CSE 310:
AI & ML

PROJECT DETAILS

Submitted to

SRI RAMACHANDRA INSTITUTE OF HIGHER EDUCATION AND RESEARCH SRI RAMACHANDRA ENGINEERING AND TECHNOLOGY

for the Award of the Degree of

BACHELOR OF TECHNOLOGY

in

COMPUTER SCIENCE AND ENGINEERING

(Cyber Security and Internet of Things) By

K.P. SHRIRAM - E0219007

M POOJITH – E021056

UMESH KUMAR - E0219019

Under the Supervision of

Prof. Ragavan Veerarajan

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRET, CHENNAI - 600116 OCTOBER 2021

TOPIC **:** AI HEALTH CARE BOT AND HEART FAILURE MACHINE LEARNING MODEL

ABSTRACT:

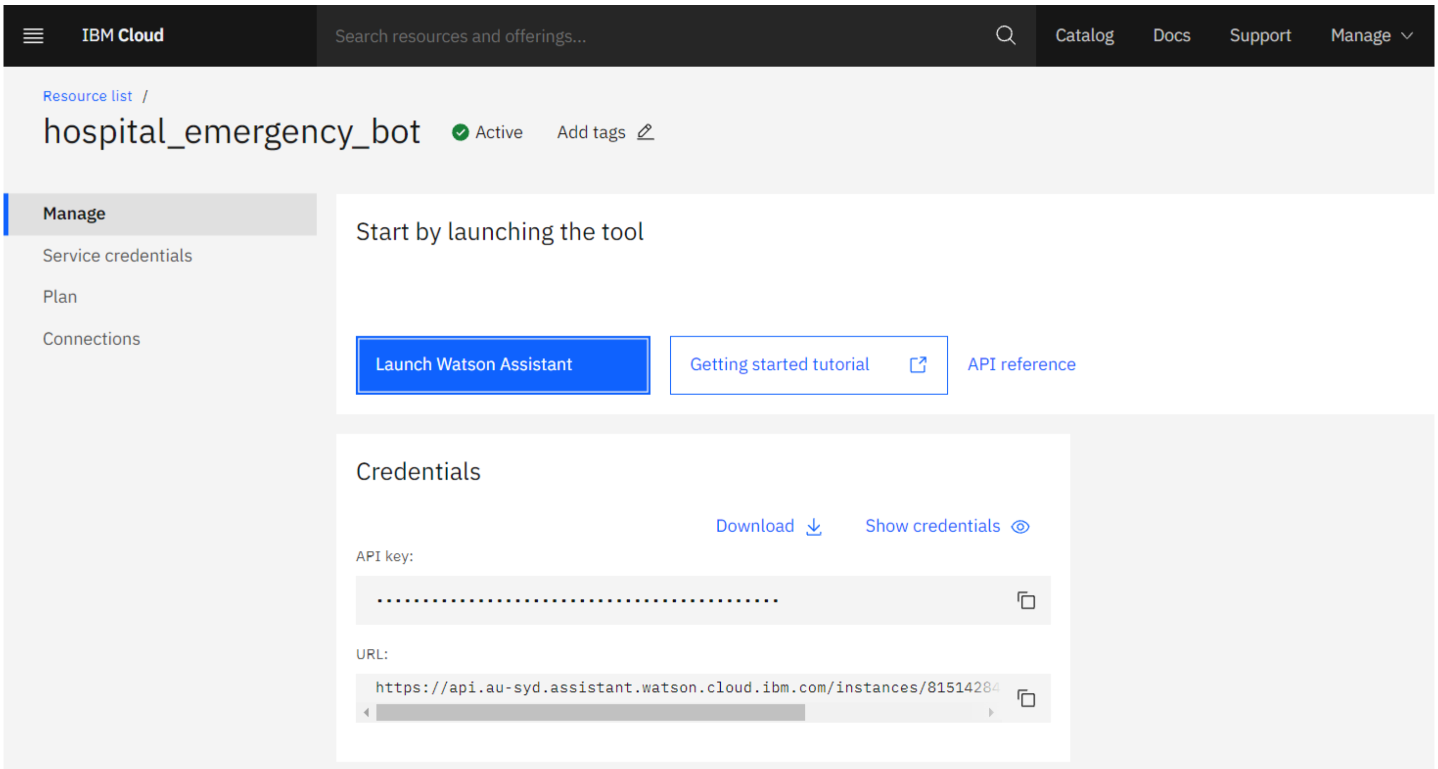
In IBM Cloud using Watson Assistant service , Simple chat box is created for Sri Ramachandra Hospital . And Using Heart_failure.csv dataset, we have preprocessed, analysed and predicted the data using different machine learning models and have acquired better model for our dataset with minimal RMSE Errors .

AIM OF THE PROJECT:

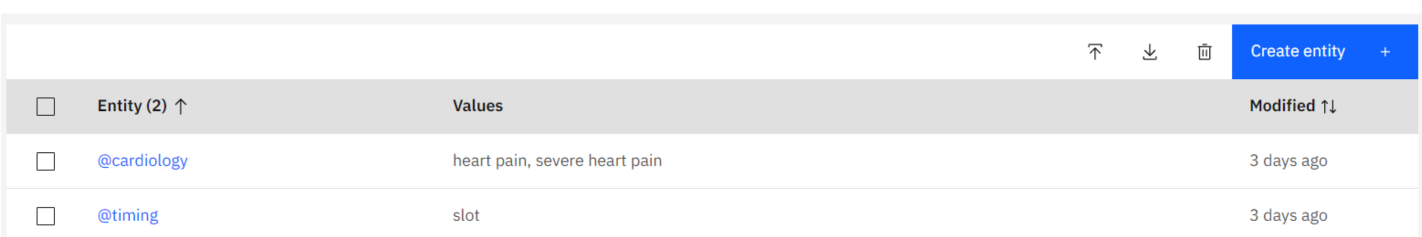
To Create Simple AI Chat bot Sri Ramachandra Hospital and Training a Machine learning model with heart Failure dataset .

AI CHAT BOT :

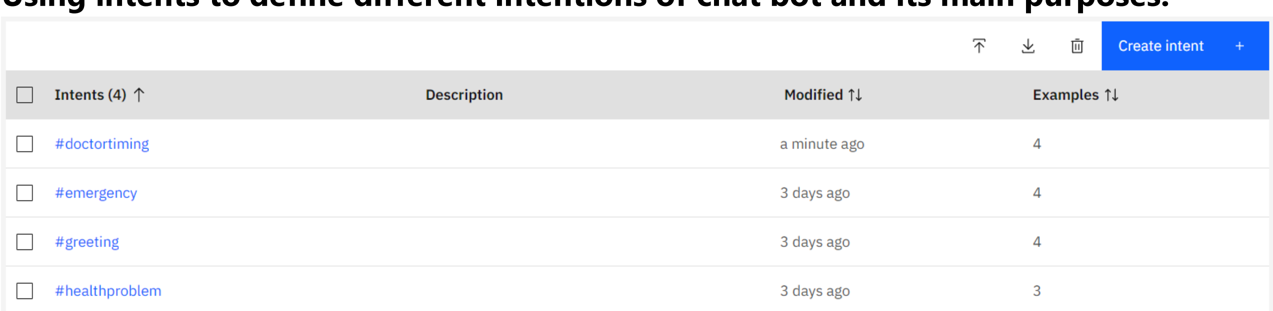
Simple ai chat bot is created using ibm cloud as interface and watson assistant as main platform .



Then using Entities to define charateristics of chat bot.

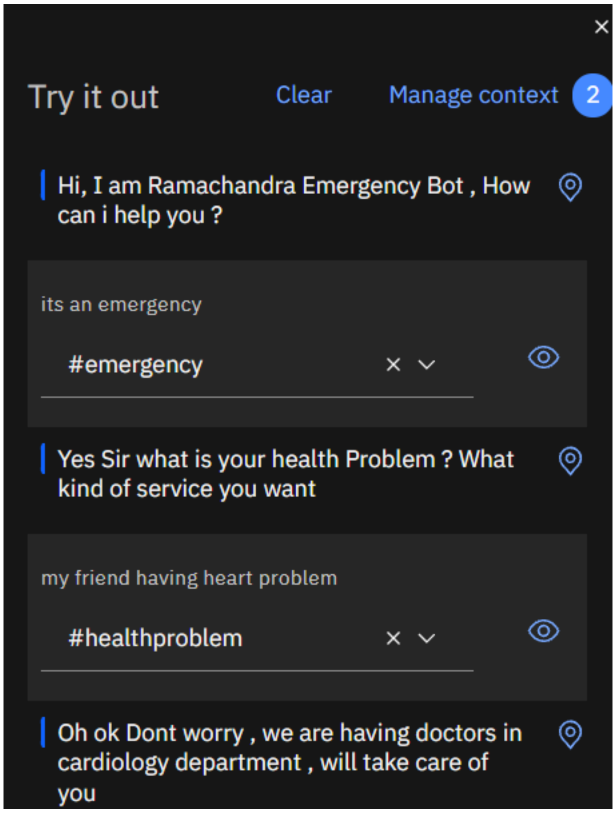


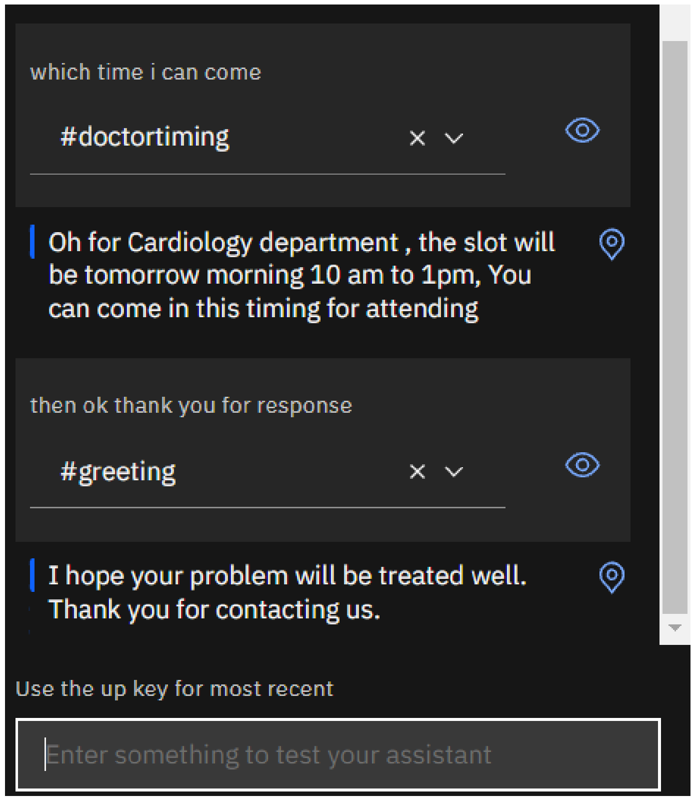
Using intents to define different intentions of chat bot and its main purposes.



Real Time Interaction :

Simple trial with the chat bot to check whether its properly responds according to the intents.





Machine Learning Model (Heart Failure Dataset) :

Importing packages and dataset :

```
In [2]: #importing dataset
        import pandas as pd
        import numpy as np
        import matplotlib.pyplot as plt
        import seaborn as sns
        from sklearn import preprocessing
```

```
In [3]: heartdata = pd.read_csv("heart_failure.csv")
```

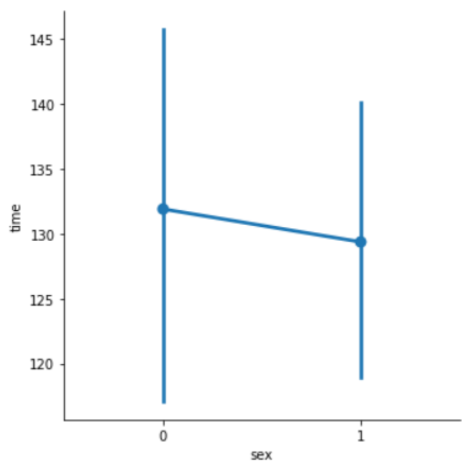
```
In [4]: heartdata.head()
```

Out[4]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00	1.9	137	1	0	7
4	65.0	1	160	1	20	0	327000.00	2.7	116	0	0	8

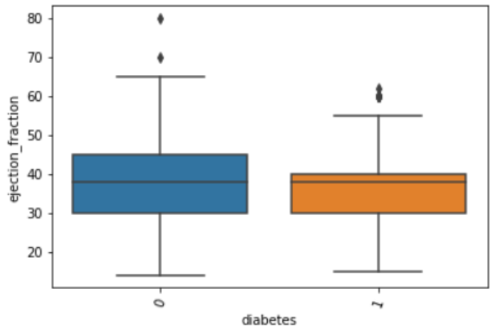
The dataset has 299 x 13 rows and columns and there is no null values in the dataset so it is clean.

```
In [13]: sns.factorplot(x='sex', y = 'time',  data = heartdata)
Out[13]: <seaborn.axisgrid.FacetGrid at 0x1a591c57e48>
```



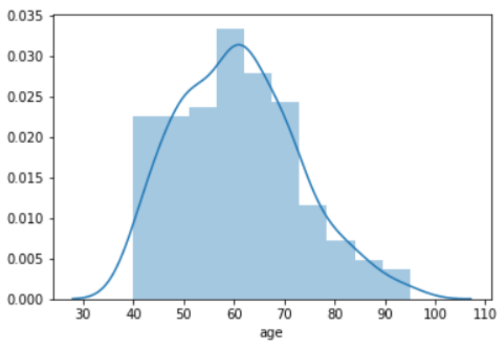
From this Graph we can see that Females (0) having more heart failure than male which is clear.

```
In [11]: sns.boxplot(x="diabetes",y="ejection_fraction",data=heartdata)
         plt.xticks(rotation = 70)#rotating x axis labels

Out[11]: (array([0, 1]), <a list of 2 Text xticklabel objects>)
```

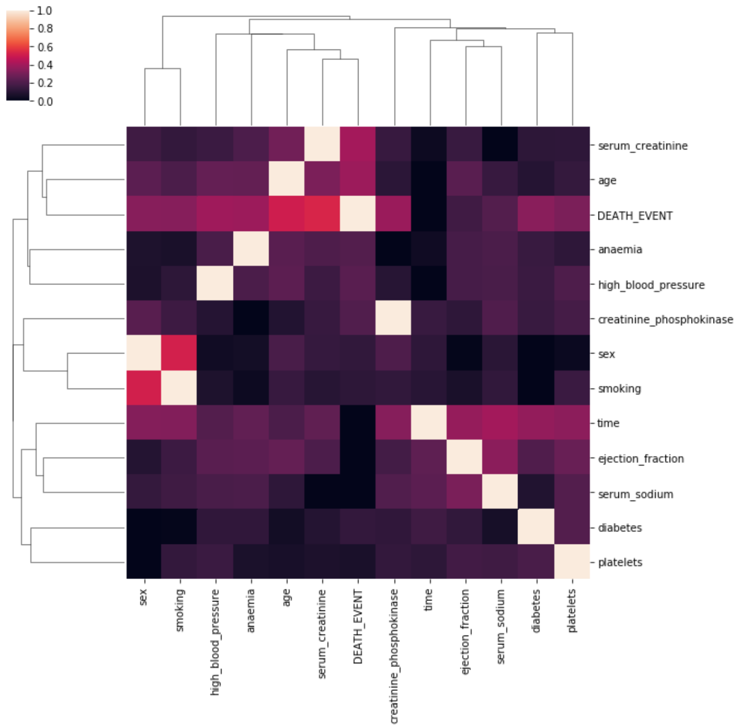


Diabetes is noted more on females and there are lot of outlier cases in female box plot, which determines about the danger and prone to heart failure.

```
In [8]: sns.distplot(heartdata["age"],kde=True,bins=10)

Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x1a5915063c8>
```



At age 60 category it had shown lots of heart failure in this dataset and 40 , 50, 70 shown significantly same average of heart failures.



Correlation of dataset is shown through the heat map for better understanding of the big picture.

In [15]:
```
heartdata['platelets'].max()
```
Out[15]:
```
850000.0
```

In [16]:
```
heartdata['platelets'].min()
```
Out[16]:
```
25100.0
```

In [17]:
```
le = preprocessing.LabelEncoder()
```

In [18]:
```
heartdata['platelets'] = le.fit_transform(heartdata['platelets'])  #to reduce the over exited values of platelets
```

In [19]:
```
heartdata.head()
```
Out[19]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	93	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	91	1.1	136	1	0	6
2	65.0	0	146	0	20	0	26	1.3	129	1	1	7
3	50.0	1	111	0	20	0	49	1.9	137	1	0	7
4	65.0	1	160	1	20	0	133	2.7	116	0	0	8

Label Encoding of the platelets column as it showing lots of deviation in data than other columns.

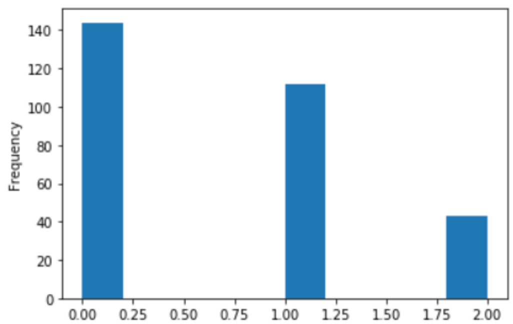
```
In [23]: #Now creating kpi
condition = [(heartdata['diabetes']>=1) & (heartdata['high_blood_pressure']==1),
                   (heartdata['diabetes']<=0) & (heartdata['high_blood_pressure']==1),
                   (heartdata['diabetes']>=1) & (heartdata['high_blood_pressure']==0),
                   (heartdata['diabetes']<=0) & (heartdata['high_blood_pressure']==0)]
```

```
In [25]: value = ['risk','average','average','normal']
```

```
In [27]: heartdata['riskratio'] = np.select(condition,value)
```

```
In [35]: heartdata['riskratio'] = le.fit_transform(heartdata['riskratio'])
```

```
In [36]: heartdata["riskratio"].plot.hist() #average 0 #normal 1 #risk 2
Out[36]: <matplotlib.axes._subplots.AxesSubplot at 0x1a59637f2e8>
```



Creating KPI using Diabetes and high blood pressure to form riskratio attribute for better analysis of the dataset in another view.

```
In [39]: #Logistic Regression
X= heartdata[['age','anaemia','creatinine_phosphokinase','diabetes','ejection_fraction','high_blood_pressure','platelets','serum_
y = heartdata['DEATH_EVENT']
```

```
In [41]: from sklearn.model_selection import train_test_split
X_train, X_test , y_train, y_test = train_test_split(X,y,test_size = 0.2, random_state = 0)
```

```
In [42]: from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

```
In [43]: from sklearn.linear_model import LogisticRegression
log_classifier = LogisticRegression(random_state = 0, solver = "liblinear",penalty='l1')
log_classifier.fit(X_train,y_train)
Out[43]: LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
                   intercept_scaling=1, l1_ratio=None, max_iter=100,
                   multi_class='warn', n_jobs=None, penalty='l1',
                   random_state=0, solver='liblinear', tol=0.0001, verbose=0,
                   warm_start=False)
```

```
In [44]: y_pred=log_classifier.predict(X_test)
```

Logistic Regression model (training the data and fitting into the model)

In [45]:
```
y_pred
```

Out[45]:
```
array([0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 0, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0], dtype=int64)
```

In [46]:
```
#accuracy score prediction
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
con_matrix = confusion_matrix(y_test,y_pred)
print(con_matrix)
accuracy_score(y_test,y_pred)
```

```
[[37  0]
 [11 12]]
```

Out[46]:
```
0.8166666666666667
```

Y Predicted values of logistic regression model and its confusion matrix for analysis of error generation in model.

In [47]:
```
from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train,y_train)
```

Out[47]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
                     metric_params=None, n_jobs=None, n_neighbors=5, p=2,
                     weights='uniform')
```

In [48]:
```
y_pred = knn.predict(X_test)
```

In [49]:
```
#accuracy score prediction
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
con_matrix = confusion_matrix(y_test,y_pred)
print(con_matrix)
accuracy_score(y_test,y_pred)
```

```
[[35  2]
 [14  9]]
```

Out[49]:
```
0.7333333333333333
```

KNN Model for predicting the y values for finding out better fit model for our dataset.

In [50]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier(criterion = 'entropy', random_state=0)
classifier.fit(X_train,y_train)
```

Out[50]:
```
DecisionTreeClassifier(class_weight=None, criterion='entropy', max_depth=None,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=0, splitter='best')
```

In [51]:
```
y_pred = classifier.predict(X_test)
```

In [52]:
```
#accuracy score prediction
from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
con_matrix = confusion_matrix(y_test,y_pred)
print(con_matrix)
accuracy_score(y_test,y_pred)
```

```
[[34  3]
 [ 6 17]]
```

Out[52]:
```
0.85
```

Decision Tree model with better accuracy but not having good score in confusion matrix showing the accuracy paradox , in this case.

```
In [53]: from sklearn.svm import SVC
         classifier = SVC()
         classifier.fit(X_train,y_train)

Out[53]: 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)

In [54]: y_pred = classifier.predict(X_test)

In [55]: #accuracy score prediction
         from sklearn.metrics import confusion_matrix, accuracy_score, classification_report
         con_matrix = confusion_matrix(y_test,y_pred)
         print(con_matrix)
         accuracy_score(y_test,y_pred)

         [[35  2]
          [11 12]]

Out[55]: 0.7833333333333333
```

SVC Model giving better results in terms of confusion matrix than decision Tree model. But in terms of accuracy it is little low than decision tree.

Finally analysing better model for our dataset which will be good for future prediction and further analysis.

Logisitic Regression showing better results in basis of confusion matrix and the accuracy score is better but it not so important while analysis the model.

Conclusion :

After doing lots of analysis with visualization and training data with different kind of machine learning models we draw an conclusion is for this dataset and attributes logistic regression is suiting well and accuracy is better with better precision and F – measure. So we proceed with our model as per the results we got for future prediction.

Thank You