SRM INSTITUTE OF SCIENCE AND TECHNOLOGY

SCHOOL OF COMPUTING

DEPARTMENT OF DATASCIENCE AND BUSINESS SYSTEMS

18CSC305J ARTIFICIAL INTELLIGENCE

MINI PROJECT REPORT

Title

Name: Gollapudi Vyshnavi

Register Number: RA1911027010112

Mail ID: gg2119@srmist.edu.in

Department: CSE

Specialization: Big Data Analytics

Semester: VIth

Team Members

Name: Geetaanjali GNS **Registration Number RA1911027010113**

Name: Kolisetty Yeshwanth Kumar **Registration Number RA1911027010111**

Name: Ketala Snehal Kumar **Registration Number RA1911027010104**

Content Page

Chapter 01, Introduction and Motivation

- Introduction:The project involved analysis of the heart disease patient dataset with proper data processing. Then, different models were trained and and predictions are made with different algorithms KNN, Decision Tree, Random Forest,SVM,Logistic Regression etc This is the jupyter notebook code and dataset I've used for my Kaggle kernel 'Binary Classification with Sklearn and Keras'

  We've used a variety of Machine Learning algorithms, implemented in Python, to predict the presence of heart disease in a patient. This is a classification problem, with input features as a variety of parameters, and the target variable as a binary variable, predicting whether heart disease is present or not.

- Motivation: preventing Heart diseases has become more than necessary. Good data-driven systems for predicting heart diseases can improve the entire research and prevention process, making sure that more people can live healthy lives. This is where Machine Learning comes into play. Machine Learning helps in predicting Heart diseases, and the predictions made are quite accurate.

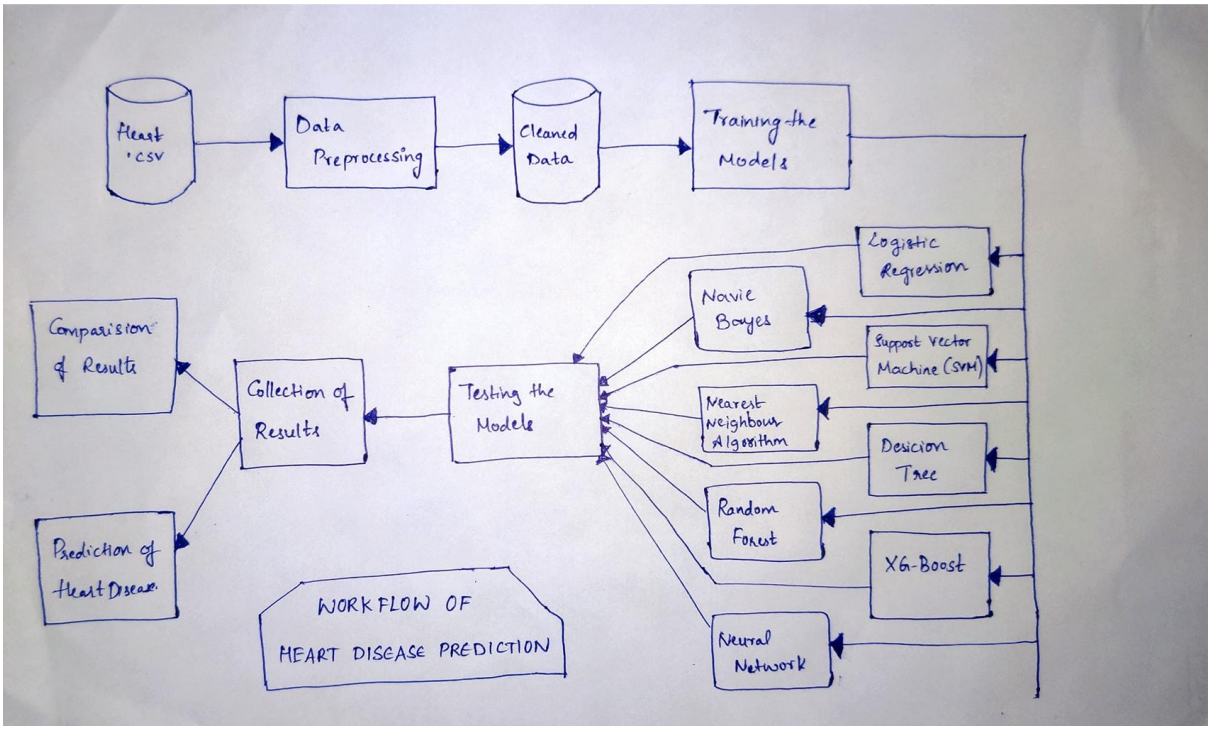
Chapter 02, Review of Existing methods and their Limitations.

- Methods:
    1. Logistic Regression
    2. Naive Bayes
    3. SVM
    4. NNA
    5. Decision Tree
    6. Random Forest
    7. Neural Network
    8. XGBoost
- Limitations
    1. Logistic Regression: assumption of linearity between the dependent variable and the independent variables
    2. Naive Bayes: assumption of independent predictors. In real life, it is almost impossible that we get a set of predictors which are completely independent.
    3. SVM: does not perform very well when the data set has more noise i.e. target classes are overlapping
    4. NNA:
    5. Decision Tree: 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
    6. Neural Network: Complex to Perform
    7. XGBoost:

Chapter 03, Proposed Method with System Architecture / Flow Diagram.

- Method:
    1. Collected the dataset named heart.csv from Kaggle to implement our M.L. algorithms.

2. Imported the dataset into google colab and then understood the data and the columns.
3. performed the exploratory data analysis on target variables and independent variables.
4. Cleaned the dataset.
5. Split the data for training and testing.
6. Trained the train dataset with models, namely, logistic regression, Naive Bayesian, Support Vector Machine, KNN , Random Forest, XG-boost, Neural Networks.
7. Tested the models on the train dataset.
8. Got the highest accuracy for Random Forest algorithm.



Chapter 04, Modules Description.

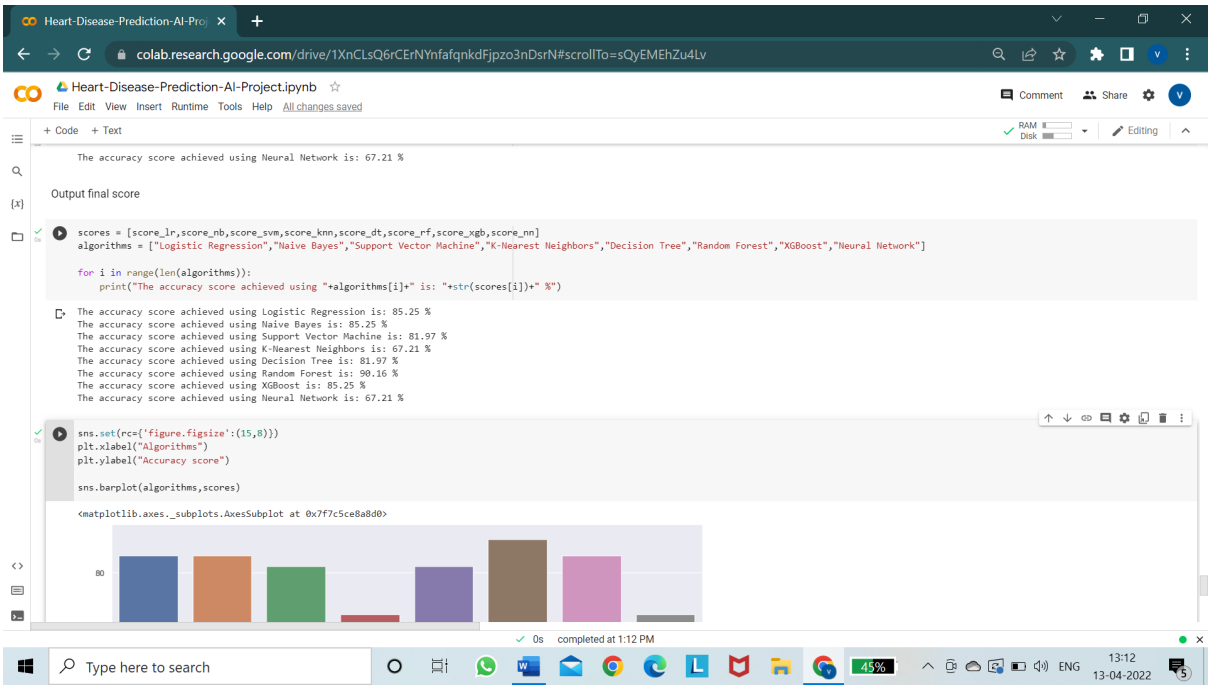
- Logistic Regression (Scikit-learn)
- Naive Bayes (Scikit-learn)
- Support Vector Machine (Linear) (Scikit-learn)
- K-Nearest Neighbours (Scikit-learn)
- Decision Tree (Scikit-learn)
- Random Forest (Scikit-learn)
- XGBoost (Scikit-learn)
- Artificial Neural Network with 1 Hidden layer (Keras)
- Accuracy achieved: 95% (Random Forest)

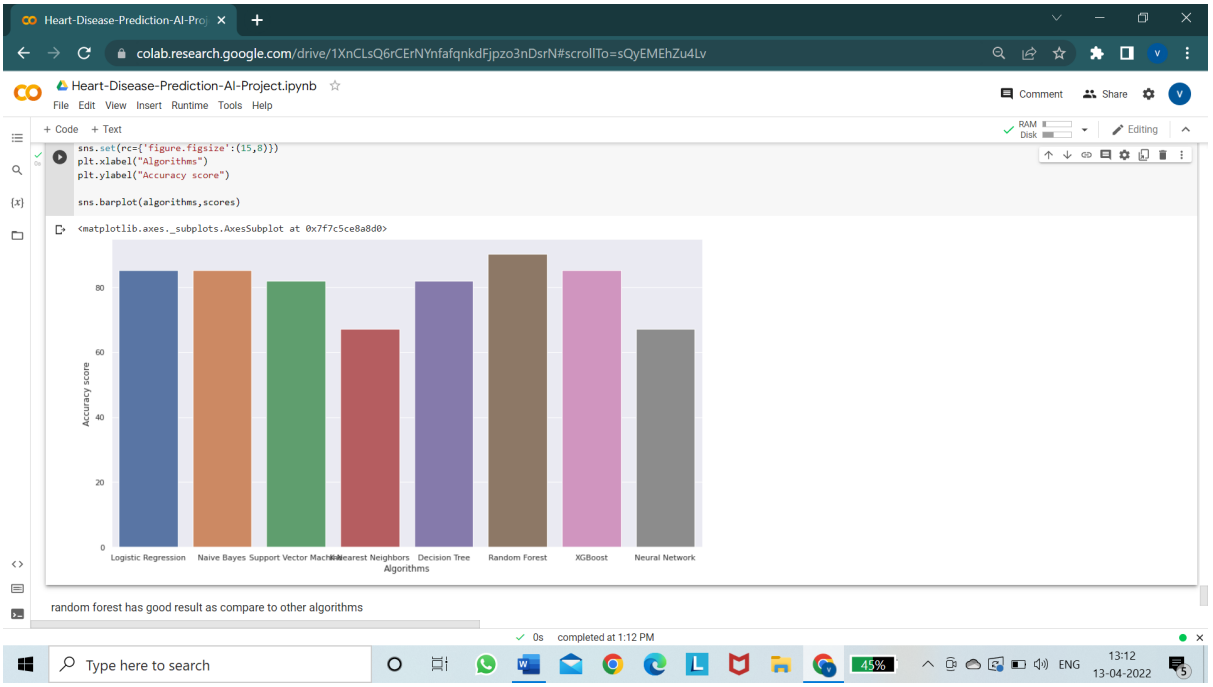
Chapter 05, Implementation Requirements.

- Google Collaboratory
- Heart.csv dataset

- Python Libraries
    1. Numpy
    2. Pandas
    3. Matplotlib
    4. seaborn
    5. OS
    6. warnings
    7. sklearn.metrics
    8. sklearn.linear_models
    9. sklearn.naive_bayes
    10. sklearn.neighbors
    11. sklearn.tree
    12. sklearn.ensemble
    13. XGBoost
    14. Keras.models
    15. Keras.layers

Chapter 06, Output Screenshots.





Conclusion

On Implementing multiple models per say, Logistic Regression (Scikit-learn), Naive Bayes (Scikit-learn), Support Vector Machine (Linear) (Scikit-learn), K-Nearest Neighbours (Scikit-learn), Decision Tree (Scikit-learn), Random Forest (Scikit-learn), XGBoost (Scikit-learn), Artificial Neural Network with 1 Hidden layer (Keras), Accuracy achieved: 95% (Random Forest), We found that Random Forest is the best algorithm to predict if a person has a Heart Disease.

References:

https://www.javatpoint.com/logistic-regression-in-machine-learning

https://www.upgrad.com/blog/naive-bayes-explained/

https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm

https://www.geeksforgeeks.org/k-nearest-neighbours/

https://en.wikipedia.org/wiki/Decision_tree

https://www.ibm.com/cloud/learn/random-forest

https://www.geeksforgeeks.org/xgboost/

Appendix A – Source Code

```
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns


%matplotlib inline


import os

print(os.listdir())


import warnings

warnings.filterwarnings('ignore')

dataset = pd.read_csv('heart.csv')

type(dataset)

dataset.shape

dataset.head(5)

dataset.sample(5)

dataset.describe()

dataset.info()

info = ["age","1: male, 0: female","chest pain type, 1: typical
angina, 2: atypical angina, 3: non-anginal pain, 4:
asymptomatic","resting blood pressure"," serum cholestoral in
mg/dl","fasting blood sugar > 120 mg/dl","resting
electrocardiographic results (values 0,1,2)"," maximum heart rate
achieved","exercise induced angina","oldpeak = ST depression
induced by exercise relative to rest","the slope of the peak
exercise ST segment","number of major vessels (0-3) colored by
flourosopy","thal: 3 = normal; 6 = fixed defect; 7 = reversable
defect"]
```

```
for i in range(len(info)):
    print(dataset.columns[i]+":\t\t\t"+info[i])
dataset["target"].describe()
dataset["target"].unique()
print(dataset.corr()["target"].abs().sort_values(ascending=False)
)
y = dataset["target"]


sns.countplot(y)



target_temp = dataset.target.value_counts()


print(target_temp)
print("Percentage of patience without heart problems:
"+str(round(target_temp[0]*100/303,2)))
print("Percentage of patience with heart problems:
"+str(round(target_temp[1]*100/303,2)))
dataset["sex"].unique()
sns.barplot(dataset["sex"],y)
dataset["cp"].unique()
sns.barplot(dataset["cp"],y)
dataset["fbs"].describe()
dataset["fbs"].unique()
sns.barplot(dataset["fbs"],y)
dataset["restecg"].unique()
sns.barplot(dataset["restecg"],y)
dataset["exang"].unique()
sns.barplot(dataset["exang"],y)
```

```
dataset["slope"].unique()

dataset["ca"].unique()

sns.countplot(dataset["ca"])

sns.barplot(dataset["ca"],y)

dataset["thal"].unique()

sns.barplot(dataset["thal"],y)

sns.distplot(dataset["thal"])

from sklearn.model_selection import train_test_split


predictors = dataset.drop("target",axis=1)

target = dataset["target"]


X_train,X_test,Y_train,Y_test =
train_test_split(predictors,target,test_size=0.20,random_state=0)

X_train.shape

X_test.shape

Y_train.shape

Y_test.shape

from sklearn.metrics import accuracy_score

fromklearn.linear_model s import LogisticRegression


lr = LogisticRegression()


lr.fit(X_train,Y_train)


Y_pred_lr = lr.predict(X_test)

Y_pred_lr.shape

score_lr = round(accuracy_score(Y_pred_lr,Y_test)*100,2)
```

```
print("The accuracy score achieved using Logistic Regression is:
"+str(score_lr)+" %")

from sklearn.naive_bayes import GaussianNB


nb = GaussianNB()


nb.fit(X_train,Y_train)


Y_pred_nb = nb.predict(X_test)

Y_pred_nb.shape

score_nb = round(accuracy_score(Y_pred_nb,Y_test)*100,2)


print("The accuracy score achieved using Naive Bayes is:
"+str(score_nb)+" %")

from sklearn import svm


sv = svm.SVC(kernel='linear')


sv.fit(X_train, Y_train)


Y_pred_svm = sv.predict(X_test)

Y_pred_svm.shape

score_svm = round(accuracy_score(Y_pred_svm,Y_test)*100,2)


print("The accuracy score achieved using Linear SVM is:
"+str(score_svm)+" %")

from sklearn.neighbors import KNeighborsClassifier


knn = KNeighborsClassifier(n_neighbors=7)
```

```
knn.fit(X_train,Y_train)

Y_pred_knn=knn.predict(X_test)

score_knn = round(accuracy_score(Y_pred_knn,Y_test)*100,2)


print("The accuracy score achieved using KNN is:
"+str(score_knn)+" %")

from sklearn.tree import DecisionTreeClassifier


max_accuracy = 0



for x in range(200):

    dt = DecisionTreeClassifier(random_state=x)

    dt.fit(X_train,Y_train)

    Y_pred_dt = dt.predict(X_test)

    current_accuracy =
round(accuracy_score(Y_pred_dt,Y_test)*100,2)

    if(current_accuracy>max_accuracy):

        max_accuracy = current_accuracy

        best_x = x


#print(max_accuracy)

#print(best_x)



dt = DecisionTreeClassifier(random_state=best_x)

dt.fit(X_train,Y_train)

Y_pred_dt = dt.predict(X_test)

print(Y_pred_dt.shape)

score_dt = round(accuracy_score(Y_pred_dt,Y_test)*100,2)
```

```
print("The accuracy score achieved using Decision Tree is:
"+str(score_dt)+" %")

from sklearn.ensemble import RandomForestClassifier


max_accuracy = 0



for x in range(2000):

    rf = RandomForestClassifier(random_state=x)

    rf.fit(X_train,Y_train)

    Y_pred_rf = rf.predict(X_test)

    current_accuracy =
round(accuracy_score(Y_pred_rf,Y_test)*100,2)

    if(current_accuracy>max_accuracy):

        max_accuracy = current_accuracy

        best_x = x


#print(max_accuracy)

#print(best_x)


rf = RandomForestClassifier(random_state=best_x)

rf.fit(X_train,Y_train)

Y_pred_rf = rf.predict(X_test)

Y_pred_rf.shape

score_rf = round(accuracy_score(Y_pred_rf,Y_test)*100,2)


print("The accuracy score achieved using Decision Tree is:
"+str(score_rf)+" %")

import xgboost as xgb
```

```
xgb_model = xgb.XGBClassifier(objective="binary:logistic",
random_state=42)

xgb_model.fit(X_train, Y_train)


Y_pred_xgb = xgb_model.predict(X_test)

Y_pred_xgb.shape

score_xgb = round(accuracy_score(Y_pred_xgb,Y_test)*100,2)


print("The accuracy score achieved using XGBoost is:
"+str(score_xgb)+" %")

from keras.models import Sequential

from keras.layers import Dense

model = Sequential()

model.add(Dense(11,activation='relu',input_dim=13))

model.add(Dense(1,activation='sigmoid'))


model.compile(loss='binary_crossentropy',optimizer='adam',metrics
=['accuracy'])

Y_pred_nn = model.predict(X_test)

Y_pred_nn.shape

rounded = [round(x[0]) for x in Y_pred_nn]


Y_pred_nn = rounded

score_nn = round(accuracy_score(Y_pred_nn,Y_test)*100,2)


print("The accuracy score achieved using Neural Network is:
"+str(score_nn)+" %")
```

```
scores =
[score_lr,score_nb,score_svm,score_knn,score_dt,score_rf,score_xg
b,score_nn]

algorithms = ["Logistic Regression","Naive Bayes","Support Vector
Machine","K-Nearest Neighbors","Decision Tree","Random
Forest","XGBoost","Neural Network"]


for i in range(len(algorithms)):

    print("The accuracy score achieved using "+algorithms[i]+"
is: "+str(scores[i])+" %")

sns.set(rc={'figure.figsize':(15,8)})

plt.xlabel("Algorithms")

plt.ylabel("Accuracy score")


sns.barplot(algorithms,scores)
```

Appendix B – GitHub Profile and Link for the Project

https://github.com/Gollapudi-vyshnavi1104/Heart-Disease-Prediction - Vyshnavi

https://github.com/GeetaanjaliGNS/8086-add-two-arrays/blob/main/Heart_Disease_Prediction_AI_Project.ipynb -Geetaanjali