Using Machine Learning Models to Improve the Risk Prediction of Cardiac Failure with help of Diagnostic Parameters

By

Nikhitha Balagani

Introduction

Cardiovascular diseases (CVDs) are the leading cause of death worldwide. Heart failure is a common clinical syndrome caused by structural and functional defects in myocardium resulting in impairment of ventricular filling or the ejection of blood. According to the American Heart Health association Heart failure is classified into 4 stages and most cardiovascular diseases can be avoided by addressing behavioral risk factors such as tobacco use, poor diet and obesity, physical inactivity, and hazardous alcohol use through population-wide strategies. The evaluation for HF is performed using various parameters: physical examination to determine the presence of clinical symptoms and signs, blood tests, including complete blood count, urinalysis, complete metabolic profile for levels of serum electrolytes (including calcium and magnesium), blood urea nitrogen, serum creatinine, glucose, fasting lipid profile, liver function tests and thyroid-stimulating hormone. People with high cardiovascular risk (due to the presence of one or more risk factors such as hypertension, diabetes, hyperlipidemia, or pre-existing disease) require early detection and management, which a machine learning model can greatly assist with. In this study, we predict the risk of cardiac failure for the patients using different machine learning models and comparing the efficiencies of those models.

Data Selection

We chose a heart failure prediction dataset for data analysis, preprocessing and training our machine learning models. This dataset contains 12 attributes like age, anemia, diabetes, platelets, sex, smoking etc.

Columns description for some of the attributes of this dataset:

1. Anemia: Decrease of red blood cells or hemoglobin (Boolean)
2. Creatinine_phosphokinase: Level of the CPK enzyme in the blood (mcg/L)
3. diabetes: If the patient has diabetes (Boolean)
4. Ejection_fraction: Ejection fraction (EF) is a measurement, expressed as a percentage, of how much blood the left ventricle pumps out with each contraction
5. High_blood_pressure: Blood hypertension
6. Platelets: Are a component of blood whose function (along with the coagulation factors)
7. Serum_creatinine: Serum creatinine is widely interpreted as a measure only of renal function
8. serum_sodium: To see how much sodium is in your blood it is particularly important for nerve and muscle function.

s.no	column	Non-null count	Dtype

0.	Age	299 non-null	float64
1.	Anemia	299 non-null	int64
2.	Creatinine_phosphate	299 non-null	int64
3.	Diabetes	299 non-null	int64
4.	Ejection_fraction	299 non-null	int64
5.	High_blood_pressure	299 non-null	int64
6.	Platelets	299 non-null	int64
7.	Serum_creatinine	299 non-null	float64
8.	Serum_sodium	299 non-null	float64
9.	Sex	299 non-null	int64
10.	Smoking	299 non-null	int64
11.	Time	299 non-null	int64
12.	Death_event	299 non-null	int64

Data Analysis and preprocessing

In Data Analysis, we analyzed our heart failure prediction dataset by importing pandas library. We used the read_csv () function in pandas to read our dataset.

s.no	Age	Anemia	creatinine_phosphokinase	Diabetes	Ejection_fraction	High blood pressure	platelets	Serum_creatinine	Serum_sodium	sex	smoking	time	Death_event
0	75	0	582	0	20	1	265000.00	1.9	130	1	0	4	1
1	55	0	7861	0	38	0	263358.03	1.1	136	1	0	6	1
2	65	0	146	0	20	0	162000.00	1.3	129	1	1	7	1
3	50	1	111	0	20	0	210000.00	1.9	137	1	0	7	1
4	65	1	160	1	20	0	327000.00	2.7	116	0	0	8	1
5	90	1	47	0	40	1	204000.00	2.1	132	1	1	8	1

Figure 2. Reading data using pandas.

We used to describe () function as part of dataset analysis, we found out mean, maximum, minimum, standard deviation, count etc. for our dataset.

contents	Age	Anemia	creatinine_phosphate	diabetes	Ejection_fraction	High_blood_pressure	Platelets	Serum-creatinine	Serum_sodium	Sex	Smoking	Time	Death_event
count	*299.00000*	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000
Mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.6254	0.648829	0.32107	130.260870	0.32107
Std	11.894809	0.496107	970.287881	0.494067	11.834841	0.478136	97804.236869	1.03451	4.412477	0.478136	0.46767	77.614208	0.46767
Min	40.000000	0.000000	23.000000	0.000000	14.000000	0.000000	25100.000000	0.50000	113.000000	0.000000	0.00000	4.000000	0.00000
25%	51.000000	0.000000	116.500000	0.000000	30.000000	0.000000	212500.000000	0.90000	134.000000	0.000000	0.000000	73.000000	0.000000
50%	60.000000	0.000000	250.000000	0.000000	38.000000	0.000000	262000.000000	1.10000	137.000000	1.000000	0.00000	115.000000	0.00000
75%	*70.000000*	1.000000	582.000000	1.000000	45.000000	0.000000	303500.000000	1.40000	140.000000	1.000000	1.00000	203.000000	1.00000
max	95.000000	1.000000	7861.000000	1.000000	80.000000	1.000000	850000.000000	9.40000	148.000000	1.000000	1.00000	285.000000	1.00000

Table 3. Describing Dataset

As the first step of data preprocessing, we tried to find out if our dataset has null values, having null values in the dataset creates ambiguity and it will lead to improper conclusions if we train the algorithms with data which contains null values. Hence, it is really important to check and handle null values. We found out that our dataset does not contain any null values.

contents	values
Age	0
Anemia	0
Creatinine_phosphokinase	0
Diabetes	0
Ejection_fraction	0
High_blood_pressure	0
Platelets	0
Serum_creatinine	0
Serum_sodium	0
Sex	0
Smoking	0
Time	0
Death-event	0

Table 4. Null values in dataset

We also tried to find out the type of our dataset and we found out that 76.9% of our dataset is of type int and 23.1% of our dataset is of type float.

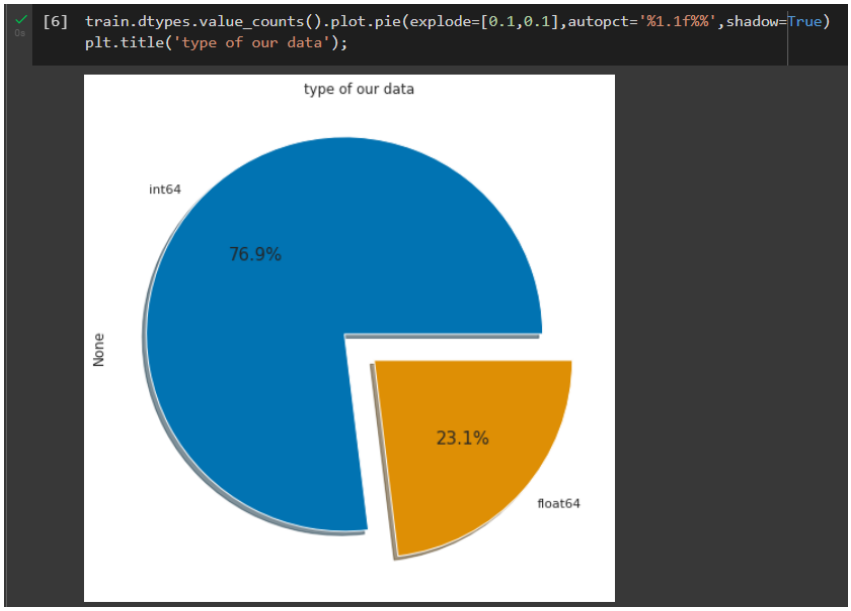


Figure 5. Type of dataset

Data Visualization

Death_Event is one of the most important attributes in our dataset as it determines the result whether the patient has cardiac arrest or not. As part of visualization, we tried to find out how much percentage of patients had died due to cardiac arrest in our dataset. We found out that 67.9% of deaths and 32.1% of patients are alive according to our heart failure dataset.

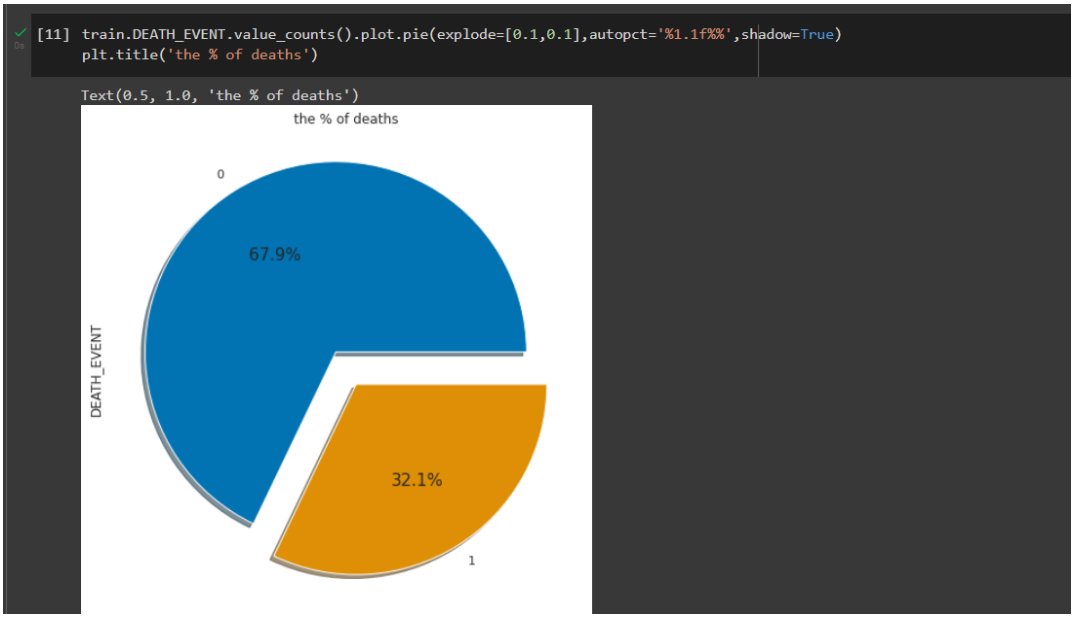


Figure 6. percentage of deaths

As we all know age is an important factor when it comes to cardiac diseases, as people are aging, they are more prone to these cardiac arrests. Hence, we tried to analyze the percentage of different age groups of patients in our dataset. We also tried to analyze which age group has faced a lot of deaths and tried to figure out if age is really an important aspect for heart failures or not.

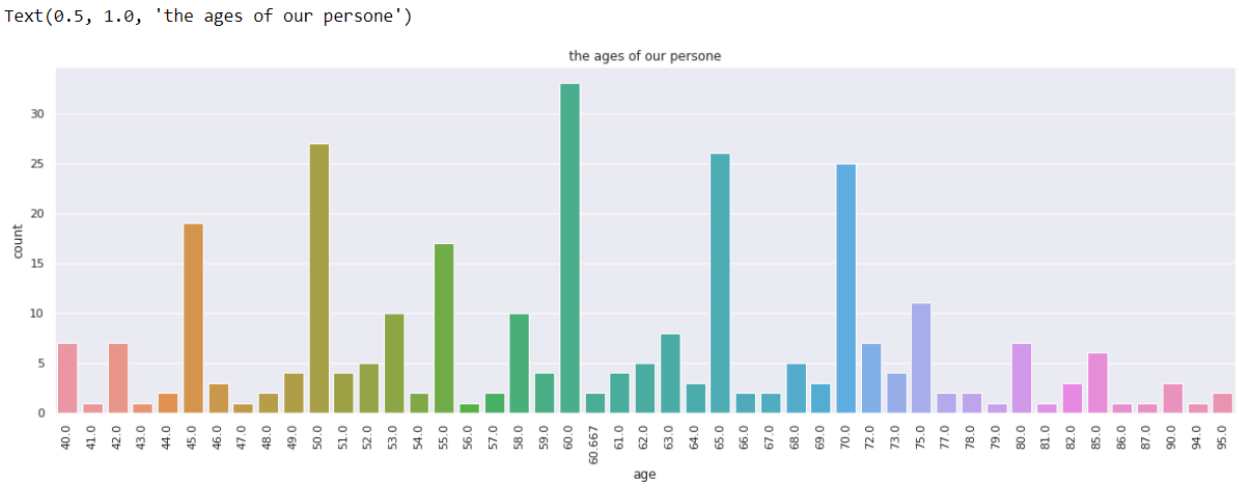
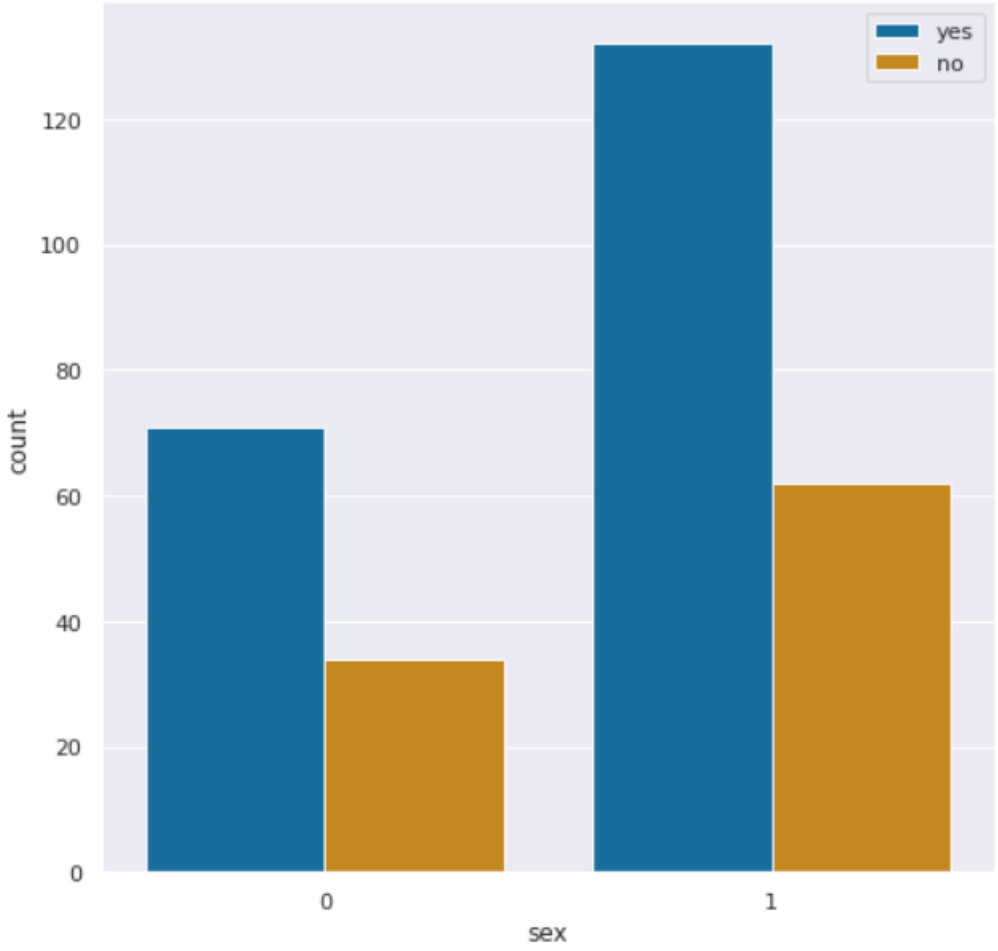
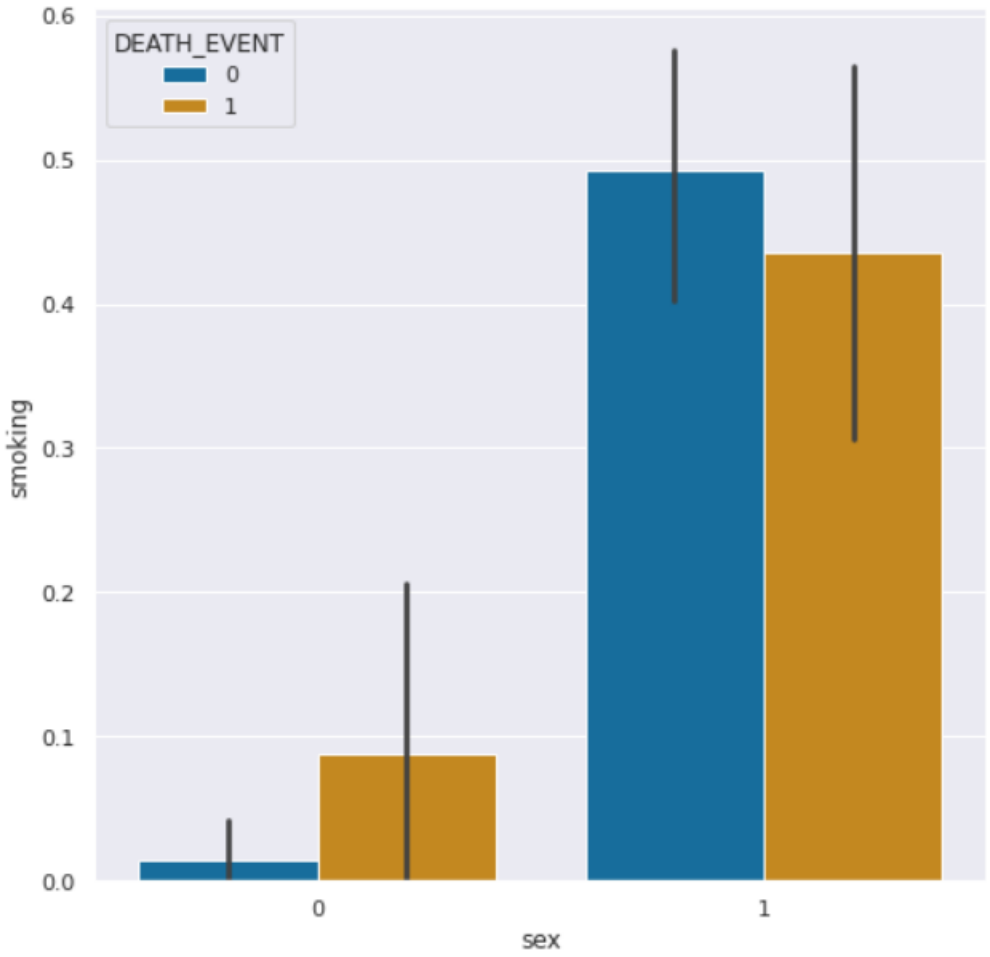


Figure 7. Distribution of Age

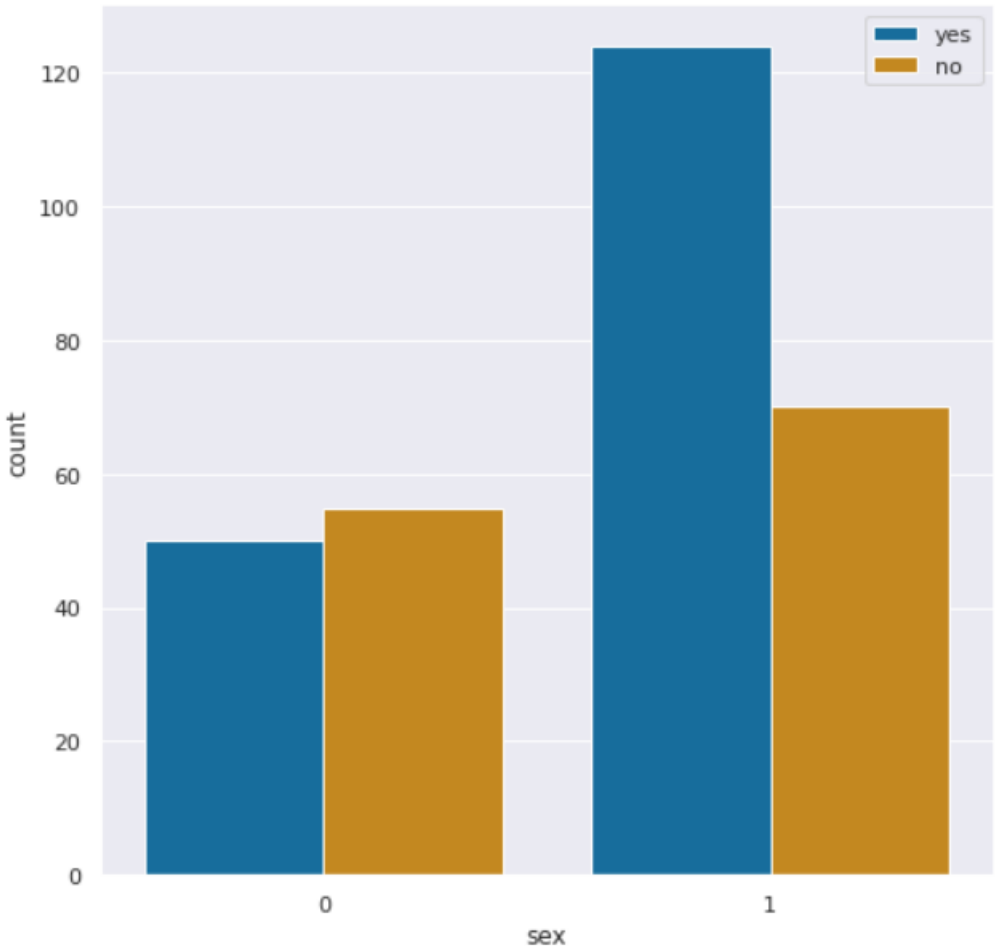
We found out that, In our dataset the highest number of patients have age in between 60 and 65 and lowest being more than 90.



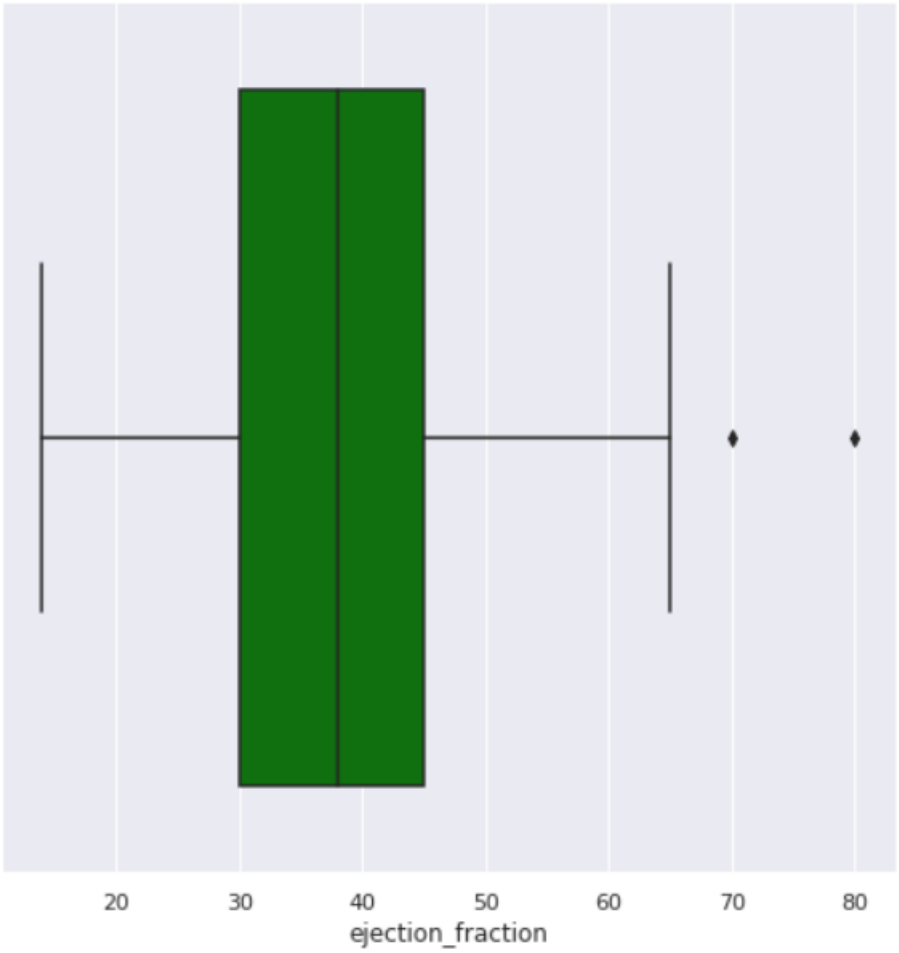
We found out that, In our dataset the highest number of male patients have Death_event when compared to women.



One more important factor that is expected to affect heart health apart from gender is smoking. Hence, we tried to analyze weather smoking is prone to death events on the gender base, we resulted with the more men are related to Death events related to smoking.



We found out that, In our dataset the highest number of male patients who have cardiac arrests or Death_events are diagnosed with the diabetes.



In our data set we found the ejection fraction value with more than or equal to 70 which is indicated in graph with the green colour.

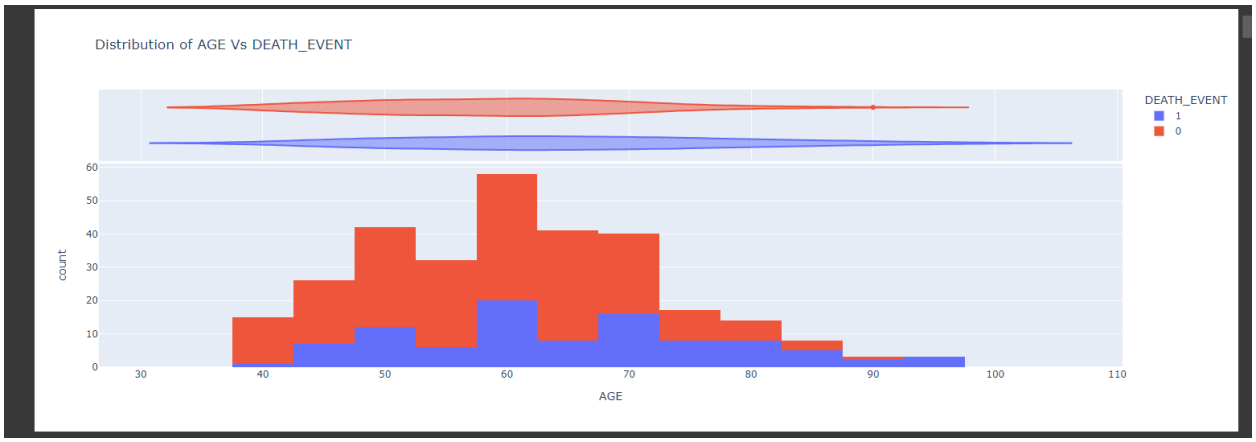


Figure 8. Distribution of Age Vs Death

One more important factor that is expected to affect heart health apart from age is gender. Hence, we tried to analyze which gender is more prone to cardiac arrests, which gender has survived more in particular cases of heart failure.

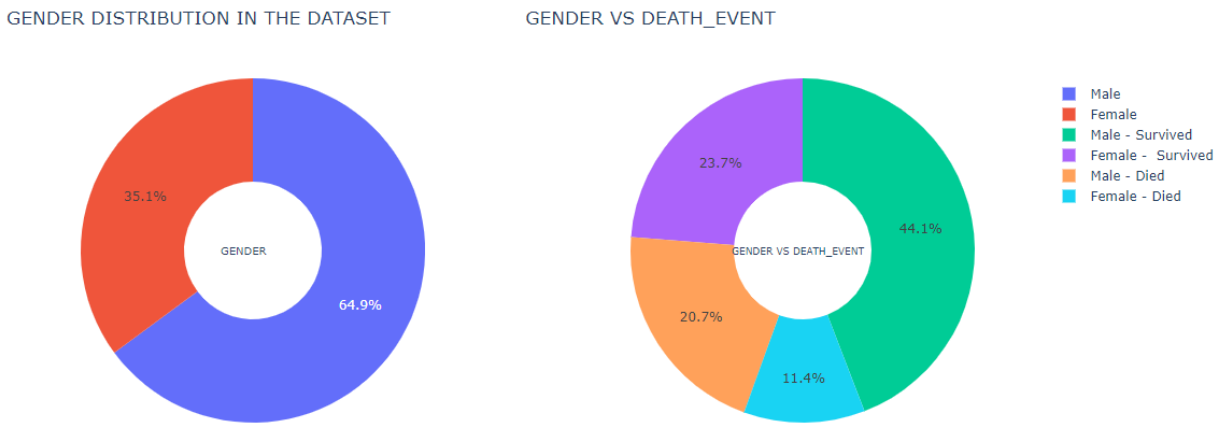


Figure 9. Gender vs Death_Event

After analysis, we found out that male is more prone to heart failures than females with a percentage of 64.9% while females being at 35.1%. Among the patients who are affected, 44.1% of male patients managed to survive and 20.7% of male patients died. Coming to females, 23.7% of female patients managed to survive and 11.4% of female patients died.

Data Correlation

One of the most important aspects of a data mining project is a way to understand the relationship between multiple variables and attributes in a dataset. It can help in predicting one attribute from another attribute

	Age	Anemia	creatinine_phosphokinase	diabetes	Ejection_fraction	High_blood_pressure	Platelets	Serum-creatinine	Serum_sodium	Sex	Smoking	Time	Death_event
Age	1.00	0.09	-0.08	-0.11	0.08	0.09	-0.05	0.19	-0.05	0.06	0.01	-0.23	0.26
Anemia	0.09	1.00	-0.19	-0.01	0.03	0.03	-0.04	0.03	0.04	-0.09	-0.11	-0.15	0.06
Creatinine_phosphokinase	-0.08	-0.19	1.00	-0.01	-0.04	-0.07	0.02	-0.01	0.06	0.08	0.00	-0.01	0.06
diabetes	-0.11	-0.01	-0.01	1.00	0.01	-0.01	0.09	-0.03	-0.09	-0.17	-0.15	0.03	0.00
Ejection_fraction	0.08	0.03	-0.04	0.01	1.00	0.02	0.09	-0.09	0.18	-0.12	-0.05	0.05	-0.28

High_blood_pressure	0.09	0.03	-0.07	-0.01	0.02	1.00	0.06	-0.04	0.04	-0.10	-0.05	-0.20	0.07
platelets	-0.05	-0.04	0.02	0.09	0.09	0.06	1.00	-0.01	0.06	-0.13	0.03	0.01	-0.04
Serum_creatinnie	0.19	0.03	-0.01	-0.03	-0.09	-0.04	-0.01	1.00	-0.21	0.04	-0.01	-0.19	0.29
Serum_sodium	-0.05	0.04	0.06	-0.09	0.18	0.04	0.06	-0.21	1.00	-0.03	0.01	0.09	-0.20
sex	0.06	-0.09	0.08	-0.17	-0.12	-0.10	-0.13	0.04	-0.03	1.00	0.44	-0.02	-0.00
smoking	0.01	-0.11	0.00	-0.15	-0.05	-0.05	0.03	-0.01	0.01	0.44	1.00	-0.02	-0.01
time	-0.23	-0.15	-0.01	0.03	0.05	-0.20	0.01	-0.19	0.09	-0.02	-0.02	1.00	-0.54
Death_event	0.26	0.06	0.06	0.00	-0.28	0.07	-0.04	0.29	-0.20	-0.00	-0.01	-0.54	1.00

Table 10. Correlation Matrix

Feature Selection

The input variable that we give to our machine learning model are features. It is an important method which is used to get rid of noise data, which is not useful for prediction our final output and using only relevant data.

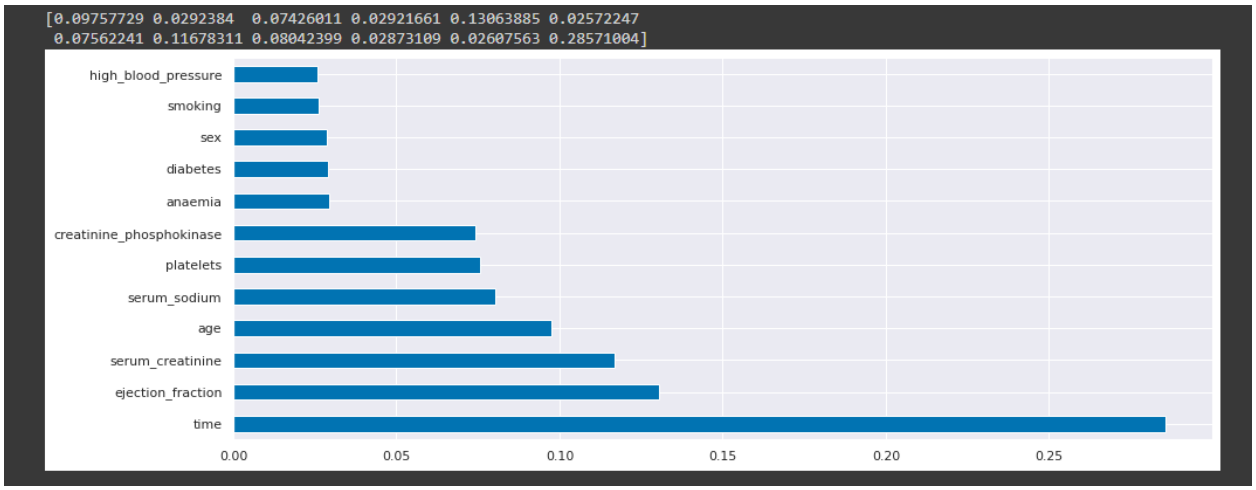


Figure 11. Feature Selection

After dropping irrelevant attributes, we trained our models only with relevant attributes.

	Ejection_fraction	Serum_creatinine	Serum_sodium	time	Death_event
0	20	1.9	130	4	1
1	38	1.1	136	6	1
2	20	1.3	129	7	1
3	20	1.9	137	7	1
4	20	2.7	116	8	1
……	…….	…….	…….	…..	…….
294	38	1.1	143	270	0
295	38	12	139	271	0
296	60	0.8	138	278	0
297	38	1.4	140	280	0
298	45	1.6	136	285	0

297 rows x 5 columns

	Ejection_fraction	Serum-creatinine	Serum_sodium	Time	Death_event
Ejection_fraction	1.000	-0.087	0.180	0.046	-0.285
Serum_creatinine	-0.087	1.000	-0211	-0.189	0.285
Serum_sodium	0.180	-0.211	1.000	0.088	-0.196
time	0.046	-0189	0.088	1.000	-0.537
Death_event	-0.285	0.286	-0.196	-0.537	1.000

Table13. Correlation after feature selection

Machine Learning Models

After data collection, data analysis, data preprocessing, data visualization, we trained a few machine learning models without our final processed dataset. We used logistic regression, K Nearest Neighbors, Support Vector Machine, Decision Tree, Artificial Neural Network, Random Forest, XG Boost and compared accuracy for these machine learning modes.

Logistic Regression

This machine learning model has an accuracy of 85.5% after training it with our dataset.

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score

model = LogisticRegression()

#Fit the model
model.fit(x_train, y_train)
y_pred = model.predict(x_test)

mylist = []
# Confusion Matrix
cm = confusion_matrix(y_test, y_pred)
# accuracy score
acc_logreg = accuracy_score(y_test, y_pred)


mylist.append(acc_logreg)
print(cm)
print(acc_logreg)


#accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', acc_logreg)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[57  4]
 [ 9 20]]
0.8555555555555555
Accuracy :  0.8555555555555555
Sensitivity :  0.9344262295081968
Specificity :  0.6896551724137931
Area Under Curve :  0.8120407009609949
```

Figure 14. Logistic Regression Accuracy

K Nearest Neighbors

This machine learning model has an accuracy of 82.2% after training it with our dataset.

```
# Finding the optimum number of neighbors

from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import confusion_matrix, accuracy_score

list1 = []
for neighbors in range(3,10):
    classifier = KNeighborsClassifier(n_neighbors=neighbors, metric='minkowski')
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred))
plt.plot(list(range(3,10)), list1)
plt.show()
```

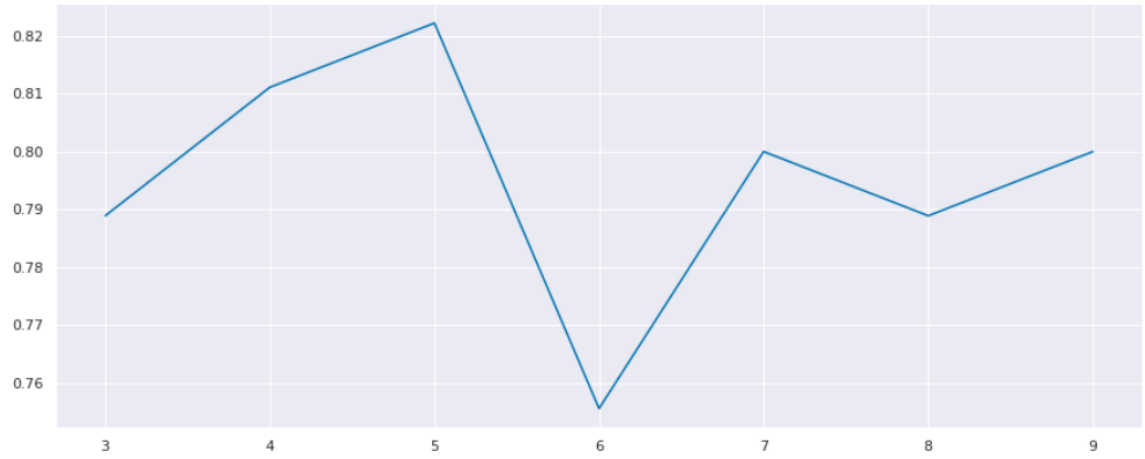


Figure 15. K Nearest Neighbors

```
# Making the confusion matrix and calculating accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
acc_knn = accuracy_score(y_test, y_pred)
mylist.append(acc_knn)
print(cm)
print(acc_knn)
total1=sum(sum(cm))

accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[55  6]
 [10 19]]
0.8222222222222222
Accuracy :  0.8222222222222222
Sensitivity :  0.9016393442622951
Specificity :  0.6551724137931034
Area Under Curve :  0.7784058790276992
```

Figure 16. K Nearest Neighbors Accuracy

Support Vector Machines

Support vector machine algorithm has an accuracy of 85.5% after training with the dataset.

```
from sklearn.svm import SVC
from sklearn.metrics import confusion_matrix, accuracy_score
list1 = []
for c in [0.5,0.6,0.7,0.8,0.9,1.0]:
    classifier = SVC(C = c, random_state=0, kernel = 'rbf')
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred))
plt.plot([0.5,0.6,0.7,0.8,0.9,1.0], list1)
plt.show()
```

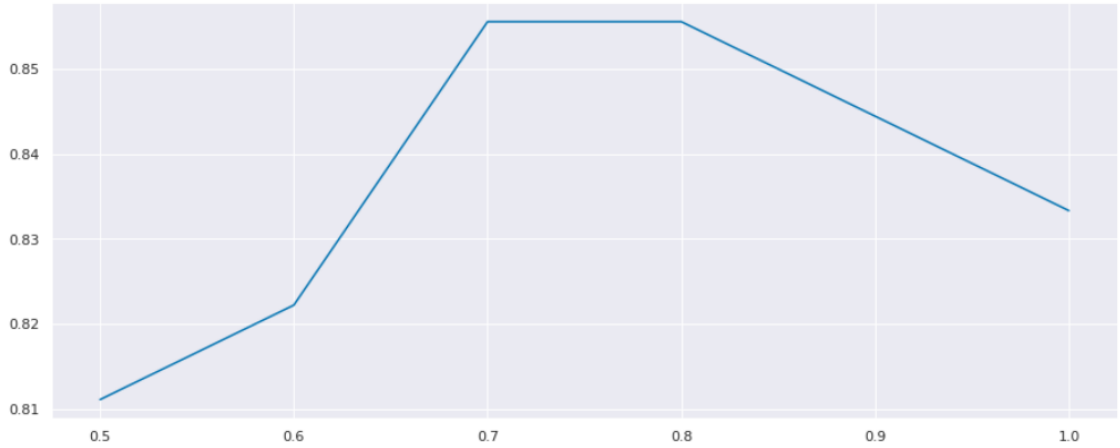


Figure 17. Support Vector Machines

```
# Making the confusion matrix and calculating accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
acc_svc = accuracy_score(y_test, y_pred)
print(cm)
print(acc_svc)
mylist.append(acc_svc)

accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[56  5]
 [ 8 21]]
0.8555555555555555
Accuracy :  0.8555555555555555
Sensitivity :  0.9180327868852459
Specificity :  0.7241379310344828
Area Under Curve :  0.8210853589598643
```

Figure 18. Support Vector Machines Accuracy

Decision Tree Classifier

This machine learning model has an accuracy of 88.8% after training with our dataset.

```
# Finding the optimum number of max_leaf_nodes

from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
list1 = []
for leaves in range(2,15):
    classifier = DecisionTreeClassifier(max_leaf_nodes = leaves, random_state=0, criterion='entropy')
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred))
#print(mylist)
plt.plot(list(range(2,15)), list1)
plt.show()
```

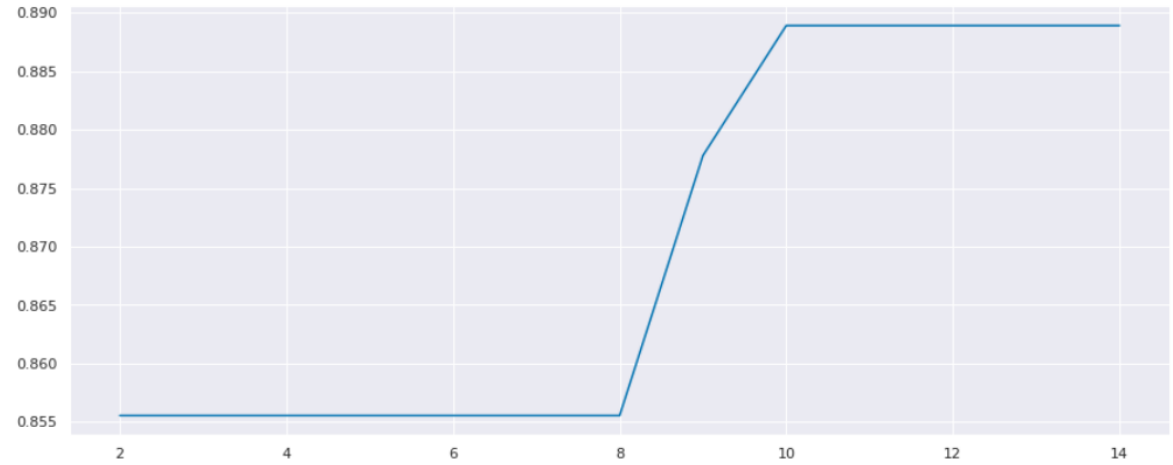


Figure 19. Decision Tree Classifier

```
# Making the confusion matrix and calculating accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
acc_decisiontree = accuracy_score(y_test, y_pred)
print(cm)
print(acc_decisiontree)
mylist.append(acc_decisiontree)

accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[56  5]
 [ 5 24]]
0.8888888888888888
Accuracy :  0.8888888888888888
Sensitivity :  0.9180327868852459
Specificity :  0.8275862068965517
Area Under Curve :  0.8728094968908987
```

Figure 20. Decision Tree Classifier Accuracy

Random Forest Classifier

This machine learning model has an accuracy of 87.7% after training with our dataset

```
#Finding the optimum number of n_estimators

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
list1 = []
for estimators in range(10,30):
    classifier = RandomForestClassifier(n_estimators = estimators, random_state=0, criterion='entropy')
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred))
#print(mylist)
plt.plot(list(range(10,30)), list1)
plt.show()
```

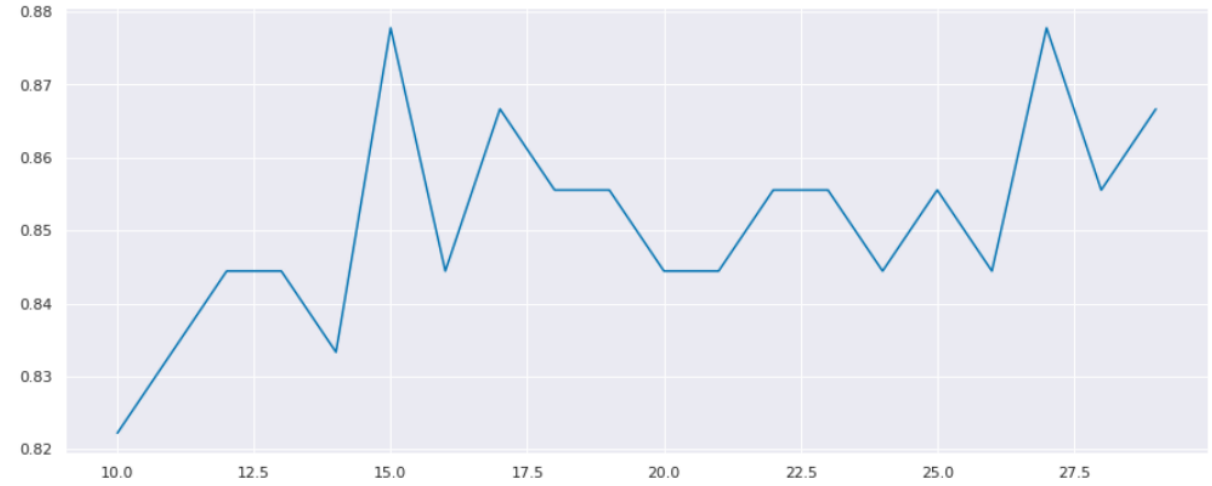


Figure 21. Random Forest Classifier

```
# Making the confusion matrix and calculating the accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
acc_randomforest = accuracy_score(y_test, y_pred)
mylist.append(acc_randomforest)
print(cm)
print(acc_randomforest)

accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[56  5]
 [ 6 23]]
0.8777777777777778
Accuracy :  0.8777777777777778
Sensitivity :  0.9180327868852459
Specificity :  0.7931034482758621
Area Under Curve :  0.8555681175805541
```

Figure 22. Random Forest Classifier Accuracy

Artificial Neural Network

Ann has an accuracy of 81.1% after training with our dataset.

```
In [214]: # Training the ANN on the training set

          ann.fit(x_train, y_train, batch_size = 16, epochs = 100)
          Epoch 92/100
          13/13 [==============================] - 0s 2ms/step - loss: 0.3189 - accuracy: 0.8792
          Epoch 93/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3171 - accuracy: 0.8792
          Epoch 94/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3173 - accuracy: 0.8792
          Epoch 95/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3164 - accuracy: 0.8792
          Epoch 96/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3157 - accuracy: 0.8744
          Epoch 97/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3150 - accuracy: 0.8696
          Epoch 98/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3158 - accuracy: 0.8792
          Epoch 99/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3142 - accuracy: 0.8792
          Epoch 100/100
          13/13 [==============================] - 0s 3ms/step - loss: 0.3143 - accuracy: 0.8792
Out[214]: <keras.callbacks.History at 0x7fa0cbe9bd50>
```

Figure 23. Artificial Neural Network

```
from sklearn.metrics import confusion_matrix, accuracy_score

# confusion matrix
cm = confusion_matrix(y_test,y_pred)
print("Confusion Matrix")
print(cm)
print()

# accuracy
ac_ann = accuracy_score(y_test,y_pred)
print("Accuracy")
print(ac_ann)
mylist.append(ac_ann)


accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
Confusion Matrix
[[54  7]
 [10 19]]

Accuracy
0.8111111111111111
Accuracy :  0.8111111111111111
Sensitivity :  0.8852459016393442
Specificity :  0.6551724137931034
Area Under Curve :  0.7702091577162238
```

Figure 24. Artificial Neural Network Accuracy

XGboost

XGBoost has an accuracy of 82.2% after training with our dataset.

In [217]:
```
from xgboost import XGBClassifier
from sklearn.metrics import confusion_matrix, accuracy_score
list1 = []
for estimators in range(10,30,1):
    classifier = XGBClassifier(n_estimators = estimators, max_depth=12, subsample=0.7)
    classifier.fit(x_train, y_train)
    y_pred = classifier.predict(x_test)
    list1.append(accuracy_score(y_test,y_pred))
#print(mylist)
plt.plot(list(range(10,30,1)), list1)
plt.show()
```

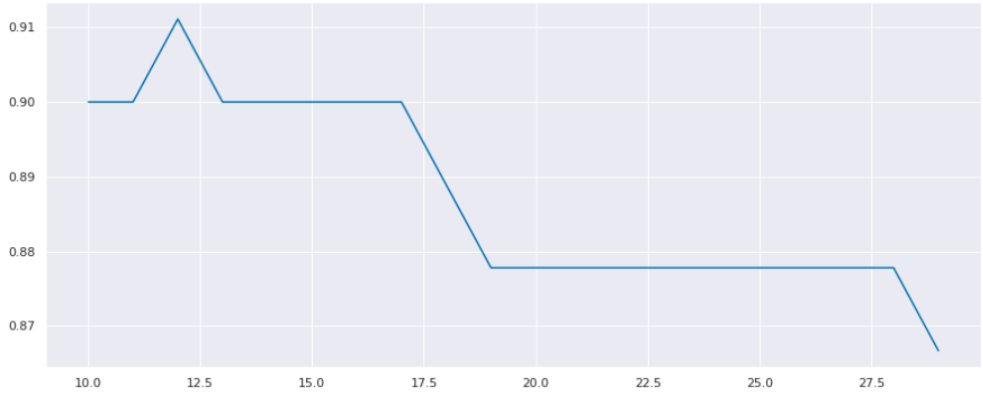


Figure 25. XG Boost

```
# Making the confusion matrix and calculating the accuracy score

from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
ac_xgboost = accuracy_score(y_test, y_pred)
mylist.append(ac_xgboost)
print(cm)
print(ac_xgboost)

accuracy1=(cm[0,0]+cm[1,1])/total1
print ('Accuracy : ', accuracy1)

sensitivity1 = cm[0,0]/(cm[0,0]+cm[0,1])
print('Sensitivity : ', sensitivity1 )

specificity1 = cm[1,1]/(cm[1,0]+cm[1,1])
print('Specificity : ', specificity1)

auc = metrics.roc_auc_score(y_test, y_pred)
print('Area Under Curve : ', auc)
```

```
[[57  4]
 [ 5 24]]
0.9
Accuracy :  0.9
Sensitivity :  0.9344262295081968
Specificity :  0.8275862068965517
Area Under Curve :  0.8810062182023741
```

Figure 26. XGBoost Accuracy

Conclusion

After experimenting with different machine learning models, we finally found out that Decision tree and random forest has highest accuracy of 88.88% and Artificial Neural Network being at the least with an accuracy of 87.77%.

Models	Sensitivity	Specificity	Accuracy	Area Under Curve
Linear regression	93.44%	68.96%	85.55%	81.20%
K Nearest neighbors	90.16%	65.51%	82.22%	77.84%

Support vector machine	91.80%	72.41%	85.55%	82.10%
Decision tree	91.80%	82.75%	88.88%	87.28%
Random forest	91.80%	79.31%	87.77%	85.55%
Artificial neural network	85.24%	72.41%	81.11%	77.02%

Figure 27. Comparing Accuracies

```
plt.rcParams['figure.figsize']=15,6
sns.set_style("darkgrid")
ax = sns.barplot(x=models.Model, y=models.Score, palette = "rocket", saturation =1.5)
plt.xlabel("Classifier Models", fontsize = 20 )
plt.ylabel("% of Accuracy", fontsize = 20)
plt.title("Accuracy of different Classifier Models", fontsize = 20)
plt.xticks(fontsize = 12, horizontalalignment = 'center', rotation = 8)
plt.yticks(fontsize = 13)
for p in ax.patches:
    width, height = p.get_width(), p.get_height()
    x, y = p.get_xy()
    ax.annotate(f'{height:.2%}', (x + width/2, y + height*1.02), ha='center', fontsize = 'x-large')
plt.show()
```

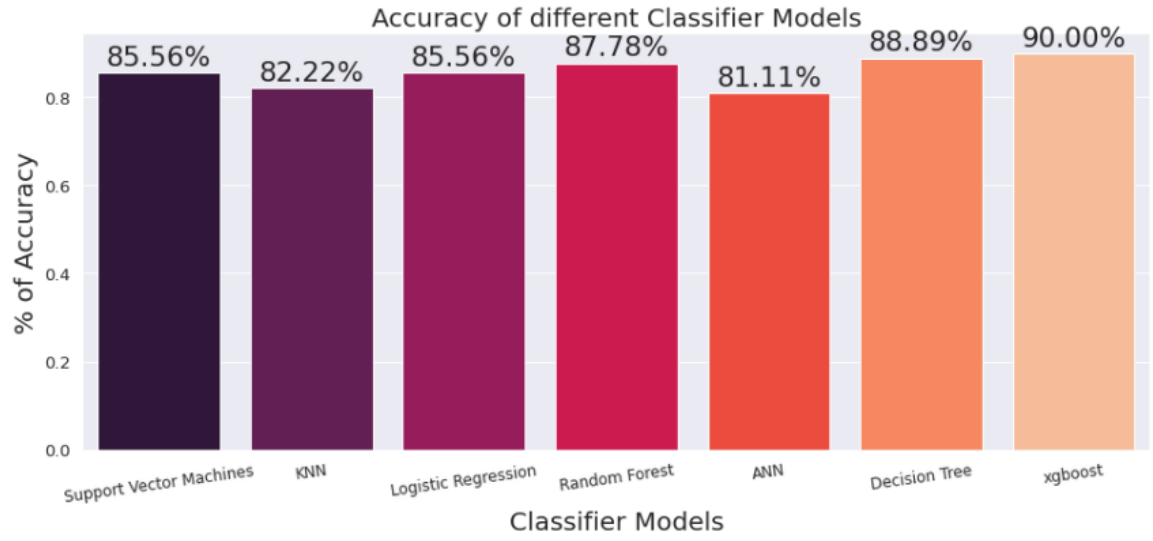


Figure 28. Comparing Accuracies using graph

What we have learnt in this project

This project has a critical role in improving our theoretical and practical knowledge in data mining. We understood the importance of data preprocessing, feature selection as we have seen a lot of difference in accuracy before and after implementing those techniques. We always had a

misconception that if a machine learning model is advanced, then its accuracy would be higher for any type of data. But in reality, this is not true as data plays a key role for improving accuracy rather than using an advanced machine learning model. Here in our case, Decision tree worked better than an advanced XG boost or Artificial Neural Network model because our data set is best fit for logistic regression.

Team Roles

Chandana Dasarraju - Data collection, preprocessing
Aarif Chittoor Mohammad - Data visualization, exploration
Nikhil Balagani - Training Machine learning models

REFERENCES

1. Davide Chicco, Giuseppe JurmanL Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone. BMC Medical

Informatics and Decision Making 20, 16 (2020)
https://doi.org/10.1186/s12911-020-1023-5

2. Excel:
   Microsoft Corporation. (2018). Microsoft Excel. Retrieved from
   https://office.microsoft.com/excel
3. Dataset retrieved from Kaggle
   https://www.kaggle.com/code/midouazerty/heart-failure-prediction/data
4. Python Software Foundation. Python Language Reference, Version 3.10.7. Available to
   https://www.python.org/downloads/
5. Kluyver, T. et al., 2016. Jupyter Notebooks – a publishing format for reproducible
   computational workflows
   https://jupyter.org/