School of Engineering and Applied Science (SEAS), Ahmedabad University

Probabilistic Graphical Models (CSE 516)

Project Report

Group Name: Research Applied Data Science 5
Team Members:

- Kshitij Shekhar(AU20202165)
- Nihar Jani(AU2040205)
- Kalp Ranpura(AU1920134)

I. Introduction

A. Background

Globally, cardiovascular disease is the leading cause of death. Finding useful information from the data that is already available can be aided by data mining from the medical field. Cardiovascular disorders are brought on by heart and blood vessel dysfunction. One-fifth of fatalities have been connected to cardiac causes, and an arrhythmic substrate is the site of the interaction. Using data mining, one can find relevant trends in a vast quantity of is a data. Understanding data and making discoveries from it are the most crucial tasks in data mining. To determine which individual has a higher risk of experiencing a heart attack based on several characteristics, we use Bayesian networks. It belongs to the category of probabilistic graphical models. It is a node and edge-based directional acyclic graph.

B. Motivation

Cardiovascular Diseases are the leading cause of death globally, especially among developed, high-income countries with advanced, standards of living and (ironically) access to high-quality healthcare. Many factors contribute to CVD, and a large majority of them cannot be understood by a layman. One of the many advantages and trickle-down effects of advances in AI and its derivatives(like probabilistic graphical models) is the democratization of data and its applications. While classic ML models like SVMs and Random Forests are able to solve problems like the prediction of heart disease given some features, they are often of limited use as they are often not very explainable. Deep Learning models often produce state-of-the-art results for almost any problem that has enough data(enough is an understatement, they often require thousands of data points to produce state-of-the-art results), but are like black boxes - we don't understand how they come to make the correct predictions. But probabilistic models like Bayesian Networks, while limited in their scope of applications pride themselves on interpretability. The project entails finding the relationships between 14 features of the dataset and performing inference. A Bayes Net would be able to show us the conditional dependencies and independencies among the features of the dataset.

C. Contribution

The paper's authors implemented the Bayes Net in R using the bnlearn library. We will implement the same in pgmpy and attempt to match and/or beat the performance metrics the authors were able to produce.

II. Methodology

A. Analytical Overview/Product Details/

Our product is a Web App made using the python framework Anvil that allows users to make web applications using only python and a google colab notebook. Its friendly and simple user interface includes label fields to describe the features of the data set that was used to make predictions and corresponding to each label field, a text box to facilitate input from the user. It has a PREDICT button which, using the server connection made from the colab notebook, performs inference on the input data and outputs a prediction(1 for positive heart disease diagnosis and 0 for a negative diagnosis). It also has the functionality to run forever, for as long as the user would like.

B. Mathematical Analysis/Working of the Product/Framework

Our project entails building a Bayesian Network to predict the onset of Cardiovascular Disease given a set of features that a patient must enter into an interactive Web App. Our model was trained on the UCI Machine Learning Repository's Cleveland heart disease data set that was put together by doctors in a clinic in Cleveland,USA. Our Bayes Net learned its structure from the training data set, which was 80% of the total data set. We did parameter training on the same training set in order to learn the conditional probability distributions that describe the relationships in the data set. we then tested the model on certain evaluation metrics like accuracy, precision and recall.

We also tried reproducing the model in the research paper that we referenced (Muibideen et.al.) but did not get as good an accuracy as the authors did. The authors used the bnlearn library for the R programming language for their structure and parameter learning and we expect the differences in model performance to arise from there.

For the uninitiated, accuracy is a common statistical method of evaluating models. It's defined to be:

$$\frac{TruePositives+TrueNegatives}{TruePositives+TrueNegatives+FalsePostives+FalseNegatives}$$

This means that accuracy is the number of correctly classified examples divided by the total number of examples

Precision is fraction of the number of True Positives divided by the sum of the True Positives and the False Positives. i.e :

$$\frac{TruePositives}{TruePositives+FalsePositives}$$

Recall is

$$\frac{TruePositives}{TruePositives+FalseNegatives}$$

There exists a precision recall tradeoff when we usually build models that can be evaulated with statistical techniques. As one increases, the other decreases, the architect of the model must decide on which one is more important for the task at hand and thus go about tuning the model to prioritise either one of high precision or high recall.

Anvil allows programmers to upload and deploy their jupyter or colab notebooks without the need for any other web development knowledge. To integrate anvil's functionality within the project notebook, we first installed the anvil library, enabled the uplink to the project's notebook so that the anvil server running in the colab notebook can connect with the frontend.

The mathematical models/tools used in our project are Bayesian Networks. We used score based structural learning algorithms for our project. For parameter learning of our model we used Bayesian Estimation, Maximum Likelihood Estimation and Expectation Maximization estimators provided by pgmpy.

Bayesian Networks belong to the category of Probabilistic Graphical Models. They are Directed Acyclic Graphs with nodes representing random variables and edges representing causality relationships between the nodes. Multiple conditional distributions between nodes and their ancestors combine to form the joint distribution that defines a Bayesian Network. Nodes that aren't connected by edges are conditionally independent of one another. Each node has an associated probability function that takes as inputs values of the node's parent variables and outputs a probability distribution that best represents that node. This represents the factorisation operation that is common when defining nodes in a Bayes Net.

$$p(x) = \Pi_{v \in V} p(x_v | x_{pa(v)})$$

where pa(v) represents the parents of the variable v. V here is the set of all nodes in the Bayes Net.
If we wanted to calculate the probability of any node(s) we can use the chain rule:

$$p(X_1 = x_1, ..., X_n = x_n) = \Pi_{v=1}^{n} P(X_v = x_v | X_{v+1} = x_{v+1}, ..., X_n = x_n)$$

Which can further be written as :

$$p(X_1 = x_1, ..., X_n = x_n) = \Pi_{v=1}^{n} P(X_v = x_v | X_j = x_j \text{ for each } X_j \text{ that is a parent of } X_v)$$

The path between any two nodes in a BN is said to be blocked if there is at least 1

III. Codes(only main document not supporting files with comments on each line)

```
# -*- coding: utf-8 -*-
"""PGM_Project.ipynb

Automatically generated by Colaboratory.

Original file is located at
    https://colab.research.google.com/drive/1iWEB9jLjh6pEaY3X2R-1ZdhwqzOu-jF
"""

!pip install anvil-uplink
!pip install pgmpy

from google.colab import files
file=files.upload()

!unzip PGM_Project.zip

import anvil.server

anvil.server.connect("QCH763PK6D7X3PIIWQ355H5L-Q3UMHGEUAHTCYLMK")

from pgmpy.factors.discrete import TabularCPD
from pgmpy.models import BayesianNetwork

import networkx as nx
import matplotlib.pyplot as plt
from IPython.display import Image, display

"""# Setting up Model Structure
Here we define the connections between edges that the authors in the paper have worked out

### Features:
"""

from IPython.display import Image
Image('PGM_Project/Features_desc.png')

"""### BayesNet Structure"""

Image('PGM_Project/bayes_net.png')
```

```
model=BayesianNetwork([('age','trestbps'),('sex','thal'),
                       ('cp','exang'),('exang','thalach'),
                       ('slope','oldpeak'),('slope','thalach'),
                       ('ca','age'),('thal','target'),
                       ('target','ca'),('target','cp'),('target','oldpeak'),('target','slope

model.add_node('chol')

model.add_node('fbs')

model.add_node('restecg')

display(Image((nx.drawing.nx_pydot.to_pydot(model)).create_png()))

"""### CPDs for 12 variables"""

Image('PGM_Project/cpd_12.png')

"""### Add the CPDs

In pgmpy the `columns are the evidences` and `rows are the states of the variable`
"""

# 1 sex
sex_cpd=TabularCPD(variable='sex',variable_card=2,values=[[0.3232323],[0.6767677]])

# 2 cp|target
cp_cpd=TabularCPD(variable='cp',variable_card=4,values=[[0.10000000, 0.05109489],
                                                        [0.25000000, 0.06569343],
                                                        [0.40625000, 0.1313868],
                                                        [0.24375000, 0.75182482]],evidence
                                                         evidence_card=[2])

# 3 fbs
fbs_cpd=TabularCPD(variable='fbs',variable_card=2,values=[[0.8552189],[0.1447811]])

# 4 restecg
restecg_cpd=TabularCPD(variable='restecg',variable_card=3,values=[[0.49494949],[0.01346801

# 5 exang|cp
exang_cpd=TabularCPD(variable='exang',variable_card=2,values=[[ 0.82608696,0.91836735,0.86
                                                               [0.17391304,0.08163265,0.132
                                                                 evidence=['cp'],evidence_c

# 6 slope|target
slope_cpd=TabularCPD(variable='slope',variable_card=3,values=[[0.64375000, 0.26277372],
                                                              [0.30000000, 0.64963504],
                                                              [0.05625000,0.08759124]],
                                                               evidence=['target'],evidenc

# 7 ca|target
ca_cpd=TabularCPD(variable='ca',variable_card=4,values=[[0.8062500, 0.3284672],
                                                        [0.1312500, 0.3211679],
                                                        [0.0437500, 0.2262774],
```

```
                                                    [0.0187500, 0.1240876],
                                                ],evidence=['target'],evidence_ca

# 8 thal|sex
thal_cpd=TabularCPD(variable='thal',variable_card=3,values=[[0.83333333, 0.41791045],
                                                    [ 0.01041667, 0.08457711],
                                                    [ 0.15625000, 0.49751244]
                                                ],evidence=['sex'],evidence_card=

# 9 target|thal
target_cpd=TabularCPD(variable='target',variable_card=2,values=[[0.7743902, 0.3333333, 0.2
                                                        [0.2256098, 0.6666667, 0.7
                                                ],evidence=['thal'],evidence_card

#10 age|ca
age_cpd=TabularCPD(variable='age',variable_card=3,values=[[0.31034483,0.07692308,0.0263157
                                                    [0.60344828,0.75384615,0.7368421
                                                    [ 0.08620690,0.16923077,0.236842
                                                evidence=['ca'],evidence_card=[4])

# 11 trestbps|age
trestbps_cpd=TabularCPD(variable='trestbps',variable_card=3,values=[[ 0.54098361,0.2564102
                                                        [ 0.39344262,0.5179487
                                                        [ 0.06557377,0.225641
                                                evidence=['age'],evidence_card=[3]

# 12 chol
chol_cpd=TabularCPD(variable='chol',variable_card=3,values=[[0.1649832], [0.3265993], [0.5

"""### CPD for Thalach variable"""

Image('PGM_Project/cpd_thalch_C.png')

# P(thalach | slope, exang):
# +-----------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
# |     slope |           0           |           1           |
2                 |
# +-----------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
# |     exang |       0     |       1     |       0     |       1     |
0     |   1         |
# +-----------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
# | thalach=0 |   0.1504425 |  0.3461538  | 0.4133333   |  0.7580645  |     0.1666667
|  0.7777778  |
# +-----------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+
# | thalach=1 |   0.8495575 |  0.6538462  | 0.5866667   |   0.2419355 |     0.8333333
| 0.2222222   |
# +-----------+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+-----+

#thalach|slope,exang
thalach_cpd=TabularCPD(variable='thalach',variable_card=2,

                       values=[
                           [0.1504425,0.3461538,0.4133333,0.7580645,0.1666667,0.7777778],
                           [0.8495575,0.6538462,0.5866667,0.2419355,0.8333333,0.2222222]
                       ],evidence=['exang','slope'],evidence_card=[2,3])
```

```
model.get_parents('thalach')
```

"""### CPD for oldpeak variable"""

```
Image('PGM_Project/cpd_olpeak.png')
```

"""<!-- P(oldpeak | slope, target):
+---------------+---------------+---------------+---------------+---------------+---------------+---
| target | 0 |
1 |
+---------------+---------------+---------------+---------------+---------------+---------------+---
| slope | 0 | 1 | 2 | 0 | 1
| 2 |
+---------------+---------------+---------------+---------------+---------------+---------------+---
| oldpeak=0 | 0.990291262|0.958333333|0.555555556| 0.944444444 | 0.651685393
| 0.166666667 |
+---------------+---------------+---------------+---------------+---------------+---------------+---
| oldpeak=1 | 0.009708738|0.041666667|0.444444444| 0.055555556 | 0.348314607
| 0.833333333 |
+---------------+---------------+---------------+---------------+---------------+---------------+---
"""

```
#oldpeak|target,slope
oldpeak_cpd=TabularCPD(variable='oldpeak',variable_card=2,

                       values=[
                           [0.990291262,0.958333333,0.555555556,0.944444444,0.651685393,0.1
                           [0.009708738,0.041666667,0.444444444,0.055555556,0.348314607,0.8
                       ],evidence=['target','slope'],evidence_card=[2,3])

model.get_parents('oldpeak')

model.add_cpds(sex_cpd,cp_cpd,fbs_cpd,restecg_cpd,exang_cpd,slope_cpd,ca_cpd,thal_cpd,targ
               trestbps_cpd,chol_cpd,thalach_cpd,oldpeak_cpd)

model.get_parents('thalach')

model.check_model()
```

"""### `model` is now ready for inference

but let's experiment with the dataset ourselves and use structure and parameter learning o
"""

```
import pandas as pd
import numpy as np

df=pd.read_csv('PGM_Project/heart_cleveland_upload.csv')

df.rename(columns = {'condition':'target'}, inplace = True)

df
```

```
df.isna().sum() #no missing values in this dataset

"""#### We have to Discretize our dataset's 5 continous features: age, trestbps, chol, tha

def preprocessing_pipeline(age,trestbps,chol,thalach,oldpeak):
    """Only takes care of 5 continous cases, categorizing them"""
    if age>=29 and age<=45:
        age=0
    elif age>45 and age<=62:
        age=1
    elif age>62 and age<=79:
        age=2
    if trestbps>=94 and trestbps<=129:
        trestbps=0
    elif trestbps>129 and trestbps<=165:
        trestbps=1
    elif trestbps>165 and trestbps<=200:
        trestbps=2
    if chol>=126 and chol<=272:
        chol=0
    elif chol>272 and chol<=419:
        chol=1
    elif chol>419 and chol<=564:
        chol=2
    if thalach>=71 and thalach<=131:
        thalach=0
    elif thalach>131 and thalach<=202:
        thalach=1
    if oldpeak>=0.0 and oldpeak<=3.1:
        oldpeak=0
    elif oldpeak>3.1 and oldpeak<=6.2:
        oldpeak=1

    return age,trestbps,chol,thalach,oldpeak

for index,row in df.iterrows():
    df.age.iloc[index],df.trestbps.iloc[index],df.chol.iloc[index],df.thalach.iloc[index],

df.isna().sum()

df

"""### Discretization and missing values imputation is done"""

from IPython.display import Image, display
from pgmpy.estimators import HillClimbSearch, BicScore, PC, K2Score
from pgmpy.estimators import BayesianEstimator,MaximumLikelihoodEstimator,ExpectationMaxim

# Create a checkpoint
df.to_csv('PGM_Project/heart_cleveland_upload_encoded.csv')
df

#split intro training and testing sets
from sklearn.model_selection import train_test_split
```

```
X_train,X_test,y_train,y_test=train_test_split(df.drop('target',axis=1),df.target,test_siz
len(X_train),len(X_test),len(y_train),len(y_test)

X_train.shape,X_test.shape

y_train.shape,y_test.shape

type(X_train),type(y_train)#X_train,X_test are dfs while y_train,y_test are series objects

#concatenate the X_train,y_train into df_train and X_test,y_test intro df_test
df_train=pd.concat([X_train,y_train],axis=1)
df_train

df_test=pd.concat([X_test,y_test],axis=1)
df_test

"""### Structure learning

We'll perform structure learning on the whole dataset, and parameter learning with a train
And we'll test out the accuracy of the model using the test set comprising the remaining 3

"""

hc1 = HillClimbSearch(df, K2Score(df))
est_model = hc1.estimate()
display(Image((nx.drawing.nx_pydot.to_pydot(est_model)).create_png()))

est_model.get_parents('slope')

model.get_parents('slope')

model

est_model=BayesianNetwork(est_model)#convert DAG object to BN object

est_model

"""### Parameter learning on df_train"""

est_model.fit(df_train, estimator=BayesianEstimator, prior_type="BDeu")

est_model.check_model()

est_model.get_independencies()

model.get_independencies()

len(df_test)

model.get_cpds()

est_model.get_cpds()

"""### Find active trail nodes"""
```

```
#model
model.active_trail_nodes('target')

est_model.active_trail_nodes('target')

"""### Find local independencies"""

model.local_independencies('target')

est_model.local_independencies('target')

"""### Find out the accuracy of both the models"""

from pgmpy.inference import VariableElimination

def accuracy_precision_recall(df_test,model):
    target_inference=VariableElimination(model)
    true_negative,true_positive,false_negative,false_postive=0,0,0,0
    truth_values={k:v for k,v in zip(list(df_test.index),list(df_test.target))}
    df_test=df_test.drop('target',axis=1)
    for index,row in df_test.iterrows():
        cols=[col for col in df_test.columns]
        evidence_dict={k:v for k,v in zip(cols,list(df_test.loc[index][cols]))}
        prediction=target_inference.map_query(['target'],evidence=evidence_dict)
        if truth_values[index]==prediction['target']:
            if truth_values[index]==0:
                true_negative+=1
            else:
                true_positive+=1
        #FalsePositives
        elif truth_values[index]==0 and prediction['target']==1:
            false_postive+=1
        #FalseNegatives
        elif truth_values[index]==1 and prediction['target']==0:
            false_negative+=1

    return [(true_positive+true_negative)/len(df_test),true_positive/(true_positive+false_
               true_positive/(true_positive+false_negative)]#accuracy,precision,recall

accuracy_precision_recall(df_test,model)

accuracy_precision_recall(df_test,est_model)

display(Image((nx.drawing.nx_pydot.to_pydot(est_model)).create_png()))

display(Image((nx.drawing.nx_pydot.to_pydot(model)).create_png()))

df_train

"""### Structure learning on only the training set"""

hc_train_set = HillClimbSearch(df_train, K2Score(df_train))
est_model_train_set = hc_train_set.estimate()
display(Image((nx.drawing.nx_pydot.to_pydot(est_model_train_set)).create_png()))
```

```
est_model_train_set=BayesianNetwork(est_model_train_set)

"""### Parameter Learning with different estimators on the training set, accuracy on test

def accuracy_precision_recall(df_test,model):
    target_inference=VariableElimination(model)
    true_negative,true_positive,false_negative,false_postive=0,0,0,0
    truth_values={k:v for k,v in zip(list(df_test.index),list(df_test.target))}
    df_test=df_test.drop('target',axis=1)
    for index,row in df_test.iterrows():
        cols=[col for col in df_test.columns]
        evidence_dict={k:v for k,v in zip(cols,list(df_test.loc[index][cols]))}
        prediction=target_inference.map_query(['target'],evidence=evidence_dict)
        if truth_values[index]==prediction['target']:
            if truth_values[index]==0:
                true_negative+=1
            else:
                true_positive+=1
        #FalsePositives
        elif truth_values[index]==0 and prediction['target']==1:
            false_postive+=1
        #FalseNegatives
        elif truth_values[index]==1 and prediction['target']==0:
            false_negative+=1

    return [(true_positive+true_negative)/len(df_test),true_positive/(true_positive+false_
            true_positive/(true_positive+false_negative)]#accuracy,precision,recall

est_model_train_set.fit(df_train)#MLE Estimator

accuracy_precision_recall(df_test,est_model_train_set)

model.save('PGM_Project/orginal_cvd_BN.xmlbif', filetype='xmlbif')

est_model.save('PGM_Project/estimated_cvd_BN.xmlbif',filetype='xmlbif')

est_model_train_set.save('PGM_Project/estimated_train_test_cvd_BN.xmlbif',filetype='xmlbif

model = BayesianNetwork.load('PGM_Project/orginal_cvd_BN.xmlbif', filetype='xmlbif')
print(model.check_model())                    # check if CPDs are present
est_model=BayesianNetwork.load('PGM_Project/estimated_cvd_BN.xmlbif',filetype='xmlbif')
print(est_model.check_model())
est_model_train_set=BayesianNetwork.load('PGM_Project/estimated_train_test_cvd_BN.xmlbif',
print(est_model_train_set.check_model())

"""Some changes made to the code to incorporate anvil functionality:"""

def accuracy_precision_recall(df_test,model):
    target_inference=VariableElimination(model)
    true_negative,true_positive,false_negative,false_postive=0,0,0,0
    truth_values={k:v for k,v in zip(list(df_test.index),list(df_test.target))}
    df_test=df_test.drop('target',axis=1)
    for index,row in df_test.iterrows():
        cols=[col for col in df_test.columns]
        evidence_dict={k:v for k,v in zip(cols,list(df_test.loc[index][cols]))}
```

```
        prediction=target_inference.map_query(['target'],evidence=evidence_dict)
        if truth_values[index]==prediction['target']:
            if truth_values[index]==0:
                true_negative+=1
            else:
                true_positive+=1
        #FalsePositives
        elif truth_values[index]==0 and prediction['target']==1:
            false_postive+=1
        #FalseNegatives
        elif truth_values[index]==1 and prediction['target']==0:
            false_negative+=1

    return [(true_positive+true_negative)/len(df_test),true_positive/(true_positive+false_
               true_positive/(true_positive+false_negative)]#accuracy,precision,recall

def preprocessing_pipeline(age,trestbps,chol,thalach,oldpeak):
  """Only takes care of 5 continous cases, categorizing them"""

  if age>=29 and age<=45:
    age=0
  elif age>45 and age<=62:
    age=1
  elif age>62 and age<=79:
    age=2
  if trestbps>=94 and trestbps<=129:
    trestbps=0
  elif trestbps>129 and trestbps<=165:
    trestbps=1
  elif trestbps>165 and trestbps<=200:
    trestbps=2
  if chol>=126 and chol<=272:
    chol=0
  elif chol>272 and chol<=419:
    chol=1
  elif chol>419 and chol<=564:
    chol=2
  if thalach>=71 and thalach<=131:
    thalach=0
  elif thalach>131 and thalach<=202:
    thalach=1
  if oldpeak>=0.0 and oldpeak<=3.1:
    oldpeak=0
  elif oldpeak>3.1 and oldpeak<=6.2:
    oldpeak=1
  return age,trestbps,chol,thalach,oldpeak

@anvil.server.callable
def predict(age,sex,cp,trestbps,chol,fbs,restecg,thalach,exang,oldpeak,slope,ca,thal):
    age,trestbps,chol,thalach,oldpeak=preprocessing_pipeline(age,trestbps,chol,thalach,old
    target_inference=VariableElimination(est_model_train_set)
    cols=[col for col in df_test.columns if col!='target']
    evidence_dict={k:v for k,v in zip(cols,[age,sex,cp,trestbps,chol,fbs,restecg,thalach,e
    prediction=target_inference.map_query(variables=['target'],evidence=evidence_dict)
    return int(prediction['target'])
```

```
anvil.server.wait_forever()

# predict(64,1,0,170,227,0,2,155,0,0.6,1,0,2)
```

IV. Results and Inferences

The model we tried to reproduce was tested for its accuracy, precision and recall on the testing set, the results were as follows : Accuracy:83.33333333333334%, Precision:82.14285714285714%, Recall:82.14285714285714%

The Bayesian Network we trained on the training set, gave these results when tested:
Accuracy:81.66666666666667%, Precision:79.31034482758621%, Recall:82.14285714285714%

Note: The Bayesian network we fit(parameter learning) used the Bayesian Estimator as the estimator argument in the fit function call. Other estimators include Maximum Likelihood Estimator and Expectation Maximization. Both of the other alternatives gave the same performance.

V. Conclusion

In this project we used various Bayesian network methods in the pgmpy python library to reproduce the research paper in the references and to learn our own Bayesian network in order to best predict heart disease.

VI. Team Activity Learning and Work distribution

Coding, report writing, and web app creation was done by Kshitij Shekhar. Video Presentation remake was done by all three members of the group. Nihar did the mathematical analysis. Kalp experimented with the code, and performed Exploratory Data Analysis on the data set and added a visualisation of the confusion matrix.

Apart from the experience we had working as a group, we got the opportunity to deep dive into the world of Bayesian Networks. We were able to appreciate the beautiful simplicity with which Bayesian Networks can solve complex problems and still be comprehensible to the end user. It was also our first time reproducing a research paper from scratch, as the authors used a different programming language entirely to validate their findings. It was also fascinating to see how powerful scientific computing libraries have become in today's day and age, using state of the art algorithms to do things like structure and parameter learning in a matter of minutes. It was also an opportunity to appreciate LaTeX as well, and the ease with which it allows you to write beautiful mathematical equations for formal settings without much effort. It was also a step out of our comfort zone as none of us were comfortable(we still aren't, we have a lot to learn) with LaTeX. In the end we are grateful to our Professor Dr. Dhaval Patel and the TAs Kashish Shah and Kirtan Kalaria who not only were lenient enough to give the students an opportunity to do a project that fit their interests and future ambitions, but also were emphasising the need for intellectual and academic rigour while we were working on the projects.They were also instrumental in lending a helping hand when we needed it throughout the course duration.

VII. References

Muibideen, Prasad. (2020, December 18). Arxiv. A FAST ALGORITHM FOR HEART DISEASE PREDICTION USING BAYESIAN NETWORK MODEL. Retrieved November 12, 2022, from https://arxiv.org/pdf/2012.09429.p

Ankan. (2013). pgmpy 0.1.19 documentation. Pgmpy 0.1.19 Documentation. Retrieved November 12, 2022, from https://pgmpy.org/

(n.d.). Turning a Google Colab Notebook into a Web App. Anvil. Retrieved November 12, 2022, from https://anvil.works/learn/tutorials/google-colab-to-web-app