National College of Ireland

Configuration Manual of Classification models for Improving Identification of heart diseases

MSc Research Project
MSc. in Data Analytics

Omkar Terdal Ramesh
Student ID: x16104439

School of Computing
National College of Ireland

Supervisor: Dr.Catherine Mulwa

National College of Ireland
Project Submission Sheet
School of Computing

National College of Ireland

Student Name:	Omkar Terdal Ramesh
Student ID:	x16104439
Programme:	MSc. in Data Analytics
Year:	2020
Module:	MSc Research Project
Supervisor:	Dr.Catherine Mulwa
Submission Due Date:	17/08/2020
Project Title:	Configuration Manual of Classification models for Improving Identification of heart diseases
Word Count:	XXX
Page Count:	17

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.

ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature:	
Date:	17th August 2020

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST:

Attach a completed copy of this sheet to each project (including multiple copies).	☐
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☐
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☐

Assignments that are submitted to the Programme Coordinator office must be placed into the assignment box located outside the office.

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Configuration Manual of Classification models for Improving Identification of heart diseases

Omkar Terdal Ramesh
x16104439

1 Introduction

Our Project "Classification Models for Improving Identification of Heart Diseases in Healthcare Industry : Eastern Europe" is about creating model for improving identification of heart diseases using data collected from a repository and showcase its severity level using Machine learning algorithms and Artificial Neural network over a train and test dataset. This configuration manual accompanies the Project report and helps in understanding the configuration process for the duration of the project.

2 Environment configuration

We have used python through jupyter notebook to execute our project. As the data was collected in a .csv file, we stored the data onto the system as the jupyter notebook can access files directly on the system and can the software on the system itself.We are using Anaconda v2.2019.10.

2.1 Anaconda

Here are the steps to download Anaconda v2.2019.10.We can access the installer from the following link https://docs.anaconda.com/anaconda/install/hashes/win-2-64/ and we can select the version we want to download.

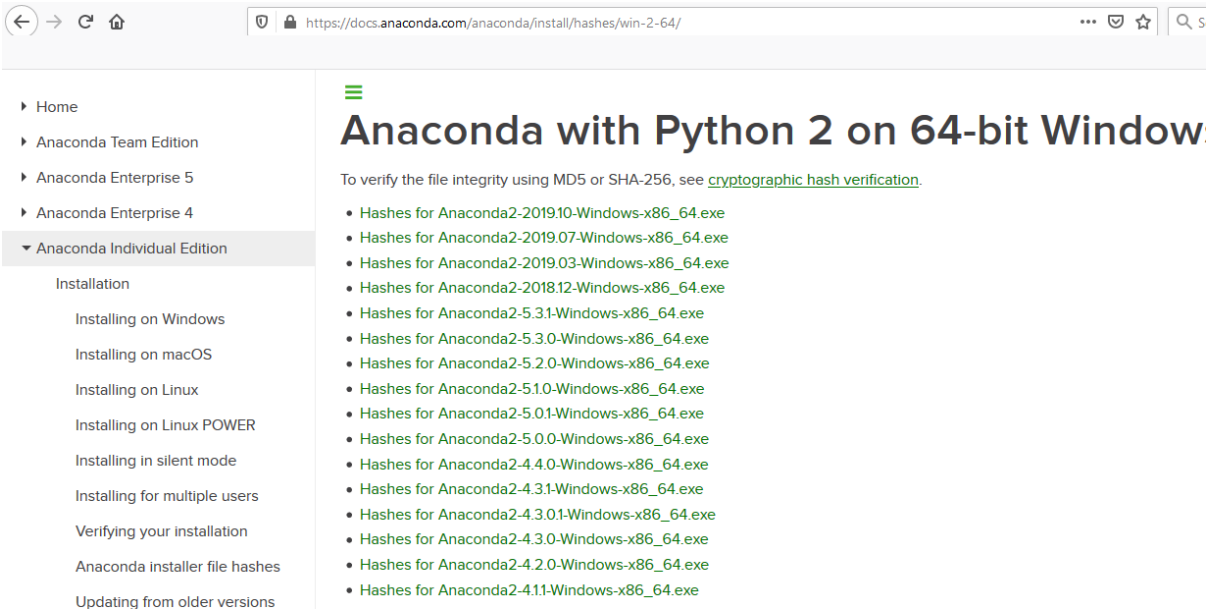


Figure 1: Anaconda Download site

2.2 Local system specifications

We used the local system to make a remote connection to the jupyter notebook. The laptop used is a Asus GL702VM and its specifications are shown below in Figure 2

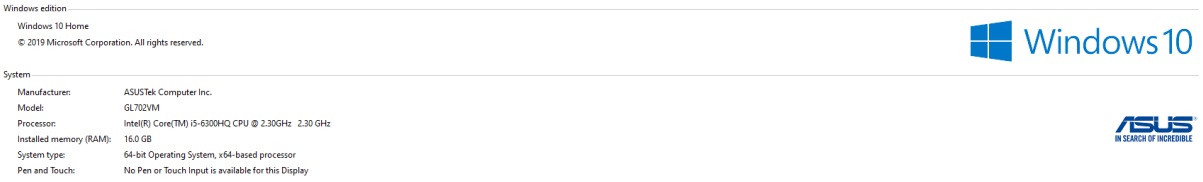


Figure 2: Local system specifications

2.3 Python Libraries

We have installed the enlisted python libraries with corresponding versions as shown below in the Table.

Libraries	Versions
numpy (numpy, 2020)	1.18.2
Pandas (Pands, 2020)	1.0.3
Matplotlib (matplotlib, 2020)	3.2.1
cv2	4.1.2
sklearn	(scikit, 2020)
1.8.0	
tensorflow (tensorflow, 2020)	2.2.0-rc3

3 Data Collection

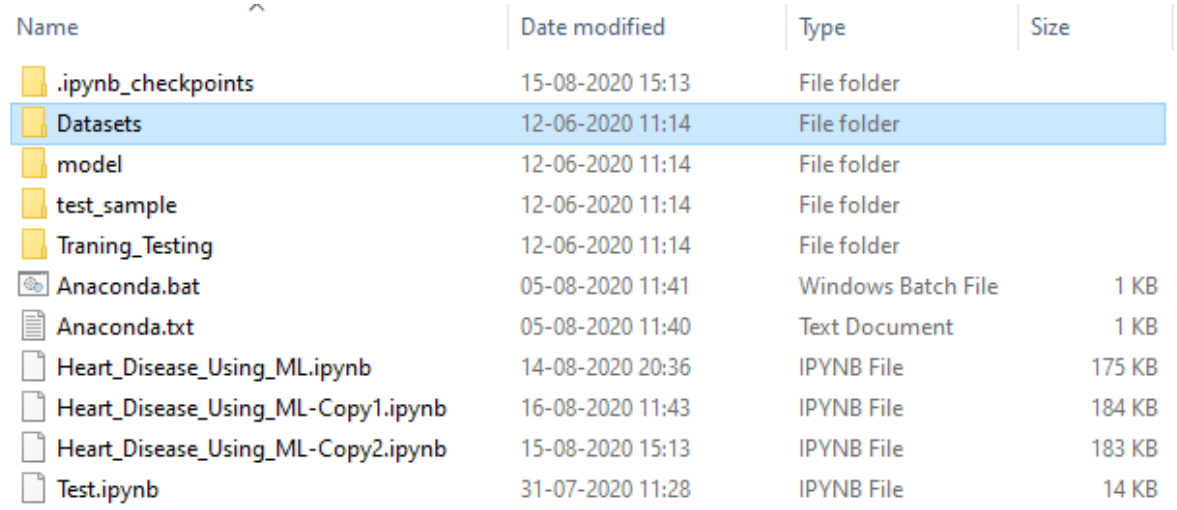
Shown Below is a part of our dataset which is stored locally on the system. The dataset has 16 attributes and over 4000 rows. They are sufficient enough to run this project. In the the below shown Figure 3 you can see the dataset.

A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
gender	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	Label
1	39	4	0	0	0	0	0	0	195	106	70	26.97	80	77	0
0	46	2	0	0	0	0	0	0	250	121	81	28.73	95	76	0
1	48	1	1	20	0	0	0	0	245	127.5	80	25.34	75	70	0
0	61	3	1	30	0	0	1	0	225	150	95	28.58	65	103	1
0	46	3	1	23	0	0	0	0	285	130	84	23.1	85	85	0
0	43	2	0	0	0	0	1	0	228	180	110	30.3	77	99	0
0	63	1	0	0	0	0	0	0	205	138	71	33.11	60	85	1
0	45	2	1	20	0	0	0	0	313	100	71	21.68	79	78	0
1	52	1	0	0	0	0	1	0	260	141.5	89	26.36	76	79	0
1	43	1	1	30	0	0	1	0	225	162	107	23.61	93	88	0
0	50	1	0	0	0	0	0	0	254	133	76	22.91	75	76	0
0	43	2	0	0	0	0	0	0	247	131	88	27.64	72	61	0
1	46	1	1	15	0	0	1	0	294	142	94	26.31	98	64	0
0	41	3	0	0	1	0	1	0	332	124	88	31.31	65	84	0
0	39	2	1	9	0	0	0	0	226	114	64	22.35	85	NA	0
0	38	2	1	20	0	0	1	0	221	140	90	21.35	95	70	1
1	48	3	1	10	0	0	1	0	232	138	90	22.37	64	72	0
0	46	2	1	20	0	0	0	0	291	112	78	23.38	80	89	1
0	38	2	1	5	0	0	0	0	195	122	84.5	23.24	75	78	0
1	41	2	0	0	0	0	0	0	195	139	88	26.88	85	65	0
0	42	2	1	30	0	0	0	0	190	108	70.5	21.59	72	85	0
0	43	1	0	0	0	0	0	0	185	123.5	77.5	29.89	70	NA	0
0	52	1	0	0	0	0	0	0	234	148	78	34.17	70	113	0
0	52	3	1	20	0	0	0	0	215	132	82	25.11	71	75	0
1	44	2	1	30	0	0	1	0	270	137.5	90	21.96	75	83	0
1	47	4	1	20	0	0	0	0	294	102	68	24.18	62	66	1
0	60	1	0	0	0	0	0	0	260	110	72.5	26.59	65	NA	0
1	35	2	1	20	0	0	1	0	225	132	91	26.09	73	83	0
0	61	3	0	0	0	0	1	0	272	182	121	32.8	85	65	1
0	60	1	0	0	0	0	0	0	247	130	88	30.36	72	74	0
1	36	4	1	35	0	0	0	0	295	102	68	28.15	60	63	0
1	43	4	1	43	0	0	0	0	226	115	85.5	27.57	75	75	0
0	59	1	0	0	0	0	1	0	209	150	85	20.77	90	88	1
1	61	NA	1	5	0	0	0	0	175	134	82.5	18.59	72	75	1
1	54	1	1	20	0	0	1	0	214	147	74	24.71	96	87	0
1	37	2	0	0	0	0	1	0	225	124.5	92.5	38.53	95	83	0
1	56	NA	0	0	0	0	0	0	257	153.5	102	28.09	72	75	0

Figure 3: Dataset and Attributes

3.1 Data Storage

Shown in Figure 4 below is the path where the collected data is stored for the project. The data is stored in the datasets folder to access the data through the Jupyter notebook. This the installation folder for Anaconda.



Name	Date modified	Type	Size
.ipynb_checkpoints	15-08-2020 15:13	File folder	
Datasets	12-06-2020 11:14	File folder	
model	12-06-2020 11:14	File folder	
test_sample	12-06-2020 11:14	File folder	
Traning_Testing	12-06-2020 11:14	File folder	
Anaconda.bat	05-08-2020 11:41	Windows Batch File	1 KB
Anaconda.txt	05-08-2020 11:40	Text Document	1 KB
Heart_Disease_Using_ML.ipynb	14-08-2020 20:36	IPYNB File	175 KB
Heart_Disease_Using_ML-Copy1.ipynb	16-08-2020 11:43	IPYNB File	184 KB
Heart_Disease_Using_ML-Copy2.ipynb	15-08-2020 15:13	IPYNB File	183 KB
Test.ipynb	31-07-2020 11:28	IPYNB File	14 KB

Figure 4: Dataset and Attributes

4 Jupyter notebook

In this section we will run jupyter notebook with the help of anaconda. The below Figure shows that in order to run jupyter notebook, we need to first launch the command prompt from the same directory. And the run the anaconda batch file to load the required libraries into jupyter notebook.

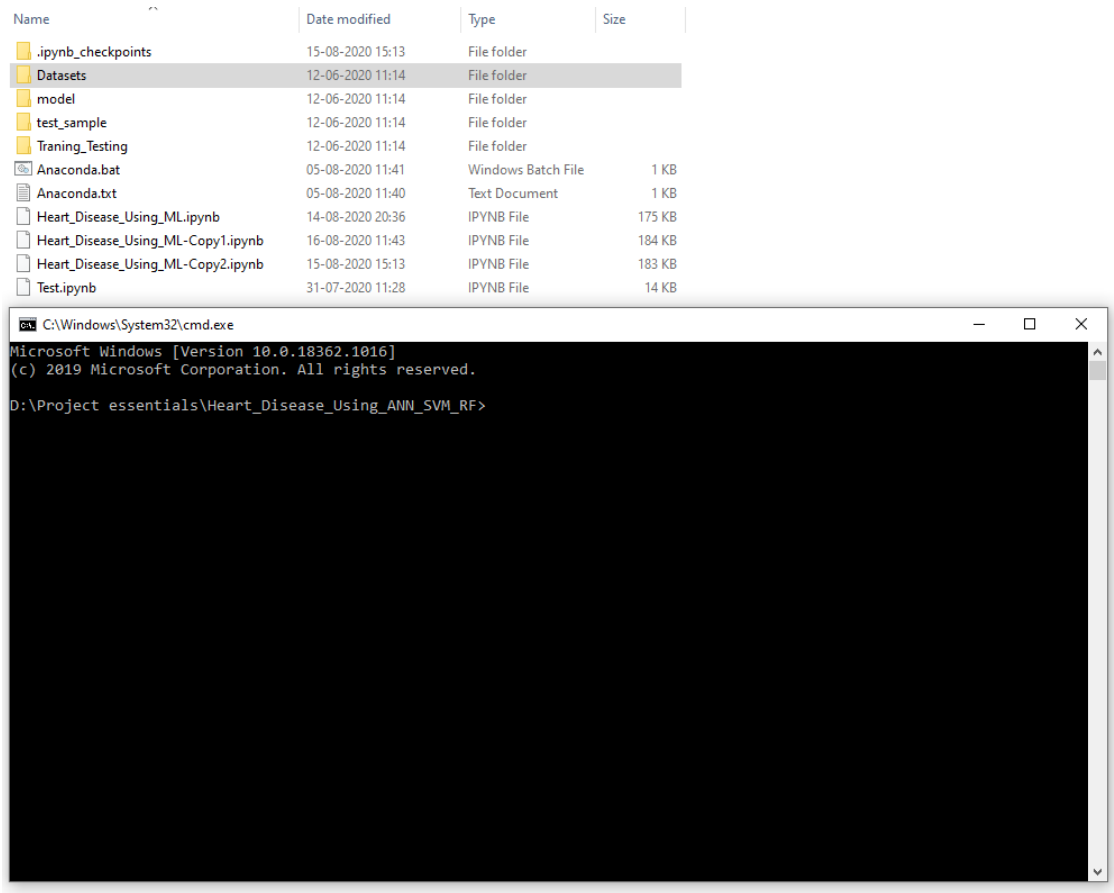


Figure 5: Command prompt

We then launch jupyter notebook after the libraries have been loaded.

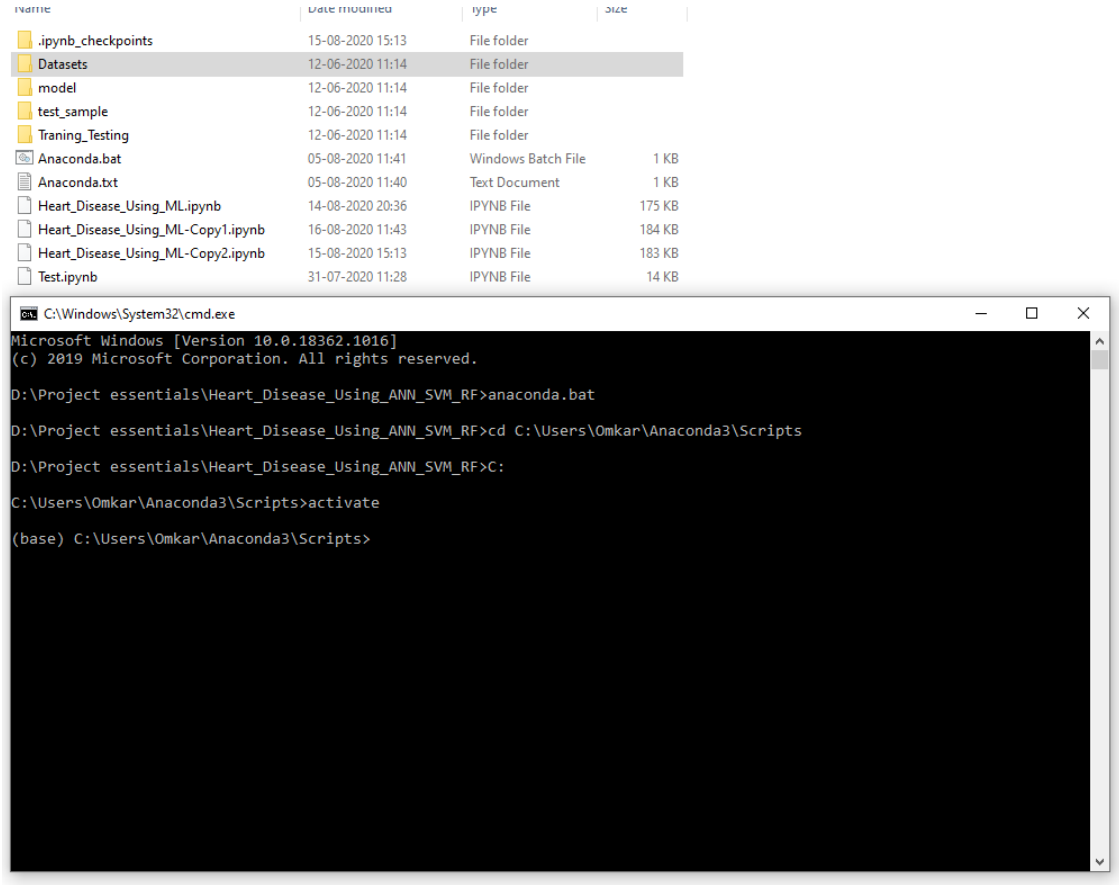


Figure 6: Launching Jupyter notebook

After the Jupyter notebook is lauched it establishes a connection to the directory and opens in the default web browser with the files from the directory.

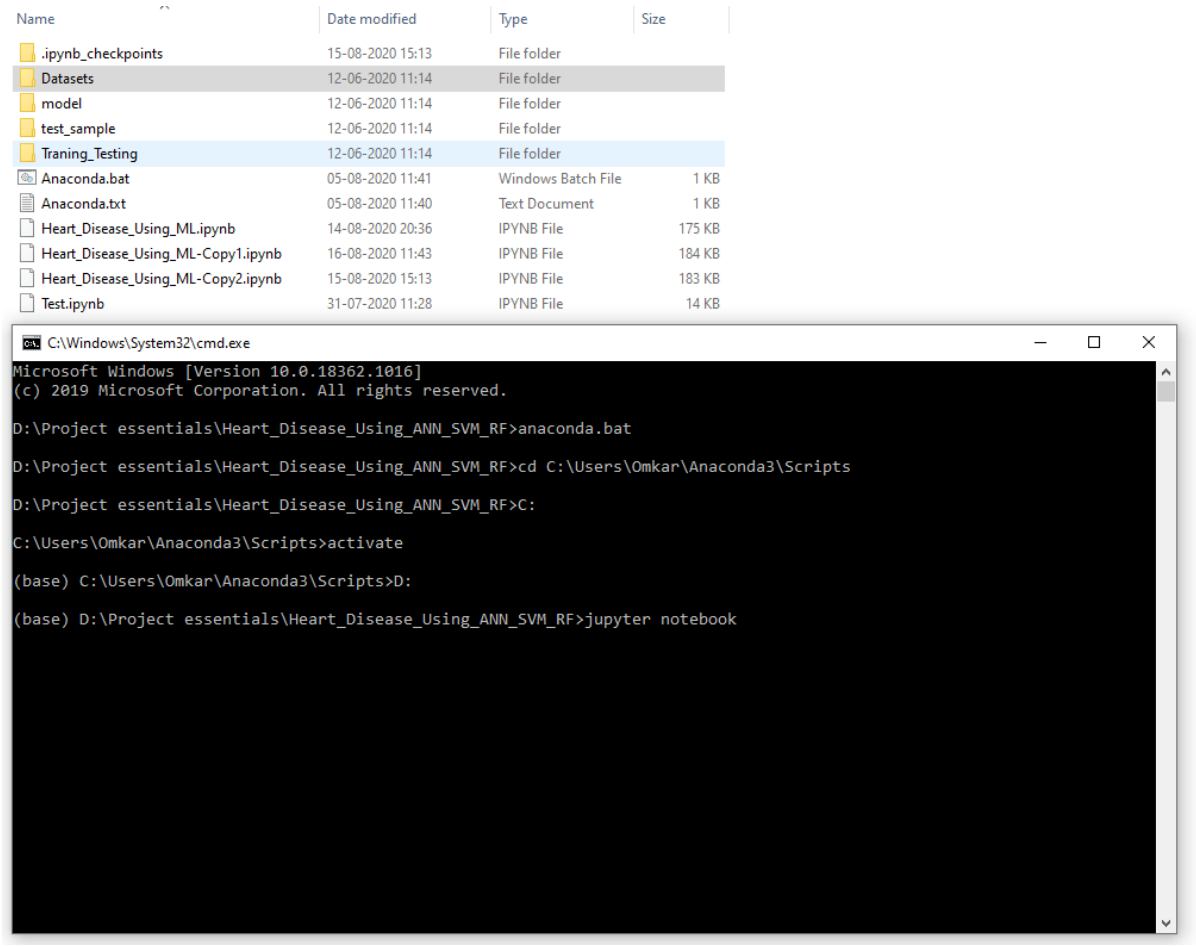


Figure 7: Loading Jupyter notebook

5 Implementation

We go ahead and import all the necessary libraries required for our project. The Figure 9 below shows the libraries imported.

We go ahead and load the data into the Jupyter notebook by using the read command. The Figure 10 shows how.

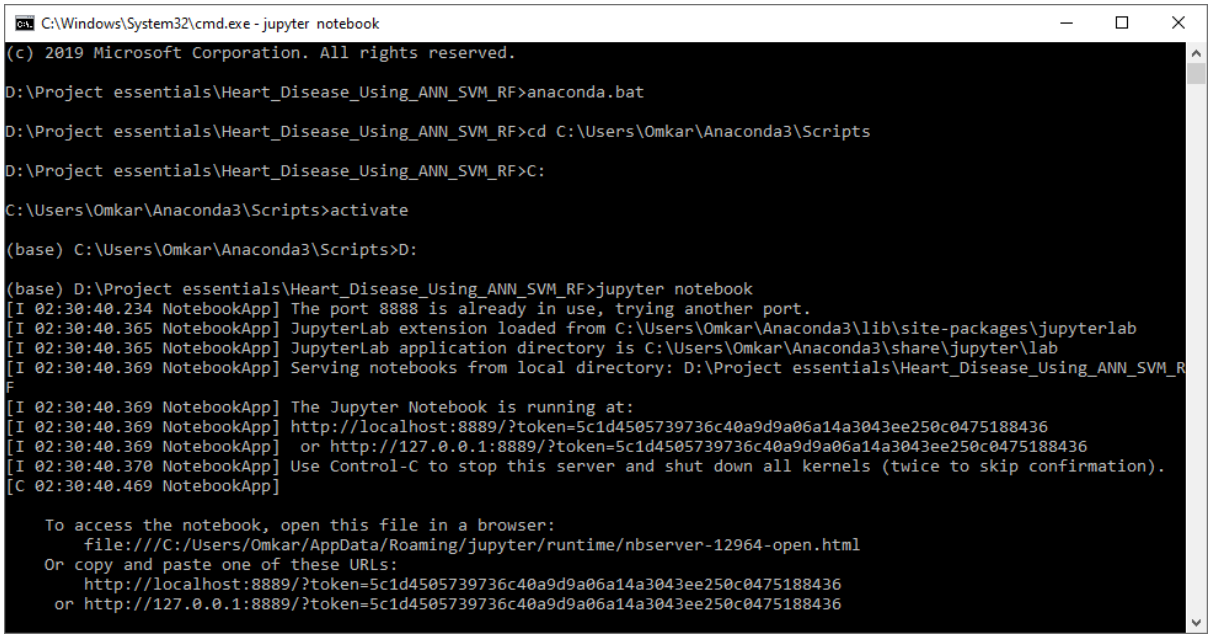


Figure 8: Jupyter Notebook

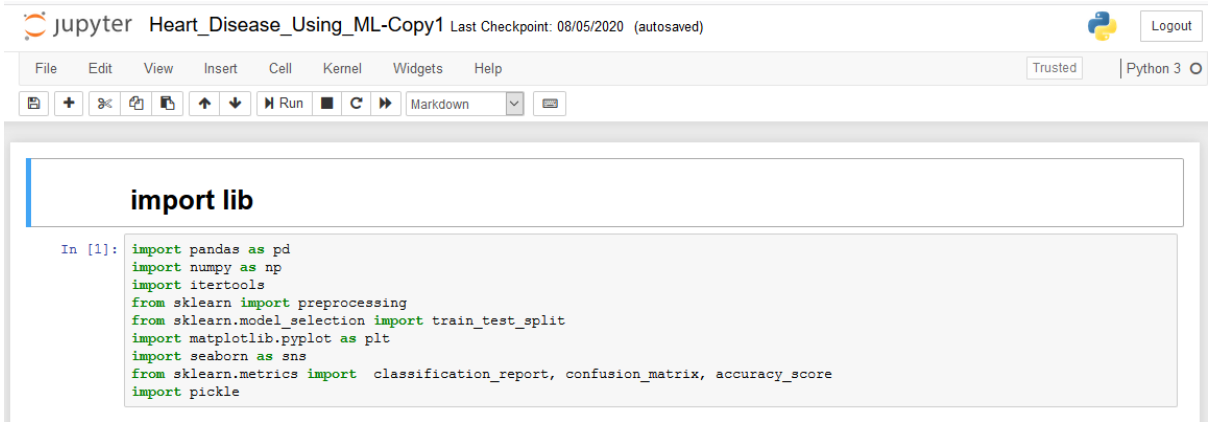


Figure 9: Jupyter Libraries

5.1 Implementation of ANN

We will first divide the data into training data and test data. This is ensure that the models can perform and hope to achieve the maximum efficiency.Figure 11

5.1.1 Data preprocessing

We preprocess the data as required by the ANN model by using keras and converting the data into arrays of data. The following figure 12 demonstrates it.

5.1.2 Building and training ANN

We will build a sequential model, compile the model and Fit the model to run through epochs. This model runs successfully through a 100 epochs and yields a result of 85.61 accuracy.

load data

```
In [2]: df = pd.read_csv('Datasets/Heart_Data.csv')

In [3]: df.head()
```

Out[3]:

	gender	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose	La
0	1	39	4.0	0	0.0	0.0	0	0	0	195.0	106.0	70.0	26.97	80.0	77.0	
1	0	46	2.0	0	0.0	0.0	0	0	0	250.0	121.0	81.0	28.73	95.0	76.0	
2	1	48	1.0	1	20.0	0.0	0	0	0	245.0	127.5	80.0	25.34	75.0	70.0	
3	0	61	3.0	1	30.0	0.0	0	1	0	225.0	150.0	95.0	28.58	65.0	103.0	
4	0	46	3.0	1	23.0	0.0	0	0	0	285.0	130.0	84.0	23.10	85.0	85.0	

```
In [4]: df.columns

Out[4]: Index(['gender', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
               'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
               'diaBP', 'BMI', 'heartRate', 'glucose', 'Label'],
              dtype='object')
```

Figure 10: Loading Data into Jupyter notebook

ANN

```
In [5]: df.dropna(axis=0,inplace=True)

In [6]: dataset = df.values

In [7]: dataset

Out[7]: array([[  1.,  39.,   4., ...,  80.,  77.,   0.],
               [  0.,  46.,   2., ...,  95.,  76.,   0.],
               [  1.,  48.,   1., ...,  75.,  70.,   0.],
               ...,
               [  0.,  52.,   2., ...,  80., 107.,   0.],
               [  1.,  40.,   3., ...,  67.,  72.,   0.],
               [  0.,  39.,   3., ...,  85.,  80.,   0.]])

In [8]: X = dataset[:,0:15]
        print(X)
        Y = dataset[:,15]

[[  1.    39.     4.   ...  26.97  80.    77.  ]
 [  0.    46.     2.   ...  28.73  95.    76.  ]
 [  1.    48.     1.   ...  25.34  75.    70.  ]
 ...
 [  0.    52.     2.   ...  21.47  80.   107.  ]
 [  1.    40.     3.   ...  25.6   67.    72.  ]
 [  0.    39.     3.   ...  20.91  85.    80.  ]]
```

Figure 11: Data Distribution to training and test data

5.1.3 Visualization

The Following plot graph shows the loss during epochs (Figure 15). And followed by that we can see the Accuracy graph of the model displayed with the help of the plot command (Figure 16)

5.1.4 Conclusion

The following figure 17 shows the accuracy, precision, recall, f1-score and support of the ANN model.

preprocessing

```
In [9]: from keras.models import Sequential
        from keras.layers import Dense, InputLayer, Flatten, BatchNormalization

        Using TensorFlow backend.

In [10]: min_max_scaler = preprocessing.MinMaxScaler()
         X_scale = min_max_scaler.fit_transform(X)

In [11]: X_scale

Out[11]: array([[1.        , 0.18421053, 1.        , ..., 0.27702375, 0.36363636,
                 0.10451977],
                [0.        , 0.36842105, 0.33333333, ..., 0.31968008, 0.51515152,
                 0.10169492],
                [1.        , 0.42105263, 0.        , ..., 0.23751818, 0.31313131,
                 0.08474576],
                ...,
                [0.        , 0.52631579, 0.33333333, ..., 0.14372273, 0.36363636,
                 0.18926554],
                [1.        , 0.21052632, 0.66666667, ..., 0.24381968, 0.23232323,
                 0.09039548],
                [0.        , 0.18421053, 0.66666667, ..., 0.13015027, 0.41414141,
                 0.11299435]])

In [12]: X_train, X_val_and_test, Y_train, Y_val_and_test = train_test_split(X_scale, Y, test_size=0.3)

         X_val, X_test, Y_val, Y_test = train_test_split(X_val_and_test, Y_val_and_test, test_size=0.5)

         print(X_train.shape, X_val.shape, X_test.shape, Y_train.shape, Y_val.shape, Y_test.shape)

         (2560, 15) (549, 15) (549, 15) (2560,) (549,) (549,)
```

Figure 12: Data preprocessing for ANN

5.2 Implementation of Random Forest and SVM Models

5.2.1 Data preprocessing

We need to preprocess the data again in a different way to feed it to both the random forest and SVM models. These are machine learing algorithms and hence need to be preprocessed differently compared to Artificial Neural networks. Figure 18

We go ahead and plot the attributes of the dataset. This provides and idea on how to approach the Models for increased performance and accuracy. Figure 19

We consider the gender attribute to be labelled and plotted to see the number of entries they have and how we can make use of it. Figure 20

We then plot the gender compared to age to see calculate the age group for the most affected are. This figure 21 shows us that the most case received are between the ages of 39-48.

Keeping age groups and gender as our two main attributes for the models we will further train the data into training set and test set. As shown in the Figure 22.

5.2.2 Random Forest Model

We train the random forest model and input the training and test data. As shown in Figure 23.The Random Forest model yielded an accuracy of 85.06. With the precision being 0.85, recall being 1.00 and f1-score being 0.92. as shown in figure 24.

Building and Training Our Artifcial Neural Network

```
In [13]: model = Sequential([
             Dense(32, activation='relu', input_shape=(15,)),
             Dense(32, activation='relu'),
             Dense(1, activation='sigmoid'),
         ])
```

```
WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:74: The name tf.get_
default_graph is deprecated. Please use tf.compat.v1.get_default_graph instead.

WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:517: The name tf.pla
ceholder is deprecated. Please use tf.compat.v1.placeholder instead.

WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:4138: The name tf.ra
ndom_uniform is deprecated. Please use tf.random.uniform instead.
```

```
In [14]: model.compile(optimizer='sgd',
                       loss='binary_crossentropy',
                       metrics=['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\optimizers.py:790: The name tf.train.Optimizer is
deprecated. Please use tf.compat.v1.train.Optimizer instead.

WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:3376: The name tf.lo
g is deprecated. Please use tf.math.log instead.

WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\tensorflow\python\ops\nn_impl.py:180: add_dispatch_suppo
rt.<locals>.wrapper (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

Figure 13: Training ANN

5.2.3 SVM Model

We train and fit the SVM model with the training and test data. The following Figure 25 shows how. The SVM model yielded an accuracy of 84.79. The precision 0.85, recall is 1.0, and f1-score is 0.92. As shown in Figure 26.

6 Comparison

Once we aquired all the accuracy from the models we ran, we felt the need to compare the results just to see how close or how far apart the artificial neural network techniques is from the machine learning techniques. The highest accuracy was yielded by ANN. Though the ANN technique's accuracy was a bit higher than other techniques, we can conclude that in the case of disease identification every small amount is considered and can come to great help. The following plot will show the differnce in the accuracies. Figure 27

7 Section 6

Your sixth section. Change the header and label to something appropriate.

References

```
In [15]: hist = model.fit(X_train, Y_train,
                 batch_size=32, epochs=100,
                 validation_data=(X_val, Y_val))
```

```
WARNING:tensorflow:From C:\Users\Omkar\Anaconda3\lib\site-packages\keras\backend\tensorflow_backend.py:986: The name tf.a
ssign_add is deprecated. Please use tf.compat.v1.assign_add instead.

Train on 2560 samples, validate on 549 samples
Epoch 1/100
2560/2560 [==============================] - 0s 190us/step - loss: 0.6465 - acc: 0.6555 - val_loss: 0.5266 - val_acc: 0.8
689
Epoch 2/100
2560/2560 [==============================] - 0s 35us/step - loss: 0.5035 - acc: 0.8422 - val_loss: 0.4504 - val_acc: 0.86
89
Epoch 3/100
2560/2560 [==============================] - 0s 34us/step - loss: 0.4656 - acc: 0.8422 - val_loss: 0.4248 - val_acc: 0.86
89
Epoch 4/100
2560/2560 [==============================] - 0s 35us/step - loss: 0.4531 - acc: 0.8422 - val_loss: 0.4145 - val_acc: 0.86
89
Epoch 5/100
2560/2560 [==============================] - 0s 34us/step - loss: 0.4475 - acc: 0.8422 - val_loss: 0.4095 - val_acc: 0.86
89
```

```
In [16]: model.evaluate(X_test, Y_test)[1]
```

```
549/549 [==============================] - 0s 60us/step
```

```
Out[16]: 0.8561020037515568
```

Figure 14: Building ANN

Visualizing Loss and Accuracy

```
In [17]: plt.plot(hist.history['loss'])
         plt.plot(hist.history['val_loss'])
         plt.title('Model loss')
         plt.ylabel('Loss')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Val'], loc='upper right')
         plt.show()
```

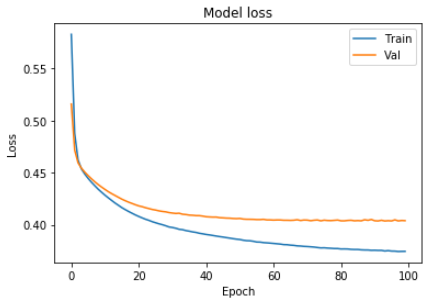


Figure 15: Loss in ANN

```
In [18]: plt.plot(hist.history['acc'])
         plt.plot(hist.history['val_acc'])
         plt.title('Model accuracy')
         plt.ylabel('Accuracy')
         plt.xlabel('Epoch')
         plt.legend(['Train', 'Val'], loc='lower right')
         plt.show()
```

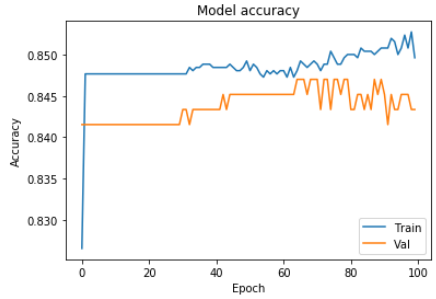


Figure 16: Accuracy for ANN

```
In [22]: Y_pred = model.predict(X_test)
         Y_pred = np.round(Y_pred.flatten())

In [23]: accuracy = accuracy_score(Y_test, Y_pred)
         print("Accuracy: %.2f%%" % (accuracy*100))

         Accuracy: 85.25%

In [24]: print(classification_report(Y_test, Y_pred))

                      precision    recall  f1-score   support

                 0.0       0.85      1.00      0.92       468
                 1.0       0.50      0.01      0.02        81

            accuracy                           0.85       549
           macro avg       0.68      0.51      0.47       549
        weighted avg       0.80      0.85      0.79       549
```

Figure 17: Conclusion for ANN

Data preprocessing and Data Analysis for ML

```
In [22]: df = pd.read_csv('Datasets/Heart_Data.csv')

In [23]: df=df.dropna(axis=0)

In [24]: from numpy import set_printoptions
         from sklearn.feature_selection import SelectKBest
         from sklearn.feature_selection import f_classif

         X = df.drop(['Label'], axis=1)
         y = df['Label']
         # feature extraction
         test = SelectKBest(score_func=f_classif, k=4)
         fit = test.fit(X, y)
         # summarize scores
         set_printoptions(precision=3)
         print(fit.scores_)
         features = fit.transform(X)
         # summarize selected features
         print(features[0:5,:])

         [ 30.996 211.751  14.698   1.343   9.918  29.291   8.572 124.379  32.195
           30.757 191.001  84.296  24.783   1.539  55.229]
         [[ 39.    0.  106.   70. ]
          [ 46.    0.  121.   81. ]
          [ 48.    0.  127.5  80. ]
          [ 61.    1.  150.   95. ]
          [ 46.    0.  130.   84. ]]
```

Figure 18: Data preprocessing for ML Methods

```
In [28]: df.columns

Out[28]: Index(['gender', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose', 'Label'],
      dtype='object')

In [29]: name = ['gender', 'age', 'education', 'currentSmoker', 'cigsPerDay', 'BPMeds',
       'prevalentStroke', 'prevalentHyp', 'diabetes', 'totChol', 'sysBP',
       'diaBP', 'BMI', 'heartRate', 'glucose']
         importance = fit.scores_

In [30]: plt.figure(figsize=(40,15), dpi=100)
         sns.barplot(x=name, y=importance)

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x23490238508>
```

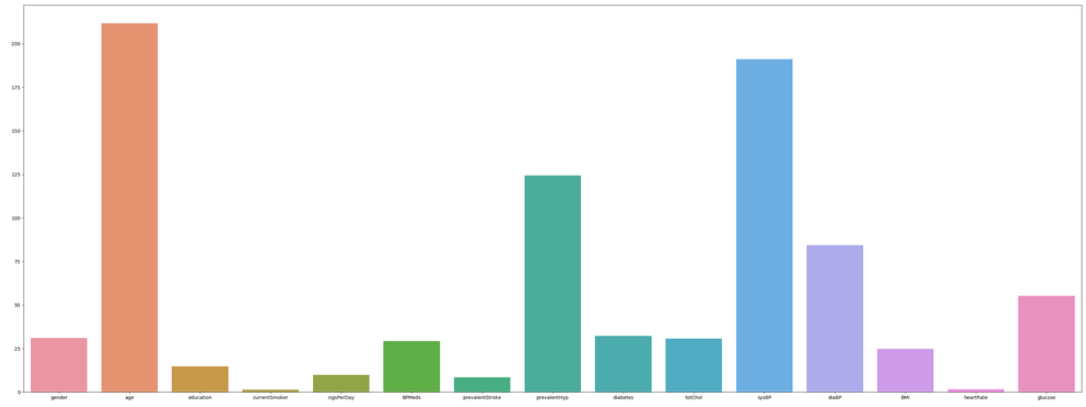


Figure 19: Plotting Attributes

```
In [28]: plt.figure(figsize=(15,7), dpi=100)
         sns.countplot(x='gender',hue='Label',data=df, palette='Set1')

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x14b70d191c8>
```

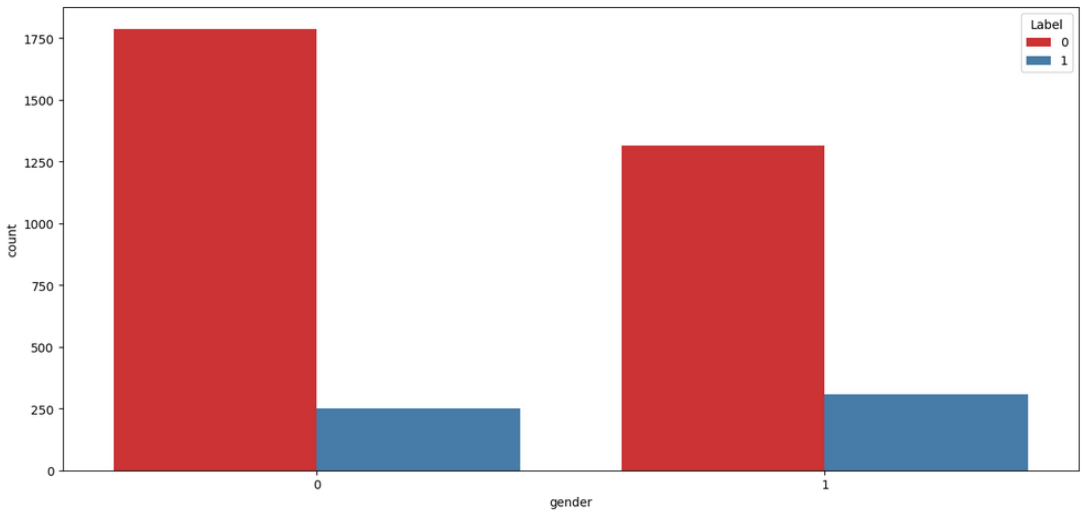


Figure 20: Gender plot

```
In [29]: plt.figure(figsize=(15,7), dpi=100)
         sns.countplot(x='age',hue='Label',data=df, palette='Set1')
Out[29]: <matplotlib.axes._subplots.AxesSubplot at 0x14b70e12248>
```

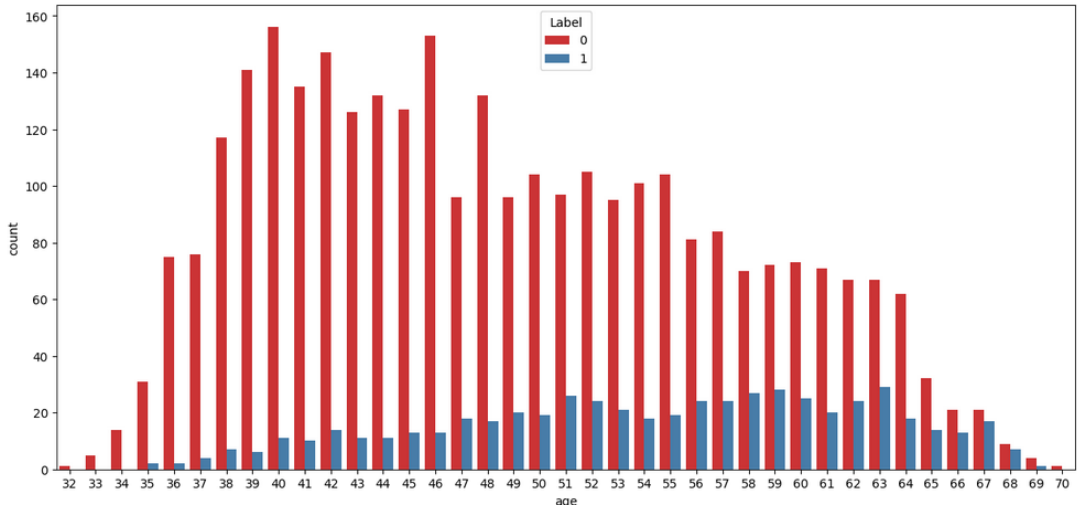


Figure 21: Age group plot

```
In [36]: X = df.drop('Label', axis=1)
         y = df['Label']
         x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=1031)
         x_train.head(10)
```

Out[36]:

	gender	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp	diabetes	totChol	sysBP	diaBP	BMI	heartRate	glucose
1111	0	52	2.0	0	0.0	0.0	0	1	1	600.0	159.5	94.0	28.27	78.0	140.0
3938	0	49	3.0	1	4.0	0.0	0	1	0	227.0	150.0	91.0	24.30	88.0	83.0
2469	1	48	4.0	1	20.0	0.0	0	1	0	259.0	135.0	90.0	20.72	102.0	81.0
221	0	38	2.0	1	9.0	0.0	0	0	0	180.0	124.0	66.0	29.29	85.0	68.0
2916	0	45	1.0	0	0.0	0.0	0	0	0	210.0	120.0	72.0	22.01	75.0	93.0
2416	1	56	2.0	1	20.0	0.0	0	1	0	205.0	210.0	130.0	25.49	95.0	127.0
2225	1	38	4.0	0	0.0	0.0	0	0	0	240.0	122.5	80.0	23.97	60.0	43.0
875	1	39	2.0	1	30.0	0.0	0	0	0	225.0	128.0	86.5	25.13	74.0	100.0
3042	0	38	4.0	1	3.0	0.0	0	0	0	177.0	126.0	80.0	23.84	90.0	79.0
1653	1	39	3.0	1	20.0	0.0	0	0	0	148.0	101.0	62.0	24.47	70.0	81.0

```
In [37]: print(x_train.shape, x_test.shape, y_train.shape, y_test.shape)
         (2560, 15) (1098, 15) (2560,) (1098,)

In [38]: x_train.to_excel('Traning_Testing/X_train.xlsx', index=False)
         x_test.to_excel('Traning_Testing/X_test.xlsx', index=False)
         y_train.to_excel('Traning_Testing/y_train.xlsx', index=False)
         y_test.to_excel('Traning_Testing/y_test.xlsx', index=False)
```

Figure 22: Training the Data

Training the Random Forest model

```
In [34]: from sklearn.ensemble import RandomForestClassifier

In [35]: rfc = RandomForestClassifier(n_estimators=600)

In [36]: rfc = rfc.fit(x_train,y_train)
```

Figure 23: Training Random Forest

Predictions and Evaluation of RF Model

```
In [42]: predictions_rf = rfc.predict(x_test)

In [43]: acc_rf = accuracy_score(y_true=y_test, y_pred= predictions_rf)
         print("Overall accuracy of ADA model using test-set is : %f" %(acc_rf*100))

         Overall accuracy of ADA model using test-set is : 85.063752

In [44]: print(classification_report(y_test,predictions_rf))

                      precision    recall  f1-score   support

                   0       0.85      1.00      0.92       931
                   1       0.64      0.04      0.08       167

            accuracy                           0.85      1098
           macro avg       0.74      0.52      0.50      1098
        weighted avg       0.82      0.85      0.79      1098


In [45]: print(confusion_matrix(y_test,predictions_rf))

         [[927   4]
          [160   7]]

In [46]: filename='model/rf_model.sav'
         pickle.dump(rfc,open(filename, 'wb'))
```

Figure 24: Evaluation of RF

Training the SVM model

```
In [47]: from sklearn.svm import SVC

In [48]: model_svm = SVC()

In [49]: model_svm.fit(x_train, y_train)

         C:\Users\Omkar\Anaconda3\lib\site-packages\sklearn\svm\base.py:193: FutureWarning: The default value of gamma will change f
         rom 'auto' to 'scale' in version 0.22 to account better for unscaled features. Set gamma explicitly to 'auto' or 'scale' to
         avoid this warning.
           "avoid this warning.", FutureWarning)

Out[49]: SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
             decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
             kernel='rbf', max_iter=-1, probability=False, random_state=None,
             shrinking=True, tol=0.001, verbose=False)
```

Figure 25: Training SVM Model

Predictions and Evaluation

```
In [45]: predictions_svm = model_svm.predict(x_test)

In [46]: acc_svm = accuracy_score(y_true=y_test, y_pred= predictions_svm)
         print("Overall accuracy of SVM model using test-set is : %f" %(acc_svm*100))

         Overall accuracy of SVM model using test-set is : 84.790528

In [48]: print(classification_report(y_test,predictions_svm))

                      precision    recall  f1-score   support

                   0       0.85      1.00      0.92       931
                   1       0.00      0.00      0.00       167

            accuracy                           0.85      1098
           macro avg       0.42      0.50      0.46      1098
        weighted avg       0.72      0.85      0.78      1098


In [48]: print(confusion_matrix(y_test,predictions_svm))

         [[931   0]
          [167   0]]

In [49]: filename='model/svm_model.sav'
         pickle.dump(model_svm,open(filename, 'wb'))
```

Figure 26: Evaluation of SVM

Comparation

In [50]:
```
li_x = [ 'ANN', 'RF', 'SVM']
li_y = [ accuracy, acc_rf,acc_svm]
```

In [51]:
```
li = []
for i in li_y:
    print(i)
    li.append(i*100)
```

```
0.8597449908925319
0.8479052823315119
0.8479052823315119
```

In [52]:
```
import seaborn as sns
print(li)
sns.barplot(x=li_x, y=li)
```

```
[85.97449908925319, 84.79052823315118, 84.79052823315118]
```

Out[52]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x15502c0d408>
```

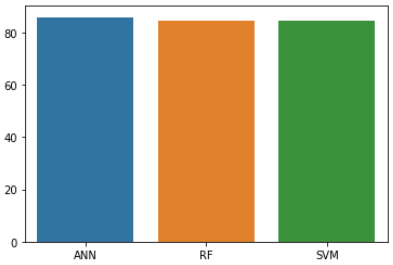


Figure 27: Comparison of ANN, RF and SVM