huge data. Medical science is yielding large amount of data daily from research and development (R&D), physicians and clinics, patients, caregivers etc. These data can be used as synchronizing the information and using it to improve healthcare infrastructure and treatments. This has potential to help so many people, to save lives and money. As per a research, big data and machine learning in pharma and medicine could generate a value of up to $100B annually, based on better decision-making, optimized innovation, improved efficiency of research/clinical trials, and new tool creation for physicians, consumers, insurers and regulators.

DATA SET FOR TARINING OF MODEL

There are 3 types of input features:

- *Objective*: factual information;
- *Examination*: results of medical examination;
- *Subjective*: information given by the patient.

Features:

1. Age | Objective Feature | age | int (days)
2. Height | Objective Feature | height | int (cm) |
3. Weight | Objective Feature | weight | float (kg) |
4. Gender | Objective Feature | gender | categorical code |
5. Systolic blood pressure | Examination Feature | ap_hi | int |
6. Diastolic blood pressure | Examination Feature | ap_lo | int |
7. Cholesterol | Examination Feature | cholesterol | 1: normal, 2: above normal, 3: well above normal |
8. Glucose | Examination Feature | gluc | 1: normal, 2: above normal, 3: well above normal |
9. Smoking | Subjective Feature | smoke | binary |
10. Alcohol intake | Subjective Feature | alco | binary |
11. Physical activity | Subjective Feature | active | binary |
12. Presence or absence of cardiovascular disease | Target Variable | cardio | binary |

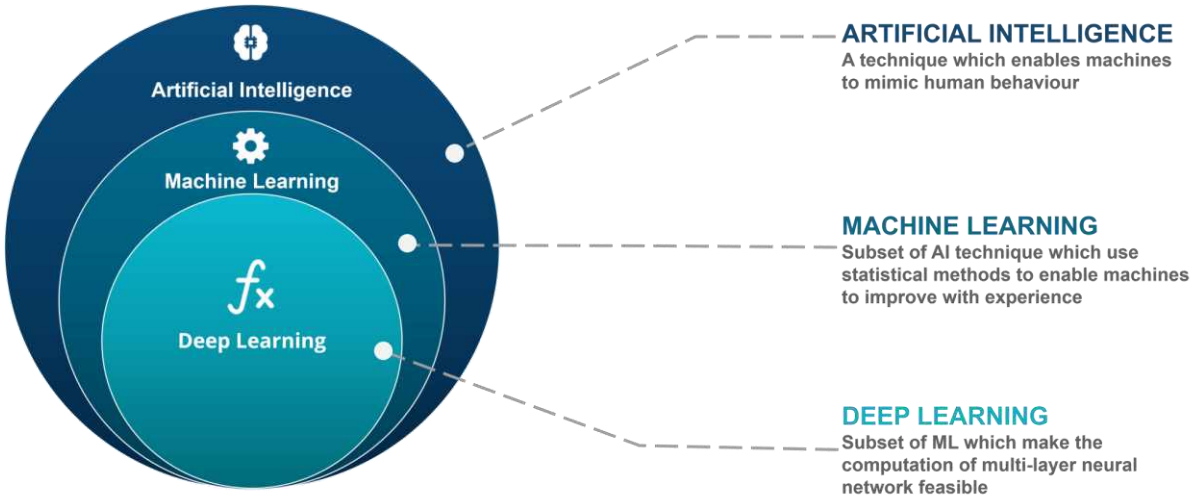
	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
id												
0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	17474	1	156	56.0	100	60	1	1	0	0	0	0

Machine learning

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers learn automatically without human intervention or assistance and adjust actions accordingly.

Some machine learning methods Machine learning algorithms are often categorized as supervised or unsupervised.



Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training.

The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output, but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.

Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

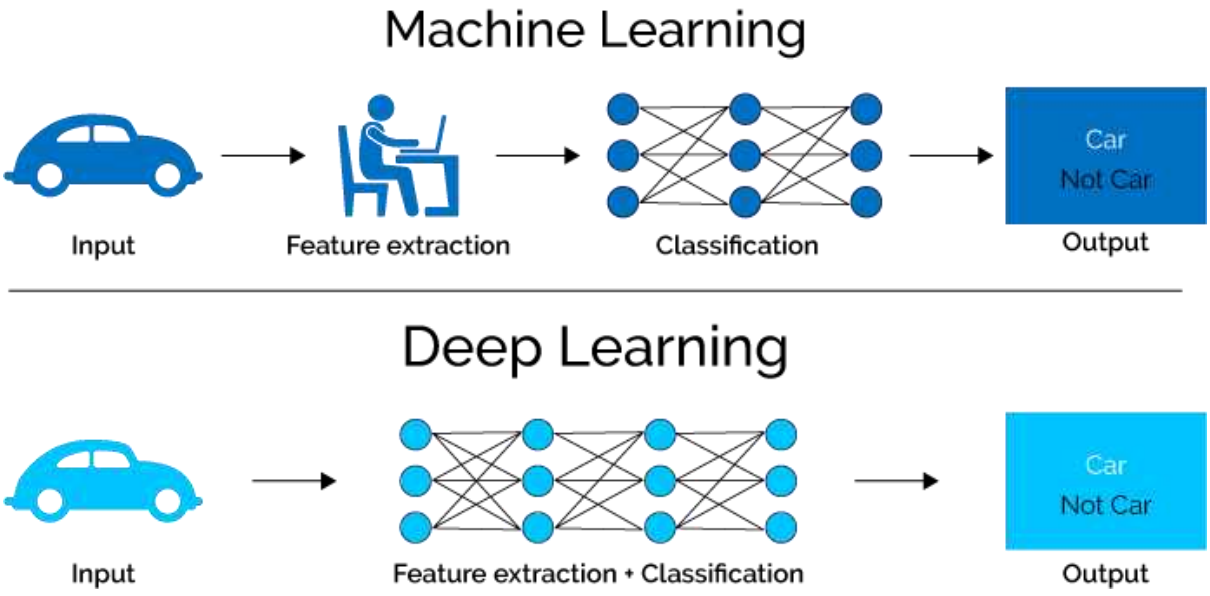
Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

Deep learning

Deep learning is an artificial intelligence function that imitates the workings of the human brain in processing data and creating patterns for use in decision making. Deep learning is a subset of machine learning in artificial intelligence (AI) that has networks capable of learning unsupervised from data that is unstructured or unlabeled. Also known as deep neural learning or deep neural network.

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

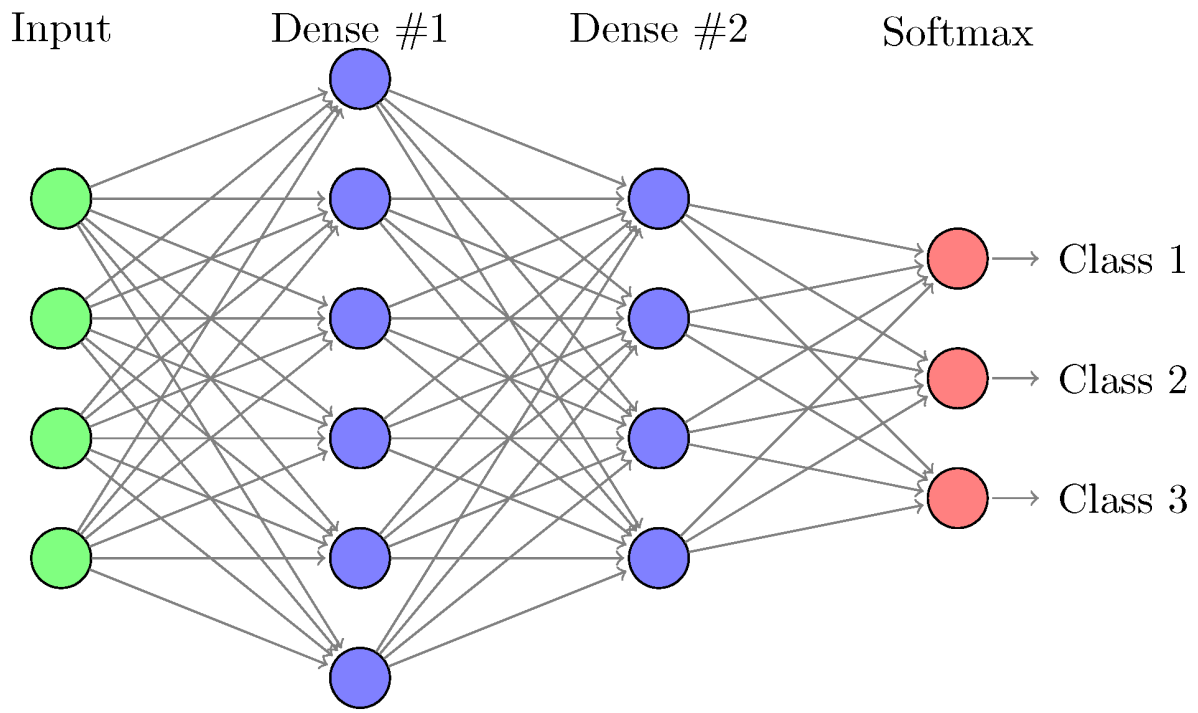
However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.



Neural Network

A neural network is a network or circuit of neurons, or in a modern sense, an artificial neural network, composed of artificial neurons or nodes.[1] Thus a neural network is either a biological neural network, made up of real biological neurons, or an artificial neural network, for solving artificial intelligence (AI) problems. The connections of the biological neuron are modeled as weights. A positive weight reflects an excitatory connection, while negative values mean inhibitory connections. All inputs are modified by a weight and summed. This activity is referred as a linear combination. Finally, an activation function controls the amplitude of the output. For example, an acceptable range of output is usually between 0 and 1, or it could be −1 and 1.

These artificial networks may be used for predictive modeling, adaptive control and applications where they can be trained via a dataset. Self-learning resulting from experience can occur within networks, which can derive conclusions from a complex and seemingly unrelated set of information.



Technology used in this project

This project includes so many different languages and frameworks listed as follows: -

- HyperText Markup Language (HTML)
- CSS
- Python 3.6
- Flask
- Flask-wtforms
- Numpy
- Pandas
- Keras
- Seaborn
- Matplotlib

Hypertext Markup Language (HTML)

HTML is the standard markup language for creating Web pages.

- HTML stands for Hyper Text Markup Language
- HTML describes the structure of Web pages using markup
- HTML elements are the building blocks of HTML pages
- HTML elements are represented by tags
- HTML tags label pieces of content such as "heading", "paragraph", "table", and so on
- Browsers do not display the HTML tags, but use them to render the content of the page

A Simple HTML Document:

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Python 3.6

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Its high-level built in data structures, combined with dynamic typing and dynamic binding, make it very attractive for Rapid Application Development, as well as for use as a scripting or glue language to connect existing components together.

Python's simple, easy to learn syntax emphasizes readability and therefore reduces the cost of program maintenance. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed.

Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy: a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception.

When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. On the other hand, often the quickest way to debug a program is to add a few print statements to the source: the fast edit-test-debug cycle makes this simple approach very effective

Keras high-level neural Network API

Keras is a high-level neural networks API, written in Python and capable of running on top of TensorFlow, CNTK, or Theano. It was developed with a focus on enabling fast experimentation. Being able to go from idea to result with the least possible delay is key to doing good research.

Use Keras if you need a deep learning library that:

- Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility).
- Supports both convolutional networks and recurrent networks, as well as combinations of the two.
- Runs seamlessly on CPU and GPU.

Read the documentation at Keras.io.

Keras is compatible with: Python 2.7-3.6.

How we can import keras -

```
from keras.models import Sequential

model = Sequential()
```

```
from keras.layers import Dense

model.add(Dense(units=64, activation='relu', input_dim=100))
model.add(Dense(units=10, activation='softmax'))
```

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

a powerful N-dimensional array object

sophisticated (broadcasting) functions

tools for integrating C/C++ and Fortran code

useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

Flask (Web Framework)

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications. It began as a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks.

Flask offers suggestions, but doesn't enforce any dependencies or project layout. It is up to the developer to choose the tools and libraries they want to use. There are many extensions provided by the community that make adding new functionality easy.

Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack. It was introduced by John Hunter in the year 2002.

One of the greatest benefits of visualization is that it allows us visual access to huge amounts of data in easily digestible visuals. Matplotlib consists of several plots like line, bar, scatter, histogram etc.

Seaborn

Seaborn is a library for making statistical graphics in Python. It is built on top of matplotlib and closely integrated with pandas data structures.

Here is some of the functionality that seaborn offers:

- A dataset-oriented API for examining relationships between multiple variables
- Specialized support for using categorical variables to show observations or aggregate statistics
- Options for visualizing univariate or bivariate distributions and for comparing them between subsets of data
- Automatic estimation and plotting of linear regression models for different kinds dependent variables
- Convenient views onto the overall structure of complex datasets
- High-level abstractions for structuring multi-plot grids that let you easily build complex visualizations

Seaborn aims to make visualization a central part of exploring and understanding data. Its dataset-oriented plotting functions operate on data frames and arrays containing whole datasets and internally perform the necessary semantic mapping and statistical aggregation to produce informative plots.

How it works

The dataset consists of 70 000 records of patients data in 12 features, such as age, gender, systolic blood pressure, diastolic blood pressure, and etc. The target class "cardio" equals to 1, when patient has cardiovascular desease, and it's 0, if patient is healthy.

The task is to predict the presence or absence of cardiovascular disease (CVD) using the patient examination results.

Data description

There are 3 types of input features:

- Objective: factual information;
- Examination: results of medical examination;

Subjective: information given by the patient.

Feature	Variable Type	Variable	Value Type
Age	Objective Feature	age	int (days)
Height	Objective Feature	height	int (cm)
Weight	Objective Feature	weight	float (kg)
Gender	Objective Feature	gender	categorical code
Systolic blood pressure	Examination Feature	ap_hi	int
Diastolic blood pressure	Examination Feature	ap_lo	int
Cholesterol	Examination Feature	cholesterol	1: normal, 2: above normal, 3: well above normal
Glucose	Examination Feature	gluc	1: normal, 2: above normal, 3: well above normal
Smoking	Subjective Feature	smoke	binary
Alcohol intake	Subjective Feature	alco	binary
Physical activity	Subjective Feature	active	binary
Presence or absence of cardiovascular disease	Target Variable	cardio	binary

Initial analysis

Let's look at the dataset and given variables. Here we are using panda's data frame code to show the top head of the data set. Df is our data frame.

```
df.head()
```

:

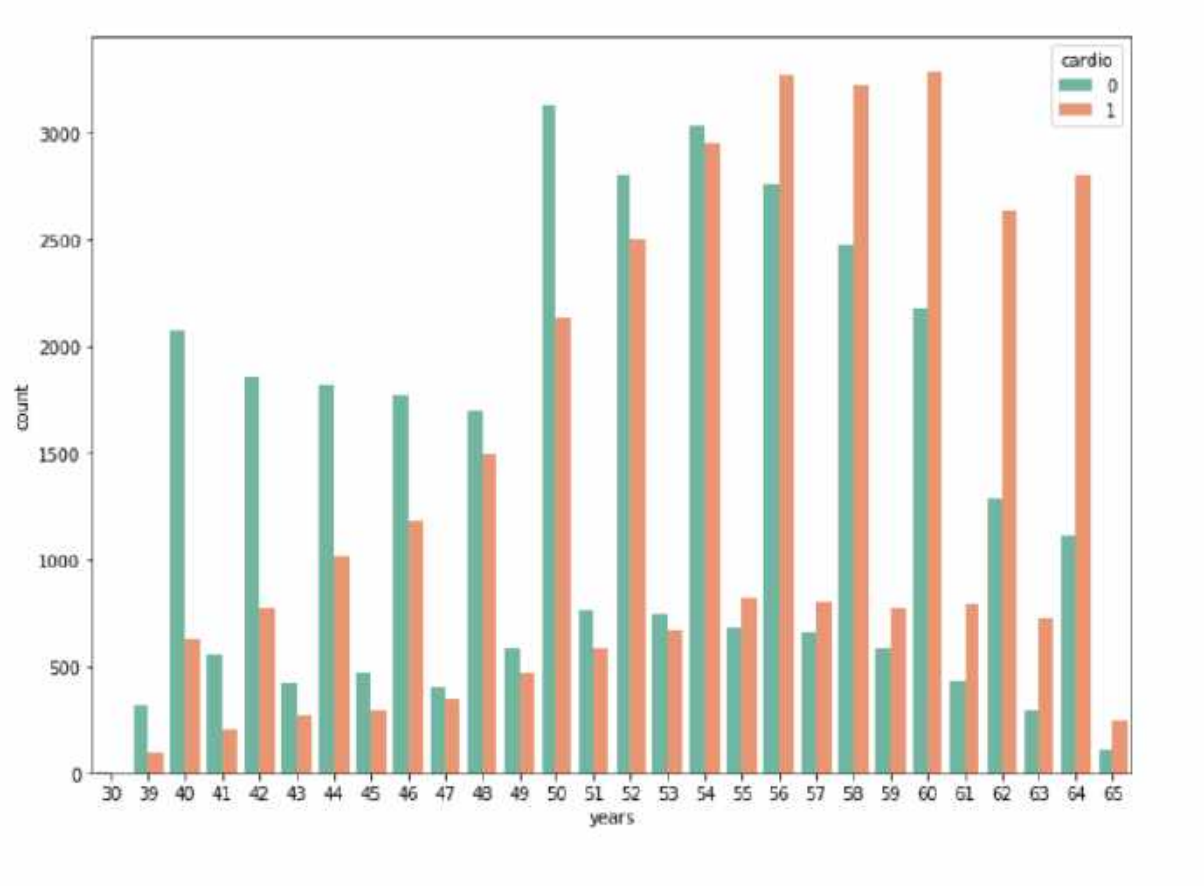
	id	age	gender	height	weight	ap_hi	ap_lo	cholesterol	gluc	smoke	alco	active	cardio
0	0	18393	2	168	62.0	110	80	1	1	0	0	1	0
1	1	20228	1	156	85.0	140	90	3	1	0	0	1	1
2	2	18857	1	165	64.0	130	70	3	1	0	0	0	1
3	3	17623	2	169	82.0	150	100	1	1	0	0	1	1
4	4	17474	1	156	56.0	100	60	1	1	0	0	0	0

Univariate analysis

To understand all our variables, at first, we should look at their datatypes. We can do it with info() function:

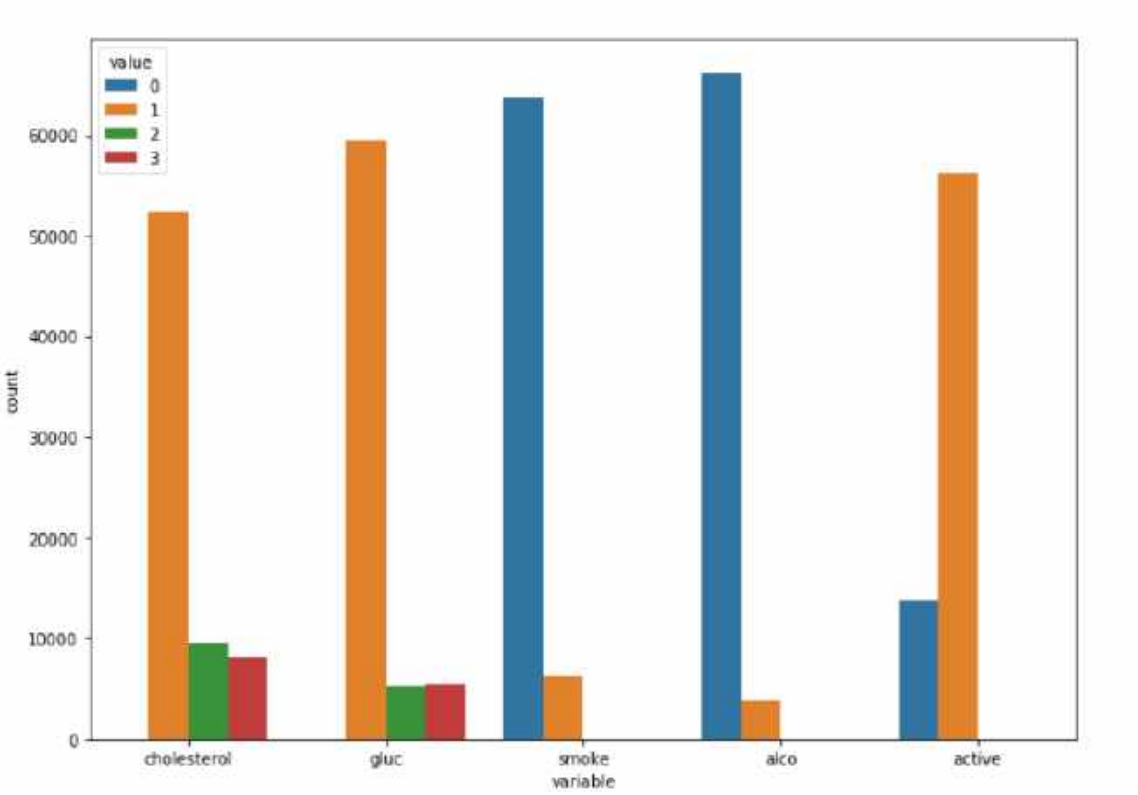
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 70000 entries, 0 to 69999
Data columns (total 13 columns):
id              70000 non-null int64
age             70000 non-null int64
gender          70000 non-null int64
height          70000 non-null int64
weight          70000 non-null float64
ap_hi           70000 non-null int64
ap_lo           70000 non-null int64
cholesterol     70000 non-null int64
gluc            70000 non-null int64
smoke           70000 non-null int64
alco            70000 non-null int64
active          70000 non-null int64
cardio          70000 non-null int64
dtypes: float64(1), int64(12)
memory usage: 6.9 MB
```

Age is measured in days; height is in centimetres. Let's look at the numerical variables and how are they spread among target class. For example, at what age does the number of people with CVD exceed the number of people without CVD?

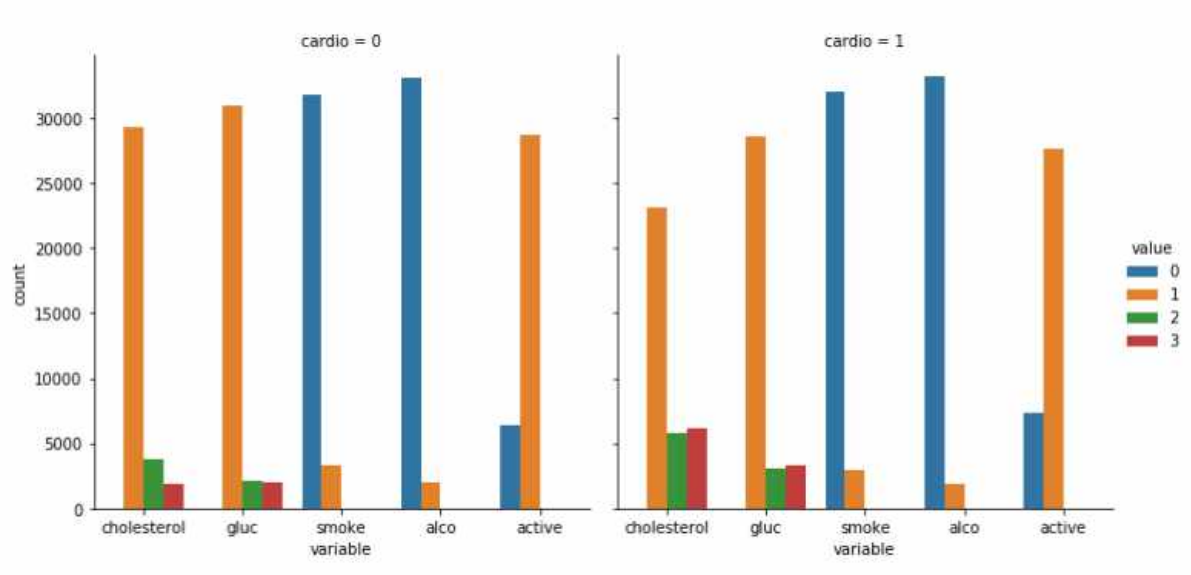


It can be observed that people over 55 of age are more exposed to CVD. From the table above, we can see that there are outliers in ap_hi, ap_lo, weight and height. We will deal with them later

Let's look at categorical variables in the dataset and their distribution:



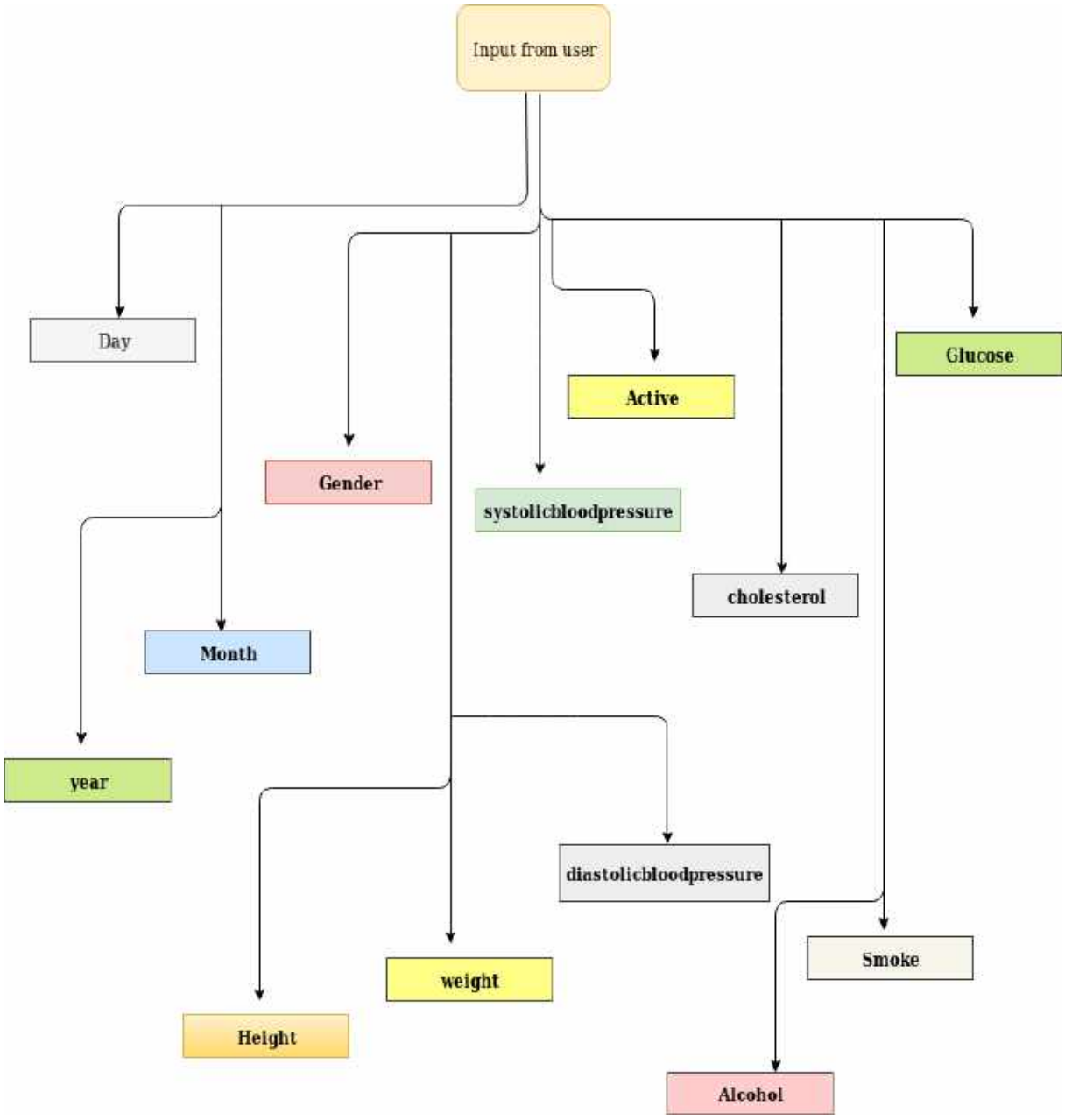
It may be useful to split categorical variables by target class:



How our model is predicting

Batch Size and Epochs

In the final kernel, this code is not gonna be available 'cause it takes too much time to run this parts of code. The summary will be written below the function.



```
def create_model(): model = Sequential() model.add(Dense(10, input_dim=11,
activation='tanh')) model.add(Dropout(0.1)) model.add(Dense(5,
activation='relu')) model.add(Dropout(0.1)) model.add(Dense(3,
activation='relu')) model.add(Dropout(0.1)) model.add(Dense(1,
activation='sigmoid')) model.compile(loss = 'binary_crossentropy',
optimizer='sgd', metrics=['accuracy']) return model
```

create model

```
model = KerasClassifier(build_fn=create_model, verbose=0)
```

define the parameters to search in grid search

```
batch_size = [10, 50, 100] epochs = [10, 50, 100] param_grid =
dict(batch_size=batch_size, epochs=epochs) grid =
GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5)
grid_result = grid.fit(x_training, y_training)
```

summarize results

```
print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams))
means = grid_result.cvresults['mean_test_score'] stds =
grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params']
for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" %
(mean, stdev, param))
```

Results:

```
Best: 0.640893 using {'batch_size': 50, 'epochs': 50}
0.638214 (0.003997) with: {'batch_size': 10, 'epochs': 10}
0.640446 (0.006447) with: {'batch_size': 10, 'epochs': 50}
0.613500 (0.060568) with: {'batch_size': 10, 'epochs': 100}
0.617089 (0.012489) with: {'batch_size': 50, 'epochs': 10}
```

0.640893 (0.004402) with: {'batch_size': 50, 'epochs': 50}

0.640661 (0.003730) with: {'batch_size': 50, 'epochs': 100}

0.613732 (0.017299) with: {'batch_size': 100, 'epochs': 10}

0.635268 (0.005980) with: {'batch_size': 100, 'epochs': 50}

0.639482 (0.007552) with: {'batch_size': 100, 'epochs': 100}

Optimization Algorithm

In the final kernel, this code is not gonna be available 'cause it takes too much time to run this parts of code. The summary will be written below the function.

function to create model

```
def create_model(optimizer='adam'): model = Sequential()
model.add(Dense(10, input_dim=11, activation='tanh'))
model.add(Dropout(0.1)) model.add(Dense(5, activation='relu'))
model.add(Dropout(0.1)) model.add(Dense(3, activation='relu'))
model.add(Dropout(0.1)) model.add(Dense(1, activation='sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer=optimizer,
metrics=['accuracy']) return model
```

create model

```
model = KerasClassifier(build_fn=create_model, epochs=50, batch_size=50,
verbose=0)
```

define the parameters to search in grid search

```
optimizer = ['SGD', 'RMSprop', 'Adagrad', 'Adadelta', 'Adam', 'Adamax',
'Nadam'] param_grid = dict(optimizer=optimizer) grid =
```

```
GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5)
grid_result = grid.fit(x_training, y_training)

summarize results


print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams))
means = grid_result.cvresults['mean_test_score'] stds =
grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params']
for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" %
(mean, stdev, param))
```

Results:

```
Best: 0.708232 using {'optimizer': 'Nadam'}
0.639339 (0.003833) with: {'optimizer': 'SGD'}
0.685446 (0.014199) with: {'optimizer': 'RMSprop'}
0.644750 (0.004682) with: {'optimizer': 'Adagrad'}
0.664732 (0.011830) with: {'optimizer': 'Adadelta'}
0.676821 (0.017248) with: {'optimizer': 'Adam'}
0.653625 (0.003463) with: {'optimizer': 'Adamax'}
0.708232 (0.010960) with: {'optimizer': 'Nadam'}
```

Optimization of SGD algorithm (Learnig Rate and Momentum)

function to create model

```
def create_model(learn_rate=0.01, momentum=0): model = Sequential()
model.add(Dense(10, input_dim=11, activation='tanh'))
model.add(Dropout(0.1)) model.add(Dense(5, activation='relu'))
```

```
model.add(Dropout(0.1)) model.add(Dense(3, activation='relu'))
model.add(Dropout(0.1)) model.add(Dense(1, activation='sigmoid')) optimizer
= SGD(lr=learn_rate, momentum=momentum) model.compile(loss =
'binary_crossentropy', optimizer=optimizer, metrics=['accuracy']) return model

create model


model = KerasClassifier(build_fn=create_model, epochs=50, batch_size=50,
verbose=0)

define the parameters to search in grid search


learn_rate = [0.01, 0.1, 0.2] momentum = [0.2, 0.6, 0.9] param_grid =
dict(learn_rate=learn_rate, momentum=momentum) grid =
GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5)
grid_result = grid.fit(x_training, y_training)

summarize results


print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams))
means = grid_result.cvresults['mean_test_score'] stds =
grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params']
for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" %
(mean, stdev, param))
```

Results:

Best: 0.642571 using {'learn_rate': 0.01, 'momentum': 0.2}

0.642571 (0.004326) with: {'learn_rate': 0.01, 'momentum': 0.2}

0.642375 (0.003565) with: {'learn_rate': 0.01, 'momentum': 0.6}

0.641786 (0.005498) with: {'learn_rate': 0.01, 'momentum': 0.9}

0.586232 (0.069370) with: {'learn_rate': 0.1, 'momentum': 0.2}

0.642089 (0.004087) with: {'learn_rate': 0.1, 'momentum': 0.6}

0.638589 (0.006434) with: {'learn_rate': 0.1, 'momentum': 0.9}

0.498554 (0.005800) with: {'learn_rate': 0.2, 'momentum': 0.2}

0.587482 (0.071161) with: {'learn_rate': 0.2, 'momentum': 0.6}

0.594286 (0.058599) with: {'learn_rate': 0.2, 'momentum': 0.9}

Activation functions

def create_model(activation='relu'): model = Sequential()
model.add(Dense(10, input_dim=11, activation=activation))
model.add(Dropout(0.1)) model.add(Dense(5, activation=activation))
model.add(Dropout(0.1)) model.add(Dense(3, activation=activation))
model.add(Dropout(0.1)) model.add(Dense(1, activation='sigmoid'))
model.compile(loss = 'binary_crossentropy', optimizer="Nadam",
metrics=['accuracy']) return model

create model

model = KerasClassifier (build_fn=create_model, epochs=50, batch_size=50,
verbose=0)

define the parameters to search in grid search activation = ['softmax',
'softplus', 'softsign', 'relu', 'tanh', 'sigmoid', 'hard_sigmoid', 'linear'] param_grid
= dict(activation=activation) grid = GridSearchCV(estimator=model,
param_grid=param_grid, n_jobs=-1, cv=5) grid_result = grid.fit(x_training,
y_training)

summarize results

```
print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams))
means = grid_result.cvresults['mean_test_score'] stds =
grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params']
for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" %
(mean, stdev, param))
```

Results:

Best: 0.717250 using {'activation': 'softsign'}

0.649786 (0.005472) with: {'activation': 'softmax'}

0.649464 (0.005333) with: {'activation': 'softplus'}

0.717250 (0.006675) with: {'activation': 'softsign'}

0.701964 (0.021487) with: {'activation': 'relu'}

0.717179 (0.002289) with: {'activation': 'tanh'}

0.648446 (0.006001) with: {'activation': 'sigmoid'}

0.651750 (0.004286) with: {'activation': 'hard_sigmoid'}

0.709232 (0.015933) with: {'activation': 'linear'}

Dropout Rate and Weight Constraint

from keras.layers import Dropout from keras.constraints import maxnorm

function to create model

def create_model(dropout_rate=0.0, weight_constraint=0): model = Sequential() model.add(Dense(10, input_dim=11, activation='softsign', kernel_constraint=maxnorm(weight_constraint))) model.add(Dropout(dropout_rate)) model.add(Dense(5, activation='softsign')) model.add(Dropout(dropout_rate)) model.add(Dense(3, activation='softsign')) model.add(Dropout(dropout_rate)) model.add(Dense(1, activation='sigmoid')) model.compile(loss = 'binary_crossentropy', optimizer='Nadam', metrics=['accuracy']) return model

create model

model = KerasClassifier (build_fn=create_model, epochs=50, batch_size=50, verbose=0)

define the parameters to search in grid search

weight_constraint = [0, 1, 2] dropout_rate = [0.0, 0.1, 0.2] param_grid = dict(dropout_rate=dropout_rate, weight_constraint=weight_constraint) grid = GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5) grid_result = grid.fit(x_training, y_training)

summarize results

print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams)) means = grid_result.cvresults['mean_test_score'] stds = grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params'] for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" % (mean, stdev, param))

Results:

Best: 0.701179 using {'dropout_rate': 0.0, 'weight_constraint': 2}

0.500839 (0.005918) with: {'dropout_rate': 0.0, 'weight_constraint': 0}

0.651839 (0.005446) with: {'dropout_rate': 0.0, 'weight_constraint': 1}

0.701179 (0.010650) with: {'dropout_rate': 0.0, 'weight_constraint': 2}

0.500446 (0.005961) with: {'dropout_rate': 0.1, 'weight_constraint': 0}

0.648250 (0.007944) with: {'dropout_rate': 0.1, 'weight_constraint': 1}

0.684536 (0.022927) with: {'dropout_rate': 0.1, 'weight_constraint': 2}

0.500304 (0.005970) with: {'dropout_rate': 0.2, 'weight_constraint': 0}

0.649732 (0.002889) with: {'dropout_rate': 0.2, 'weight_constraint': 1}

0.666893 (0.011692) with: {'dropout_rate': 0.2, 'weight_constraint': 2}

Number of neurons in the first layer

```
def create_model(neurons=1): model = Sequential()
model.add(Dense(neurons, input_dim=11, activation='softsign',
kernel_constraint=maxnorm(2))) model.add(Dropout(0)) model.add(Dense(5,
activation='softsign')) model.add(Dropout(0)) model.add(Dense(3,
activation='softsign')) model.add(Dropout(0)) model.add(Dense(1,
activation='sigmoid')) model.compile(loss = 'binary_crossentropy',
optimizer='Nadam', metrics=['accuracy']) return model
```

create model

```
model = KerasClassifier(build_fn=create_model, epochs=50, batch_size=50,
verbose=0)
```

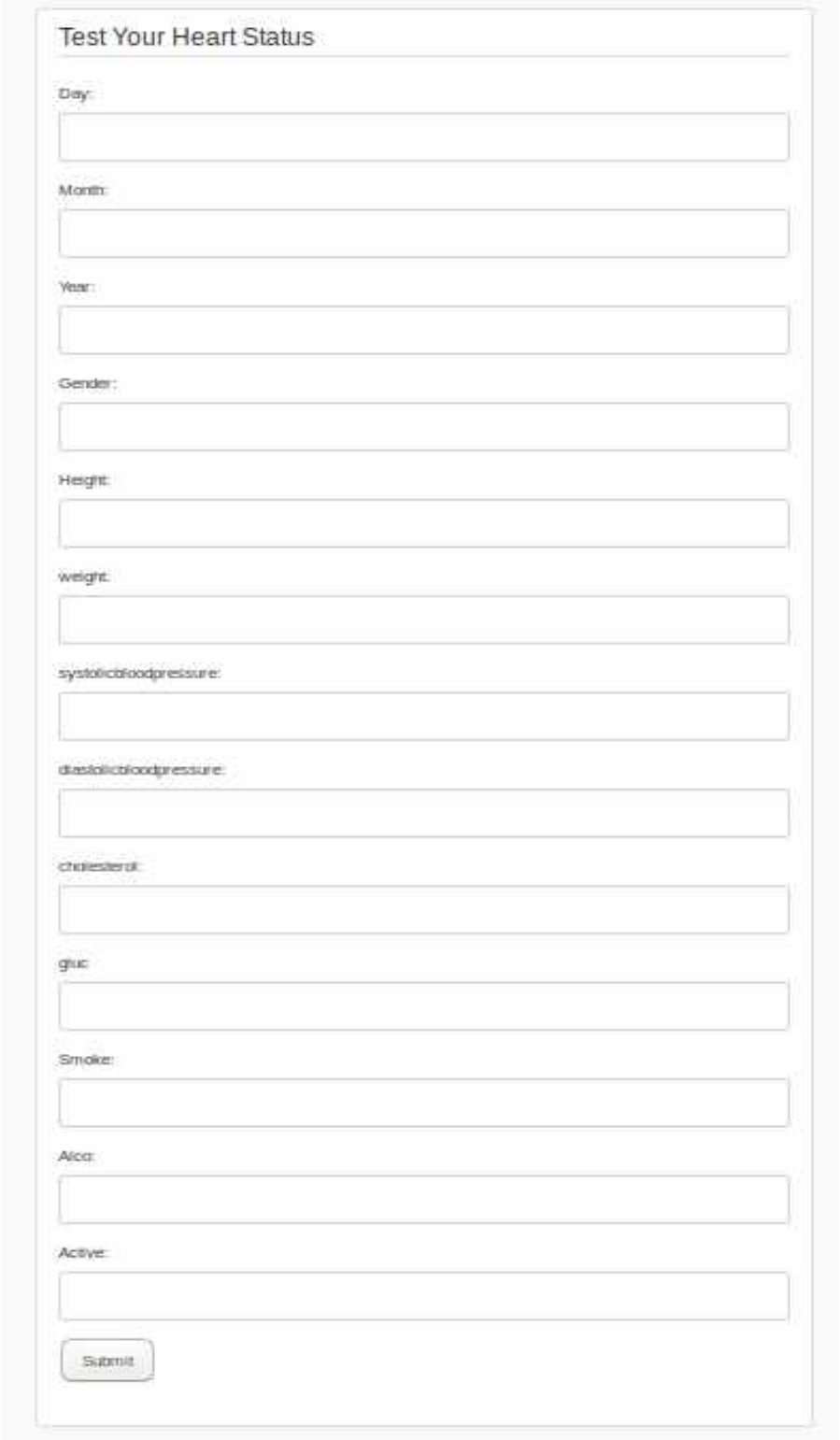
define the parameters to search in grid search

```
neurons = [1, 5, 10, 15, 20, 25, 30] param_grid = dict(neurons=neurons) grid =
GridSearchCV(estimator=model, param_grid=param_grid, n_jobs=-1, cv=5)
grid_result = grid.fit(x_training, y_training)

summarize results


print("Best: %f using %s" % (grid_result.bestscore, grid_result.bestparams))
means = grid_result.cvresults['mean_test_score'] stds =
grid_result.cvresults['std_test_score'] params = grid_result.cvresults['params']
for mean, stdev, param in zip(means, stds, params): print("%f (%f) with: %r" %
(mean, stdev, param))
```

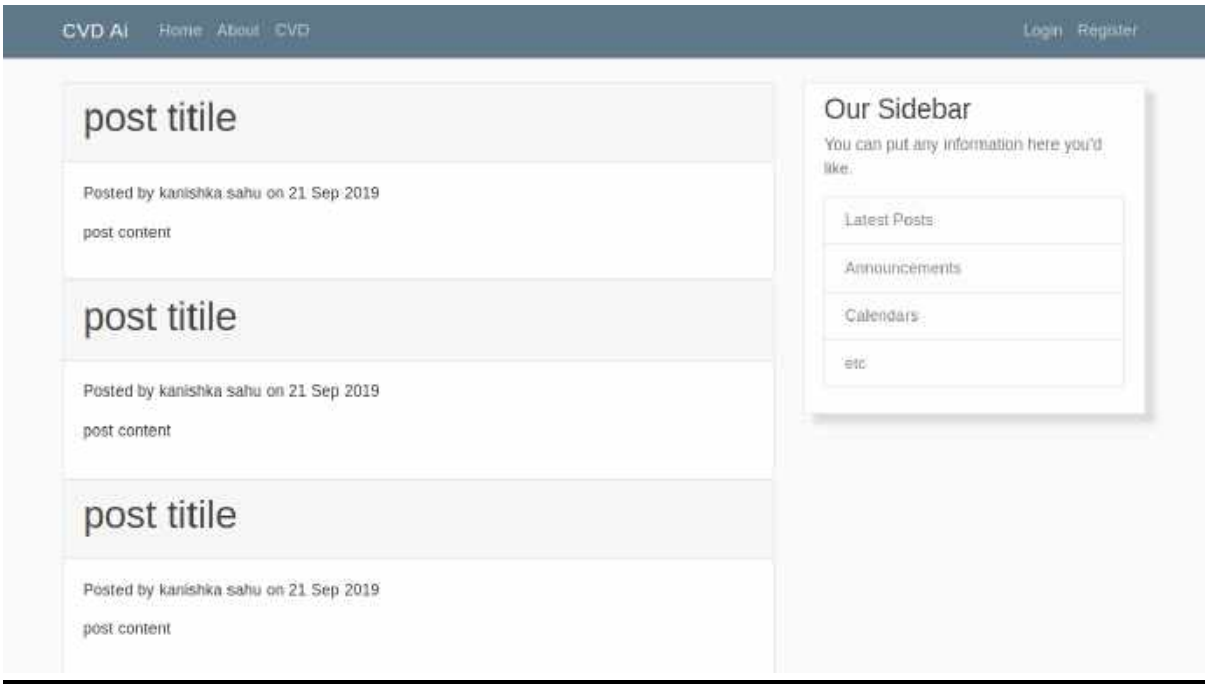
Results:

```
Best: 0.703107 using {'neurons': 25}
0.683607 (0.003922) with: {'neurons': 1}
0.700411 (0.003911) with: {'neurons': 5}
0.694625 (0.009716) with: {'neurons': 10}
0.699536 (0.009807) with: {'neurons': 15}
0.698893 (0.013422) with: {'neurons': 20}
0.703107 (0.014787) with: {'neurons': 25}
0.688821 (0.026863) with: {'neurons': 30}
```

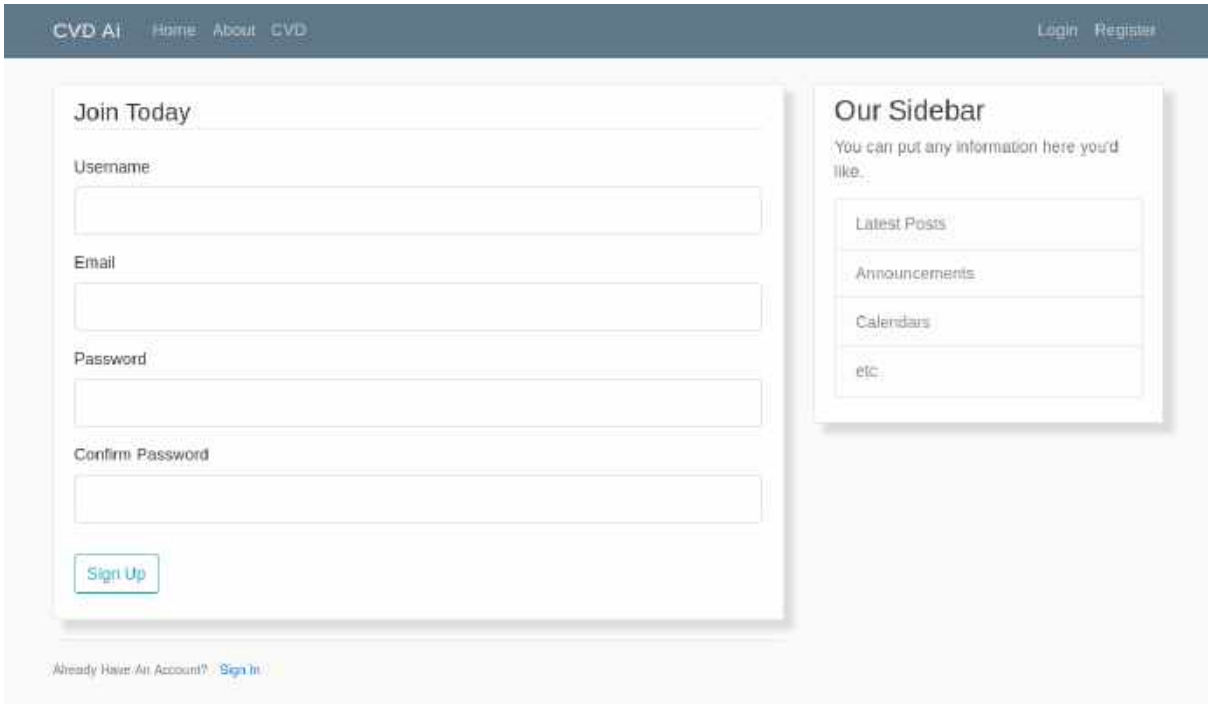
User input Form



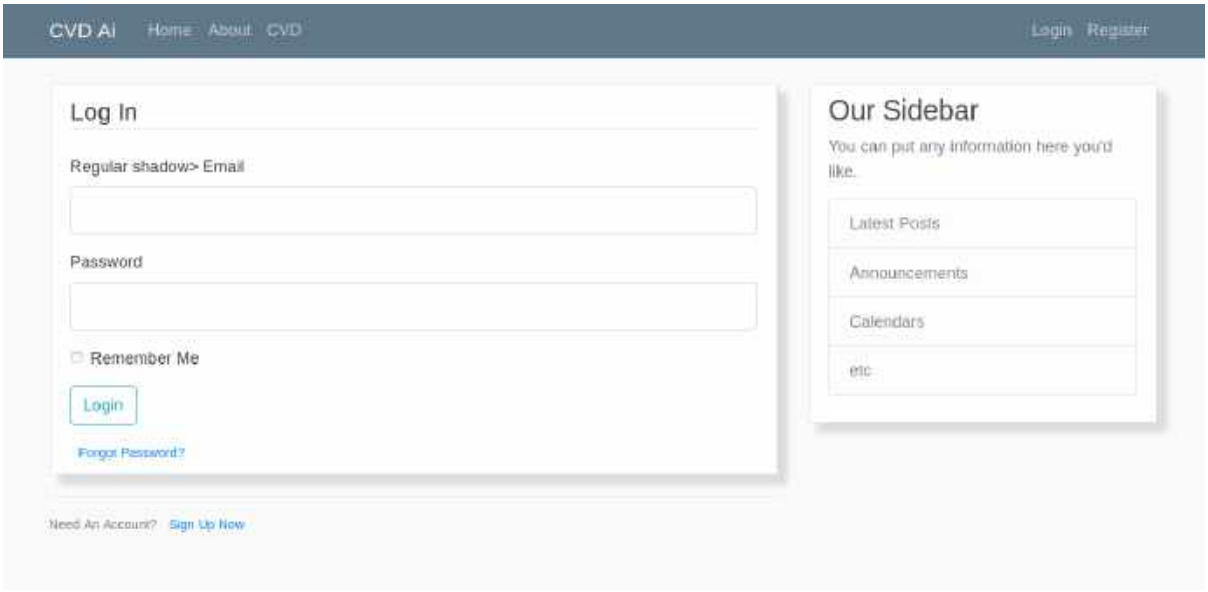
Homepage


CVD Ai
Home
About
CVD
Login
Register
post titile
Posted by kanishka sahu on 21 Sep 2019
post content
post titile
Posted by kanishka sahu on 21 Sep 2019
post content
post titile
Posted by kanishka sahu on 21 Sep 2019
post content
Our Sidebar
You can put any information here you'd like.
Latest Posts
Announcements
Calendars
etc

Register



Login



Scope of Disease Prediction application

It is estimated that more than 70% of people in India are prone to general body diseases like viral, flu, cough, cold. etc, in every 2 months. Because many people don't realize that the general body diseases could be symptoms to something more harmful, 25 % of the population succumbs to death because of ignoring the early general body symptoms. This could be a dangerous situation for the population and can be are alarming. Hence identifying or predicting the disease at the earliest is very important to avoid any unwanted casualties. The currently available systems are the systems that are either dedicated to a particular disease or are in research phase for algorithms when it comes to generalized disease.

The purpose of this system is to provide prediction for the general and more commonly occurring disease that when unchecked can turn into fatal disease. The system applies data mining techniques and ID3 decision tree algorithms. This system will predict the most possible disease based on the given symptoms and precautionary measures required to avoid the aggression of disease, it will also help the doctors analyses the pattern of presence of diseases in the society. In this project, the disease prediction system will carry out data mining in its preliminary stages, the system will be trained using machine learning and data mining.

Advantages & Disadvantage

Advantages

- User can search for doctor's help at any point of time.
- User can talk about their Heart Disease and get instant diagnosis.
- Doctors get more clients online.
- Very useful in case of emergency.

Disadvantages

- Accuracy Issues: A computerized system alone does not ensure accuracy, and the warehouse data is only as good as the data entry that created it.
- The system is not fully automated, it needs data from user for full diagnosis.

Hardware Required

Deep learning is a very CPU intensive program-sequel thing to be running, so be prepared to shell out a lot of money for a good enough system. Here are some system requirements to adhere to (parenthesis is what you can maybe get away with)

- Quad core Intel Core i7 Skylake or higher (Dual core is not the best for this kind of work, but manageable)
- 16GB of RAM (8GB is okay but not for the performance you may want and or expect)
- M.2 PCIe or regular PCIe SSD with at least 256GB of storage, though 512GB is best for performance. The faster you can load and save your applications, the better the system will perform. (SATA III will get in the way of the system's performance)
- Premium graphics cards, so things with GTX 980 or 980Ms would be the best for a laptop, and 1080s or 1070s would be the best for the desktop setup. (try not to sacrifice too much here. While a 980TI or a 970m may be cheaper, this is also a critical part of the system, and you'll see a performance drop otherwise.

Conclusion

The purpose of creating this project is to solve the problem for the mankind. I ma very excited about the machine learning and artificial intelligence. This project gave me an idea and confidence that how algorithms are working to give a great prediction. In this flask gave us a user interface to interact with user and easy to use web-based application that is also a best part of this project. At the last I would like to thanks my all teachers, friends and online documentation that helped me a lot for creating the project.

Thankyou

Reference

- https://flask.palletsprojects.com/en/1.1.x/)
- Stack Overflow (https://stackoverflow.com/)
- Python Docs (https://docs.python.org/3/)
- Kaggle (https://www.kaggle.com/)
- Keras (https://keras.io/)
- Github (https://github.com/)
- American association of hearth health
- W3school.com