Q&A Chatbot Project Manish Garg

October 5th, 2019

# Definition

**Project Overview**

Messaging and voice-controlled devices are the next big platforms, and conversational computing has a big role to play in creating engaging augmented and virtual reality experiences.

Chatbots are now used in apps because of the numerous benefits they have; for

example, users don't need to install different varieties of apps on their mobile. If

there is a chatbot that provides you the news, then you can ask for news that is on

CNN or The Economic Times. Big tech giants such as Facebook, Hike, WeChat,

Snapchat, Slack, and so on provide chatbots for better customer engagement. They

achieve this by making a chatbot that one can guide their customers in order to

perform some operations; it also provides useful information about the product and

its platforms.

There are many advantages of using conversational chatbots, as shown in the following list:

* Personalized assistance: Creating a personalized experience for all customers might be a tedious task, but not doing so can make a business suffer. Conversational chatbots are a convenient alternative to providing a personalized experience to each and every customer.
* Around-the-clock support: Using customer service representatives 24/7 is expensive. Using chatbots for customer service out of office hours removes the need to hire extra customer representatives.
* Consistency of responses: Responses provided by the chatbot are likely to be consistent, whereas responses given to the same questions by different customer service representatives are likely to vary. This removes the need for a customer to call multiple times if they are not satisfied with the answer provided by a customer service representative.
* Patience: While customer service representatives might lose their patience when attending to a customer, this is not a possibility for a chatbot.
* Querying records: Chatbots are much more efficient in querying records than human customer service representatives.

Chatbots are of different types based on the approach of development:

• Retrieval-based approach

• Generative-based approach

In a **retrieval-based** approach, we need to define the set of predefined responses, and we will apply some kind of heuristics on predefined responses so that the chatbot can generate the best possible answers for the given questions. The answers are very dependent on the input question and the context of that input question. The heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers.

Pro’s

1. No grammatical or meaning less errors as we store the answers
2. Works 100% well for the business problems and customer satisfaction and attention can be gained
3. Super easy to build these models as we don’t require huge data.

Con’s

1. These systems don’t generate any new text, they just pick a response from a fixed set.
2. A lot of hard coded rules have to be written so not much intelligent.

In the **generative-based** approach, there aren't any predefined responses given to the chatbot. The chatbot generates the responses from scratch. In order to build the generative-based chatbot, we need to provide a lot of data and the machine will learn how to answer the questions asked by users just by seeing the data.

Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, we “translate” from an input to an output (response).

Pro’s

1. No need to worry about the predefined responses and the rules.

Con’s

1. Super difficult to implement these and the output may not be accurate (grammatical / meaning less errors may occur)
2. Not applicable for the business problem (unless you are providing a service which may require text summarization techniques)
3. Huge data is required to train these models.

Chatbots can be build related to the conversational domain:

• Open domain

• Closed domain

**Open domain** is the place where the chat conversation can go anywhere, users can type/ask anything. There isn’t necessarily have a well-defined goal or intention. The domain of the conversation is not fixed. You can talk about life, jobs, travelling, family, and so on.

Developing an open domain chatbot is difficult because ideally, this kind of chatbot can answer every question from any kind of domain with human-level accuracy.

**Closed domain** is the place where you are solving a particular business problem (The business could be in any sector/industry). In this kind of conversation, where we have restricted the areas we can talk about.

If a financial institute such as a bank launches a chatbot for their customers, then the developed chatbot cannot answer questions such as can you tell me what the weather in Singapore is today? But it helps you check the procedure of applying for a credit card, and this is because a chatbot can understand questions related to a specific domain.

The closed domain bots have the limited functionalities/ services based on the business problem.

A chatbot using the generative-based approach, which operates on the open domain is an example of **Artificial General Intelligence (AGI).**

A chatbot using the **generative-based approach, which operates on the closed domain** are very useful and common in industry. The development of this kind of chatbot takes less coding time, and the quality of the answers improves as well. If we want our chatbot to understand long contexts and intents over a series of questions from the user, then the generative-based approach is the right choice. After training on large corpus and optimization, the chatbot can understand the context and intent of questions as well as be able to ask reasoning types of questions.

The example can be a bank chatbot to apply for a home loan from a bank.

**Problem Statement**

In this project our primary objective is to build a basic **question-answering system (QA systems)**, chatbot.

QA systems can also be of different types like:

**Information retrieval (IR) -based question answering**

It fully relies on the huge amount of information available as text on the Web or in specialized collections such as PubMed. The method processes the question to determine the likely answer type (often a named entity like a person, location, or time)

IR based gets the answer from the documents collected, again it doesn’t generate the answer, it just does copy-paste from the documents, so if the text is not present in the documents, these models can’t give the answer.

They can also **fetch data from large data base** by converting the natural language query into structured database query and returning back the response to the user.

**Story based Question and answering**

This is more like asking the question based on the passage/story given here the input is given in form of triplets for building models. (Story, Question, Answer)

This project intends to use Deep Learning and other Artificial Intelligence algorithms in order to solve the Question Answering System problem.

**Our QA systems chat will be based on Story based Question and answering.**

In this system there will be short story followed by set of questions. The bot will answer the questions by **understating the context, remembering the story and apply logical reasoning.**

Question Answer systems are very useful as they allow users to enter a query based on some facts or stories and the system tries to use the context in the supporting facts and stories to answer the questions effectively instead of just giving out the best suited keywords.

Besides, most of the problems in artificial intelligence and Natural language processing can all be modelled as a question answering problems. For example, the task of text summarization can be modelled as a question answering task in the sense that if the user asks the system "What is the summary of the text?", it can answer the user by providing the appropriate summary.

Metrics

The performance of each classification model is evaluated using statistical measures;

classification accuracy, recall, sensitivity and specificity. It is using true positive (TP), true negative (TN), false positive (FP) and false negative (FN).

We will also summaries the performance of a classification algorithm using confusion

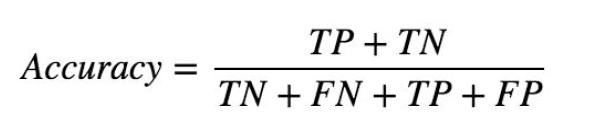
metrics as calculating a confusion matrix can give you a better idea of what your

classification model is getting right and what types of errors it is making.

|  |  |  |
| --- | --- | --- |
|  | **Predicted - Positive** | **Predicted - Negative** |
| **Actual - Positive** | True Positive | False Negative |
| **Actual - Negative** | False Positive | True Negative |

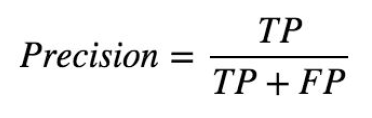
Classification accuracy is defined as the ratio of the number of correctly classified cases and is equal to the

sum of TP and TN divided by the total number of cases (TN + FN + TP + FP).



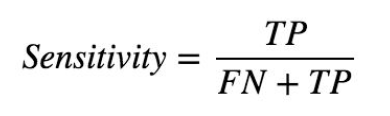
Precision is defined as the number of true positives (TP) over the number of true positives plus the number of

false positives (FP).



Sensitivity refers to the rate of correctly classified positive and is equal to TP divided by the sum of TP and FN.

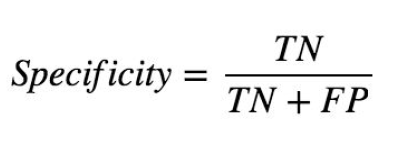
Sensitivity may be referred as a True Positive Rate.



Specificity refers to the rate of correctly classified negative and is equal to the ratio of TN to the sum of TN and

FP

<https://www.sciencedirect.com/science/article/pii/S1877050918308226>



# Data

In order to develop a chatbot, we are using two datasets. These datasets are as follows:

• Cornell Movie-Dialogs dataset

• bAbI dataset

**Cornell Movie-Dialogs dataset**

This dataset has been widely used for developing chatbots. This is available at –

Cornell Movie-Dialogs corpus from this link:

<https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html>

This corpus contains a large metadata-rich collection of fictional conversations extracted from raw

movie scripts.

This corpus has 220,579 conversational exchanges between 10,292 pairs of movie characters. It involves 9,035 characters from 617 movies. In total, it has 304,713 utterances. This dataset also contains movie metadata. There are the following types of metadata:

• Movie-related metadata includes the following details:

* + - Genre of the movie
    - Release year
    - IMDb rating

• Character-related metadata includes the following details:

* + - Gender of 3,774 characters
    - Total number of characters in movies

**The bAbI dataset**

This dataset is built by Facebook AI Research (FAIR), where AI stands for artificial intelligence. This dataset belongs to the bAbI project.

download the dataset from

<https://research.fb.com/downloads/babi/>

The goal of the bAbI project is to try to build an automatic text understanding and reasoning system. This dataset consists of the following sub datasets:

• The (20) QA bAbI tasks

• The (6) dialog bAbI tasks

• The Children's Book Test

• The Movie Dialog dataset

• The WikiMovies dataset

• The Dialog-based Language Learning dataset

• The SimpleQuestions dataset

• HITL Dialogue Simulator

We will be using only one subset here, which is the (20) QA bAbI tasks because it is the one that's most useful for building the chatbot.

If we look at this sub dataset in detail. Here, 20 different tasks have been performed using this (20) QA bAbI dataset. Let's see what these tasks are. These tasks give machines the capacity to perform some reasoning, and based on that, the machine can answer a question. You can refer to the task name given in the following figure:

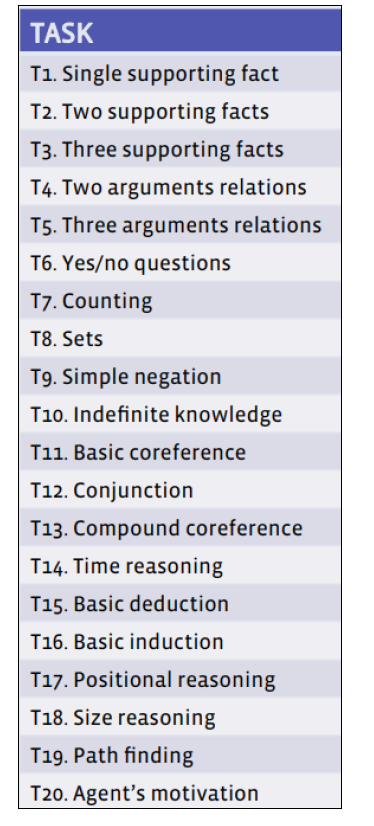


Image source: <http://www.thespermwhale.com/jaseweston/babi/abordes-ICLR.pdf>

# Algorithms and Techniques

## Base Line model

The Deep Learning Neural Network is used as a binary classifier. A neural network takes in inputs, which are then processed in hidden layers using weights that are adjusted during training. Then the model spits out a prediction. The weights are adjusted to find patterns in order to make better predictions. The user does not need to specify what patterns to look for — the neural network learns on its own. The algorithm outputs an assigned probability for each class; this can be used to reduce the number of false positives using a threshold.

Once the data is scaled we will do train test split where 30% of the data is divided up as test data. To maintain the balance of target classes both in train and test we use Stratified Shuffle splitting.

The following parameters can be tuned to optimize the classifier:

* Classification threshold
* Training length (number of epochs)
* Batch size (how many data points to look at once during a single training step)
* Solver type (what algorithm to use for learning)
* Learning rate (how fast to learn; this can be dynamic)
* Momentum (takes the previous learning step into account when calculating the next one)

Neural network architecture

* Number of layers
* Layer types (Dense)
* Layer parameters

## seq2seq model

## End-to-end memory networks

# Benchmark Model

The given dataset is a supervised classification learning problem for which we first tried with simple K-nearest neighbours to get base value of score.

KNN is an algorithm that's used in pattern recognition for object classification based on the characteristics of the nearest objects. An object is classified according to the majority of the votes of its neighbouring k cluster. **KNeighborsClassifier** gave model accuracy for the data as 0.5934.

# Methodology

## Implementation

The project workflow is as shown below:



We first started with the exploratory analysis by seeing distribution of some of the feature that seems to be relevant to the problem. We also did a correlations check using scatterplot matrix in this process.

In the **data cleaning** to identify and fill the missing values. In the project data we have identified ‘ca’ column had 4 missing values and ‘thal’ has 2 missing values. We have filled them with mean.

In the **Data transformation** step we have scaled the data using the StandardScaler so that all features are in similar range to improve the prediction's accuracy.

Before we began the model evaluation, we split our full dataset into training and validation data sets with 70:30 ratio using stratified strategy to ensure balance target class splits.

We used **Deep Neural Network** for binary classification and used **Random search** approach to get the optimized hyper parameters. Once we optimized the parameters we trained the model. After training we did validation and verified the model on key metrices.

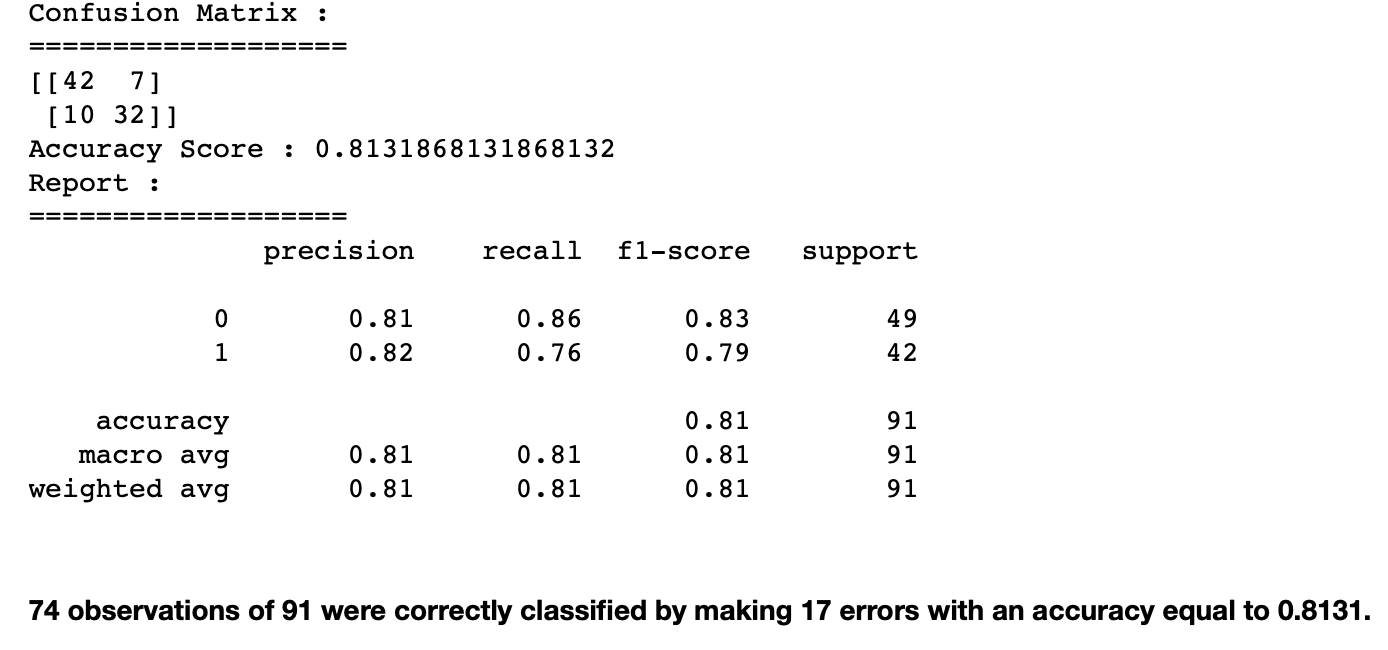
We also evaluated our data set using **XGBoost classifier** to check if we can get higher accuracy. For XGBoost classifier modelling we also used **Grid search** approach to get the optimized hyper parameters. Once we optimized the parameters we trained the model. After training we did validation and verified the model on key metrices.

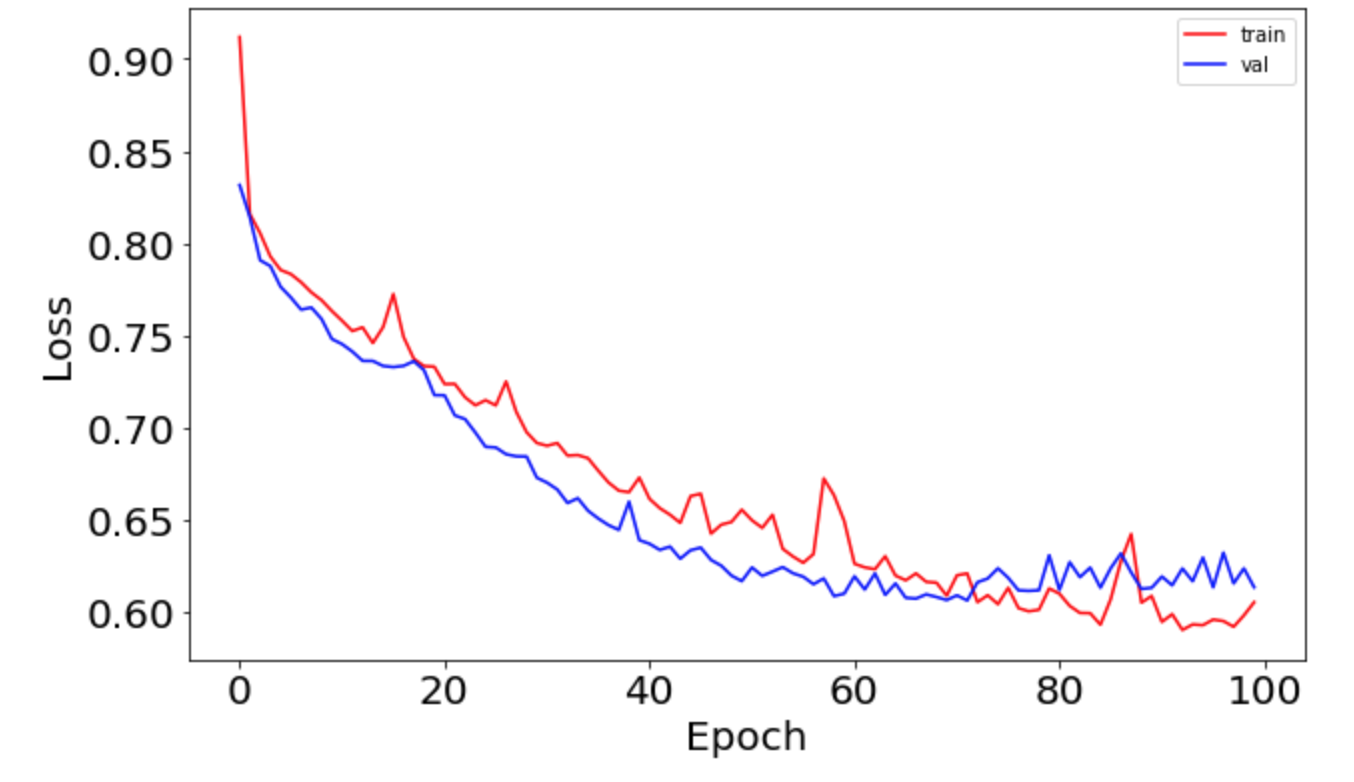
## Refinement and Results

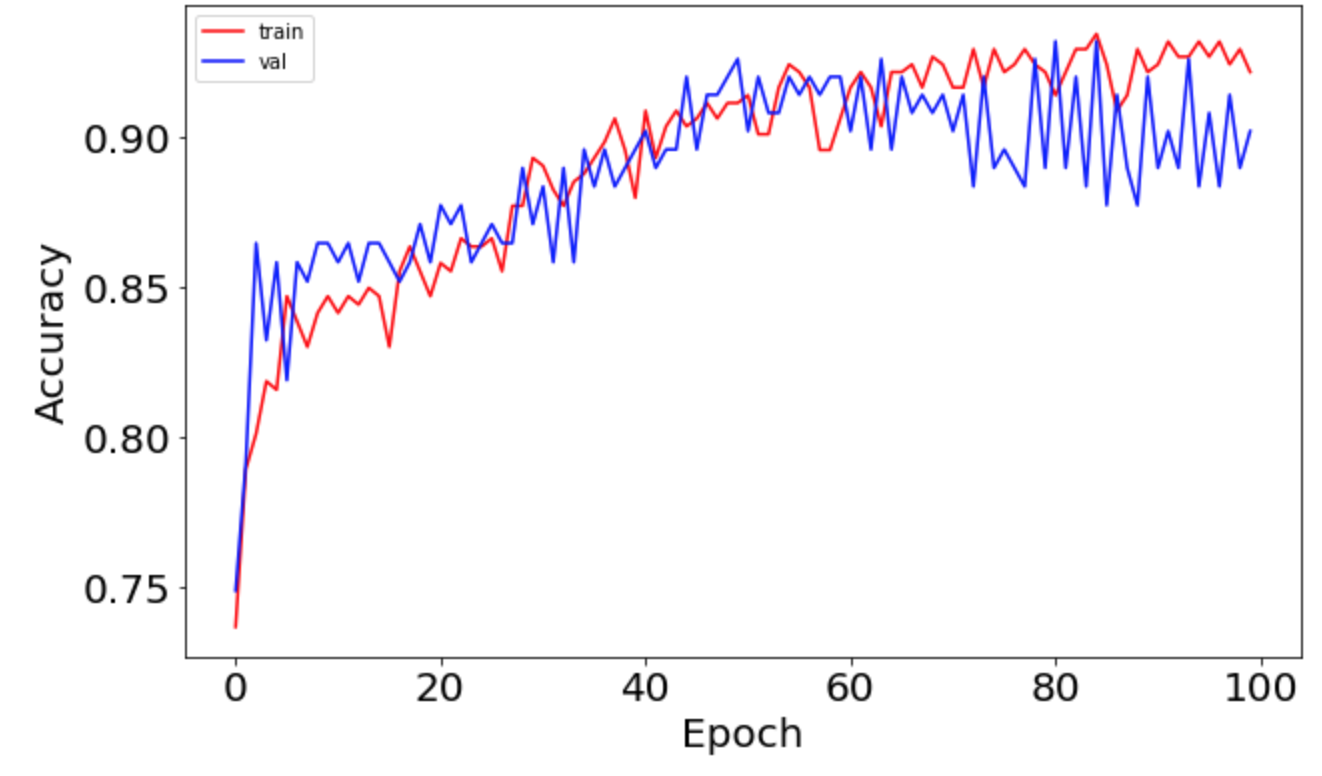
### Deep Neural Network

We performed hyper tuning of parameters of **Deep Neural Network** for binary classification by implementing the interface *tensorflow.keras.wrappers.scikit\_learn.KerasClassifier* and using the *sklearn.model\_selection.RandomizedSearchCV*

|  |  |  |
| --- | --- | --- |
| Parameter | Values Tested | Best Value |
| batch\_size | [10, 20, 40] | 40 |
| epochs | [100, 500, 1000] | 100 |
| optimizer | ['SGD', 'RMSprop', 'Adagrad', 'Adam'] | Adam |
| learning\_rate | [0.001, 0.01, 0.1] | 0.001 |
| init\_mode | ['uniform', 'normal', 'zero'] | 'normal' |



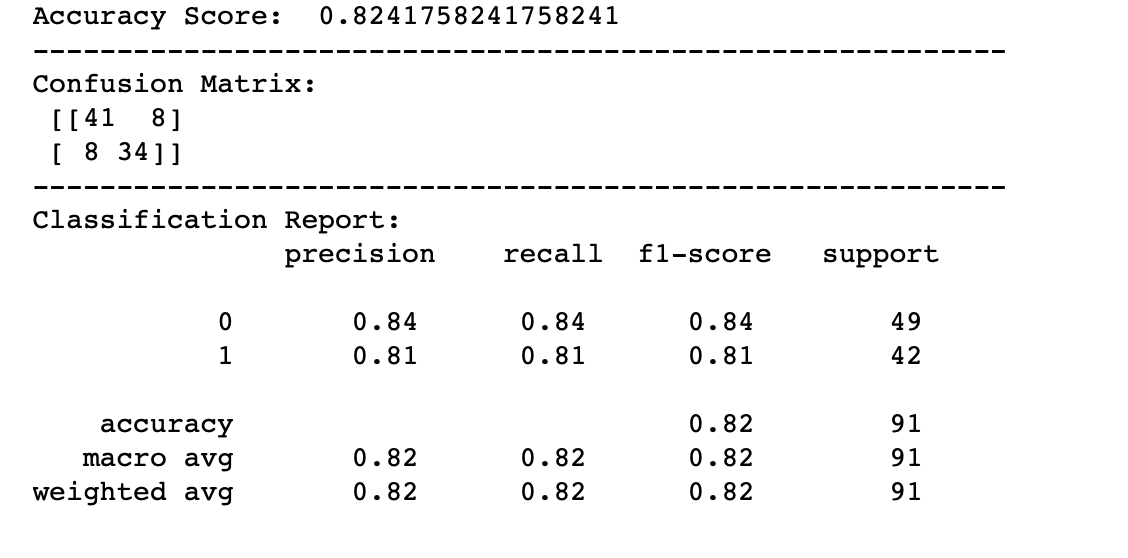




### XGBoost Classifier

We performed hyper tuning of parameters for binary classification by using the *sklearn.model\_selection.GridSearchCV*

|  |  |  |
| --- | --- | --- |
| Parameter | Values Tested | Best Value |
| colsample\_bytree | [0.6, 0.8, 1.0] | 0.6 |
| gamma | [0.5, 1, 1.5, 2, 5] | 5 |
| max\_depth | [3, 4, 5] | 3 |
| min\_child\_weight | [1, 5, 10] | 5 |
| subsample | [0.6, 0.8, 1.0] | 0.8 |



The performance of the model is k-fold cross validated which is the gold-standard for evaluating the performance of a model on unseen data with k set to 3, 5, or 10. Stratified cross validation is used to enforce balanced class distributions in the splits.

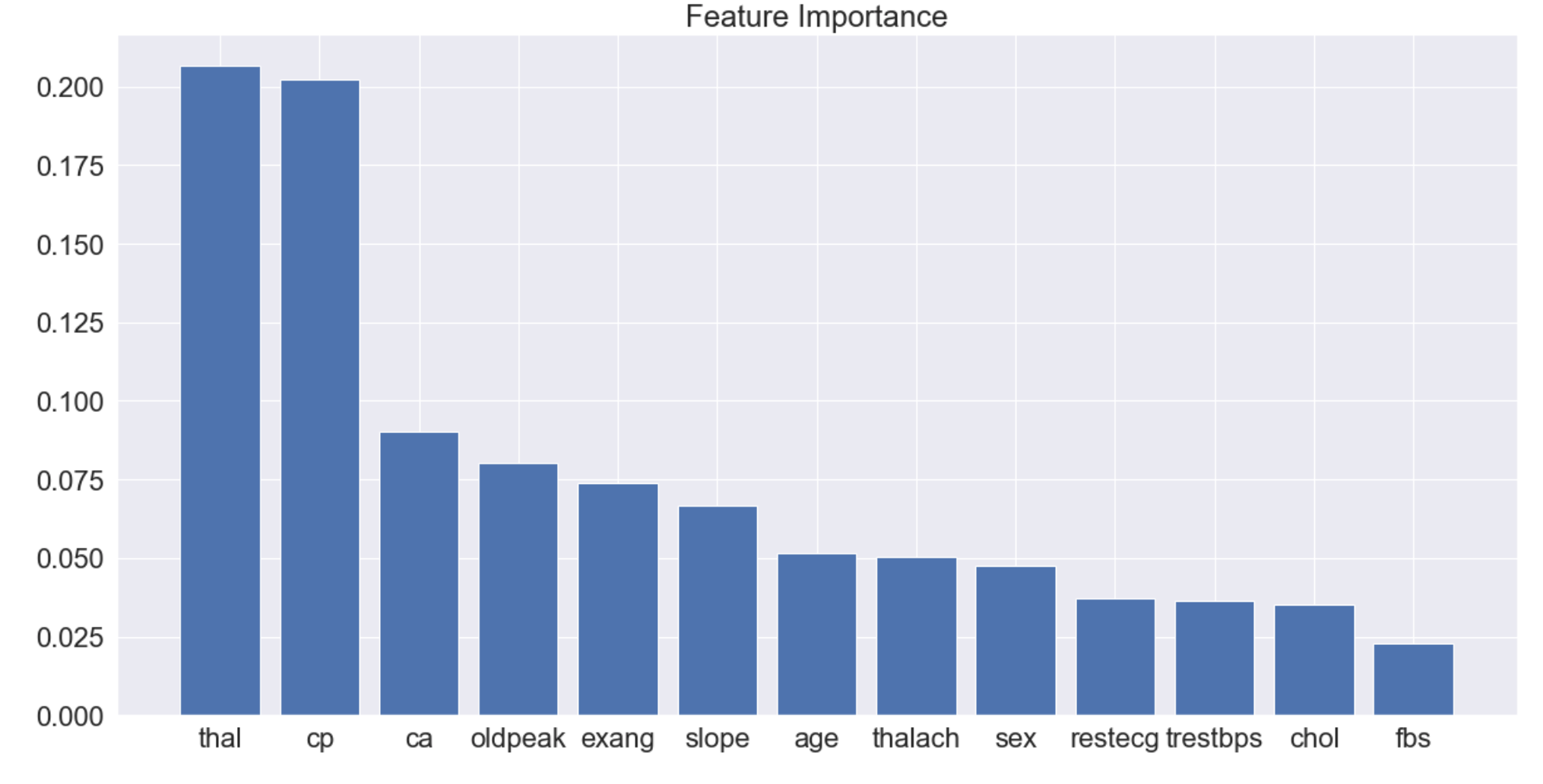
After running cross validation, we end up with k different performance scores that we can summarize using a mean and a standard deviation.

The result is a more reliable estimate of the performance of the algorithm on new data with each fold. It is more accurate because the algorithm is trained and evaluated multiple times on different data.

**Based on above results the model looks quite robust as the mean accuracy is quite high Accuracy: 83.40% (6.80%)**

## Conclusion

The feature importance is visualized as:



The graph depicts the most influencing factors for the Heart Disease. Higher values mean the feature is more important in determining the Heart Disease.

As per the above graph 'thal','cp' and 'ca' are the highest influencing factor for the Heart Disease. This also matches with our initial finding in the data visualization section.

* 'cp' is Chest pain type, which determines the type angina. This is one of the major issues reported by the Heart Disease patients. Therefore, Heart Disease dependency on this feature is natural.
* 'thal' Thalassemia is an inherited blood disorder characterized by less hemoglobin and fewer red blood cells in your body than normal. Thalassemia is one of the main causes of Heart disease as Low levels of hemoglobin may cause anemia. Anemia is associated with a special risk in proatherosclerotic conditions and heart disease.
* 'ca' A noninvasive method for coronary artery diseases diagnosis. It indicates Number of major vessels to be get narrowed and are not able to convey enough fresh blood to this blood-pumping organ. This is also quite major factor to determine the heart disease.

**Model evaluations:**

We have used three different models for classification- KNN, Neural Network binary classifier, and XGBoost classifier. Based upon various evaluations matrix we see the XGBoost classifier performs best with highest accuracy of cross validation accuracy score 83.40%. Also, it is much easier to tune XGBoost classifier for the given problem and the dataset.

### Reflection

The most important and time-consuming part of the problem was data cleansing and processing as we need to ensure that we have correctly prepared the data well for the algorithm to work in its full potential.

Once the data was prepared and ready, the next challenge was to pick the optimized parameters for the selected algorithm and the data set.

As the data set was complex but small so may be Deep Neural Network was not the correct algorithm for the problem as it has not given any major advantage and XGBoost classifier has given better results.

The other issue I see is the inconsistency of the models to given consistent results in-spite of similar data set, parameters, setting of initializers and seed values.

There are many other ways to improve the model further like :

* Having more training data.
* Better model tuning using various hyper parameters
* Trying more different Algorithms