Heart Failure Prediction

Sumika Mukerji

Abstract

This research work investigates the factors that lead to heart failure and how to predict heart failure using a machine learning model. Dataset obtained from Kaggle (https://www.kaggle.com/andrewmvd/heart-failure-clinical-data) has been analyzed and a machine learning model was built using the Decision Tree algorithm. This algorithm was finalized after comparing the outcomes with 3 other machine learning models (using Logistic Regression, Support Vector and K Neighbors algorithms). Statistical plots were used to visualize the influences of the different factors. The most important influences on heart failure seem to be due to age and clinical features of ejection fraction, serum creatinine and serum sodium. Blood pressure does not have a strong effect on its own but may influence in combination with other factors. In summary, it is possible to predict heart failure using a machine learning model upto an accuracy of around 76%.

Motivation

Every year significant number of people across the globe lose their loved ones suddenly due to heart failure. Cardiovascular diseases (CVDs) are the topmost causes of death worldwide, taking around 17.9 million lives annually, which accounts for 31% of all deaths globally. Heart failure is a common event caused by CVDs. Most cardiovascular diseases can be prevented by addressing the risk factors. People with cardiovascular disease or who are at high cardiovascular risk need early detection and management wherein a machine learning model can be of great help.

Dataset(s)

In the research, the Heart Failure Prediction dataset(1) has been used. It contains individual records of patients with 12 clinical features for predicting death events. The features are namely, age, anaemia, creatinine phosphokinase, diabetes, ejection fraction, high blood pressure, platelets, serum creatinine, serum sodium, sex, smoking, time and the target variable is death event. This dataset contains 300 records where each individual record contains the medical features of a person.

This data was created by Davide Chicco, Giuseppe Jurman and published on Kaggle.

(1) https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

Data Preparation and Cleaning

This dataset contains 12 features and 300 records, and the following steps were taken to prepare and clean the data:

- First, data download was performed manually from Kaggle
- Data was loaded into a Pandas dataframe
- Nature of data was understood using Exploratory Data Analysis (EDA)
- Preliminary analysis to select the relevant features for model building
- Dropped rows that had missing values
- Dropped rows that had data gaps e.g. negative values of age
- Split the data into training and test set
- Four classification algorithms were used to build models and their outcomes were compared

Research Question(s)

- What are the key factors that can lead to death due to heart failure?
- Build a machine learning model to predict whether a particular person will die due to heart failure given his medical features

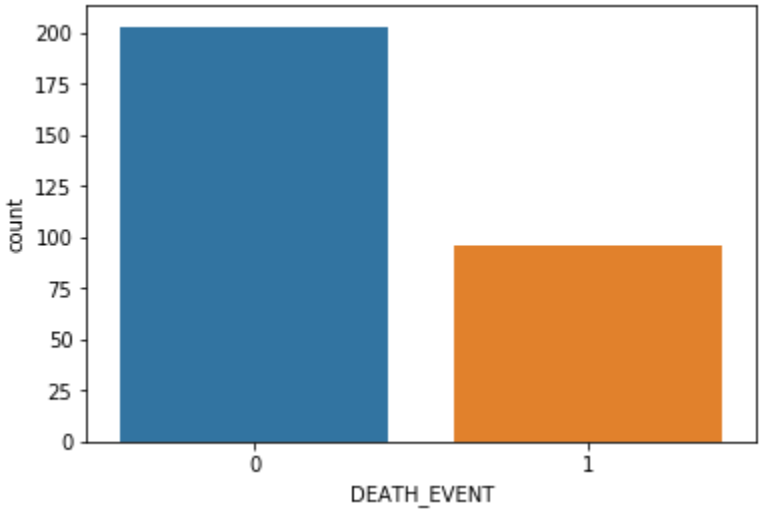
Methods

Data Analysis was done using the following methods:

- Correlations Heatmap was used to find the pairwise correlation of all features
- Top 5 features showing high correlation with Death Event feature were selected
- Statistical methods were used to visualize the influence of these top 5 features on Death Event: Age, High Blood Pressure, Ejection Fraction, Serum Creatinine and Serum Sodium
- Four Machine Learning models were fitted to the data after splitting the data into training and test sets: Decision Tree, Logistic Regression, Support Vector, K Neighbors
- The outcomes of the four models were compared and Decision Tree was found to be the most promising model

Findings: EDA

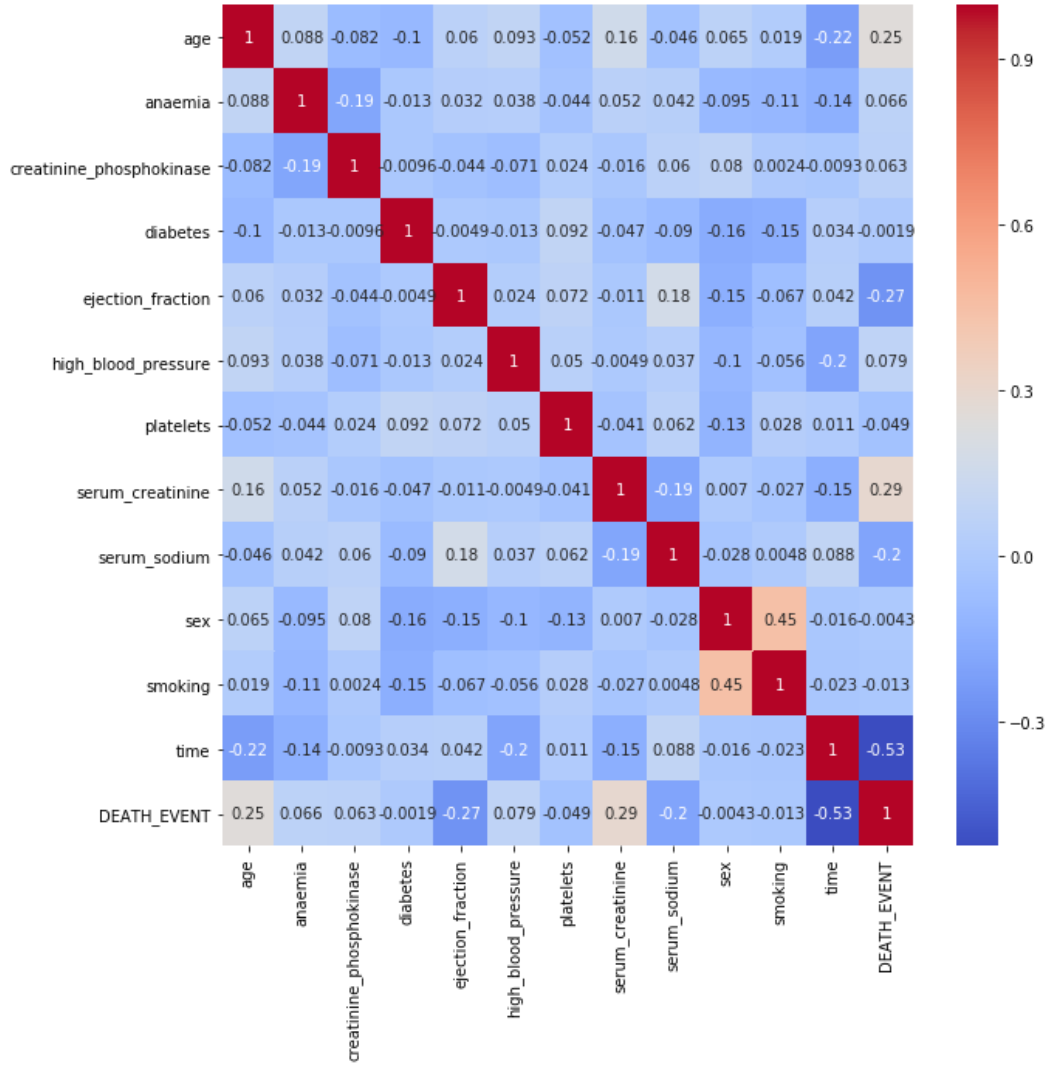
Classification Count: Death Event



DEATH_EVENT
1: Person Died
0: Person Survived

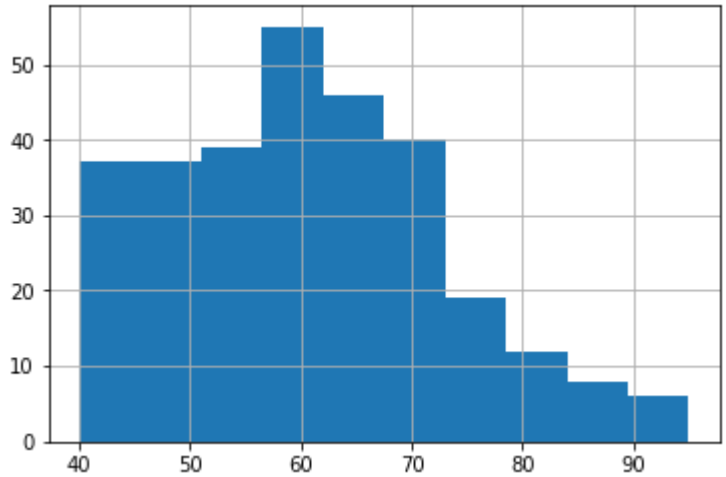
Findings: EDA

Correlations Heatmap

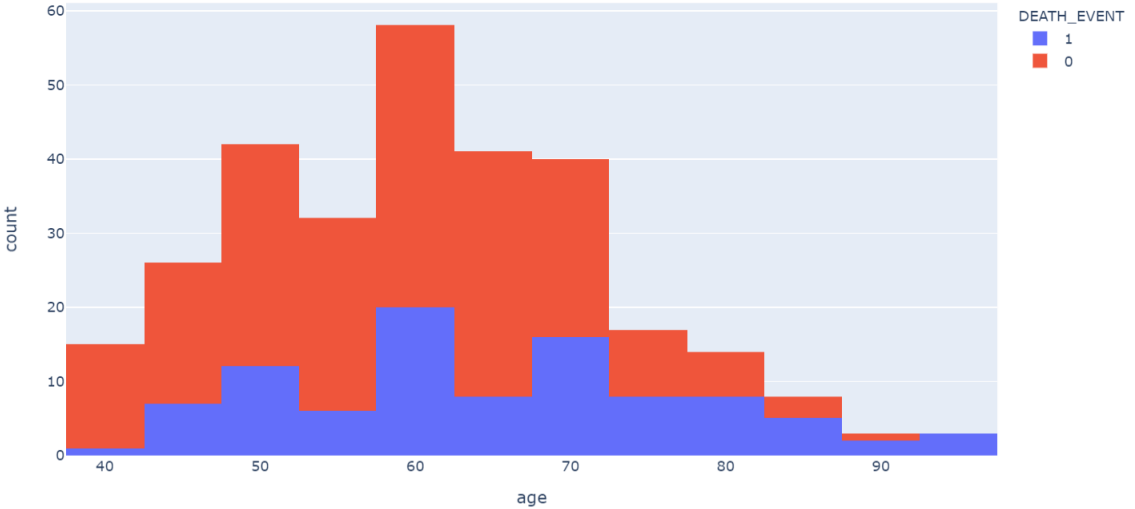


Findings: EDA

Feature: Age



Age Distribution

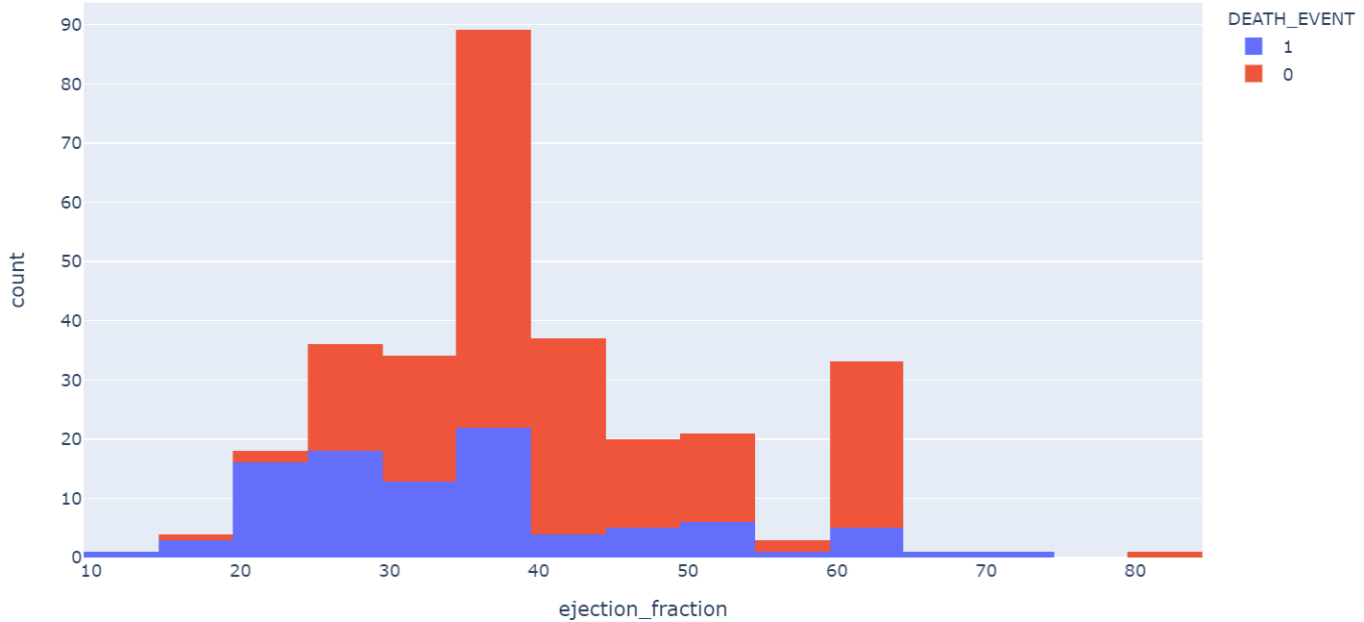


Age Vs. Death Event

- Age distribution in the dataset is between 40 to 95 years
- The count of people is maximum close to 60 years of age, followed by 70 years
- The count of people is very low for age greater than 80
- Death count is highest in the age group of 58-62 years

Findings: EDA

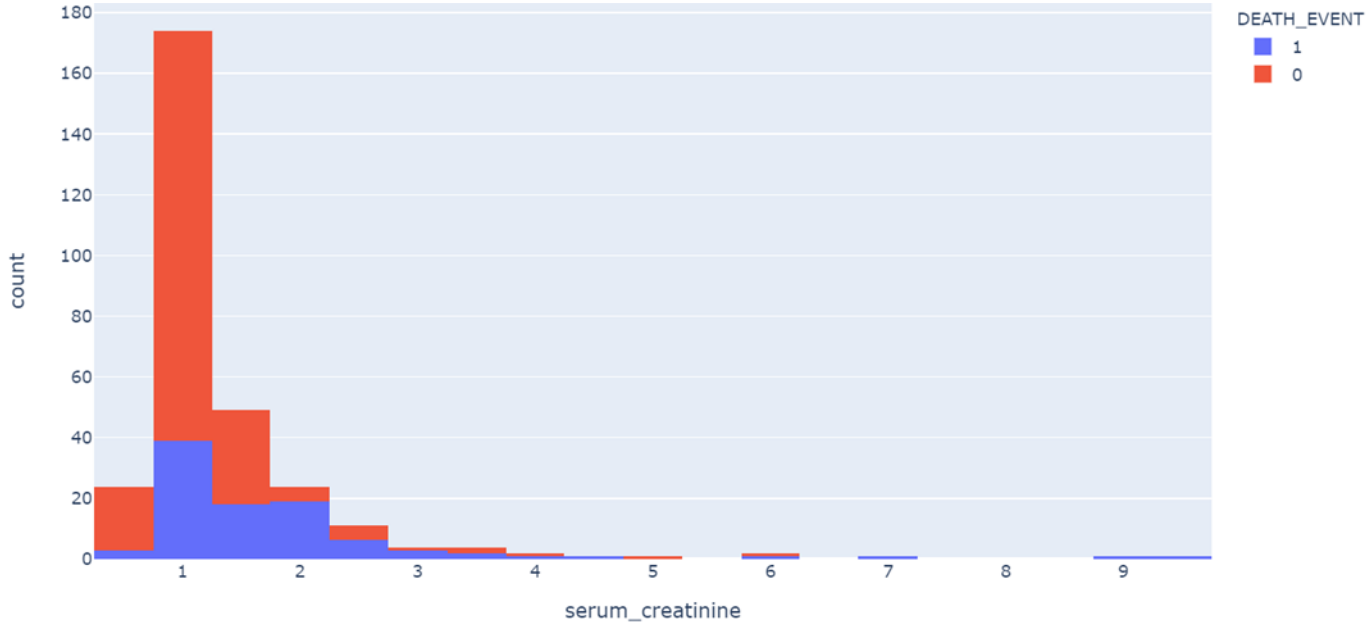
Feature: Ejection Fraction



Ejection Fraction Vs. Death Event

Findings: EDA

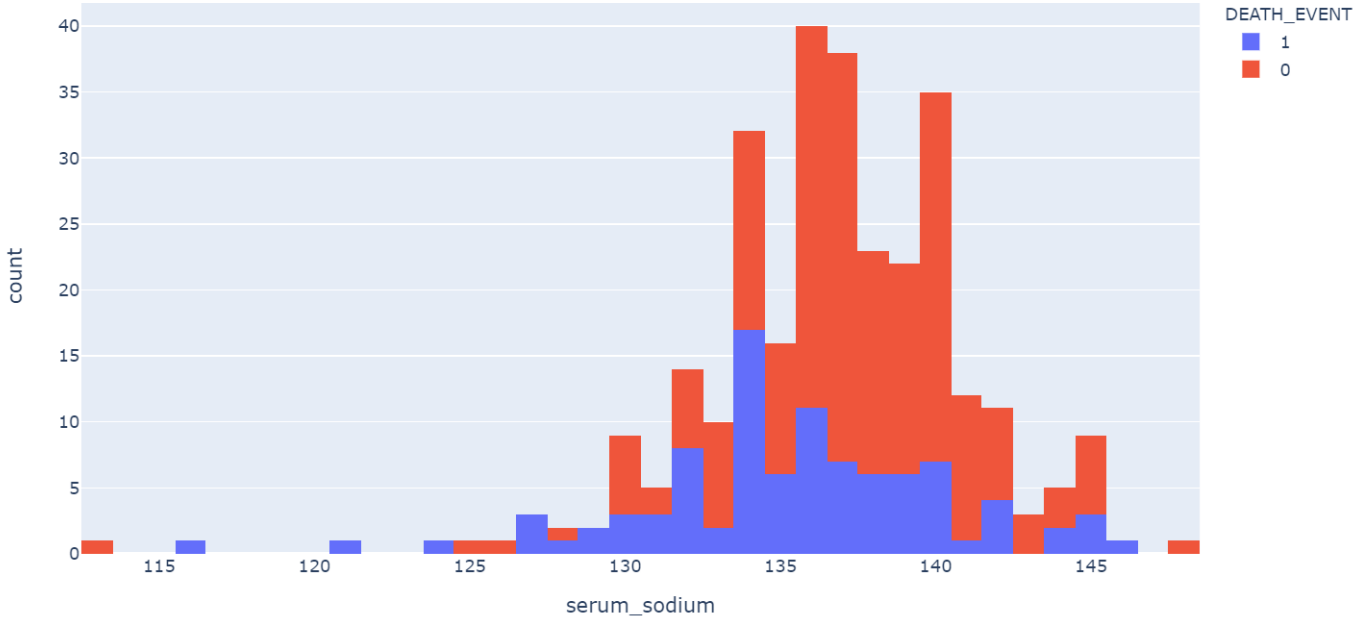
Feature: Serum Creatinine



Serum Creatinine Vs. Death Event

Findings: EDA

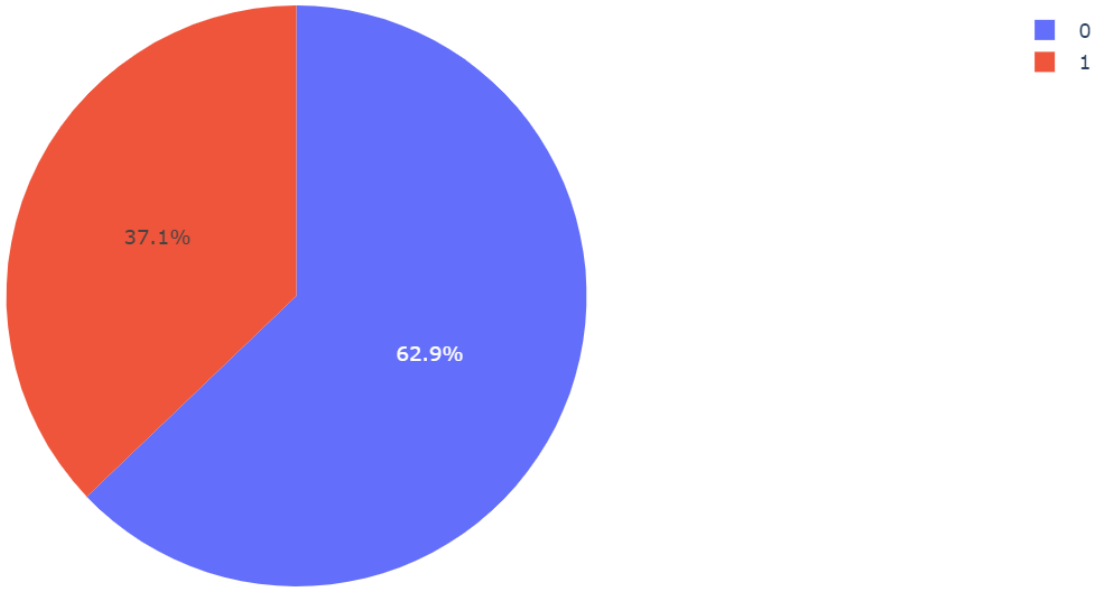
Feature: Serum Sodium



Serum Sodium Vs. Death Event

Findings: EDA

Feature: High Blood Pressure



Only 37% of the people with High Blood Pressure have died and hence this feature was **not selected for model building**.

Findings: Model Building

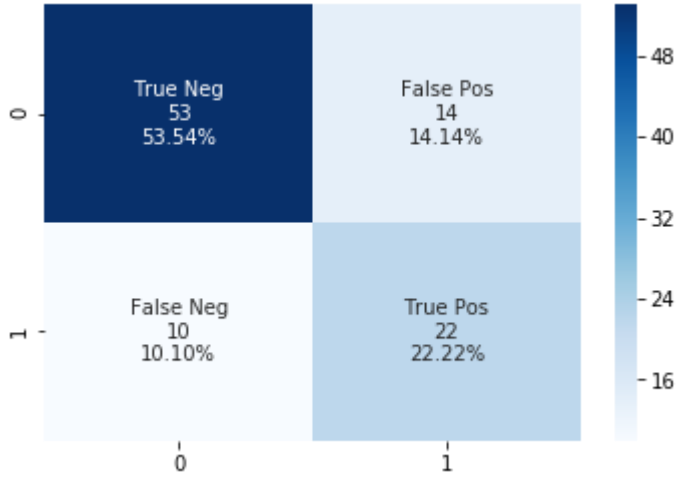
For Model Building, four algorithms were used, and the results were compared:

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision Tree 	75.76%	61.11%	68.75%	0.647058
Logistic Regression	73.74%	61.54%	50%	0.55172
Support Vector	69.70%	100%	6.25%	0.11764
K Neighbors	72.73%	69.23%	28.13%	0.4

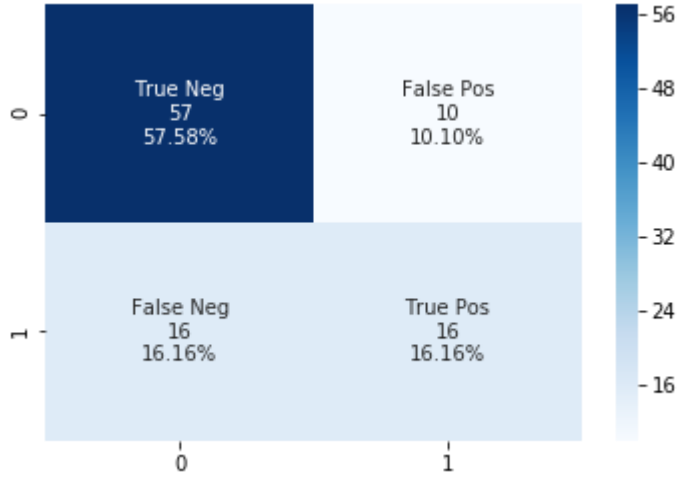
Decision Tree gave the best outcomes

Findings

Confusion Matrix results of Decision Tree and Logistic Regression algorithms:



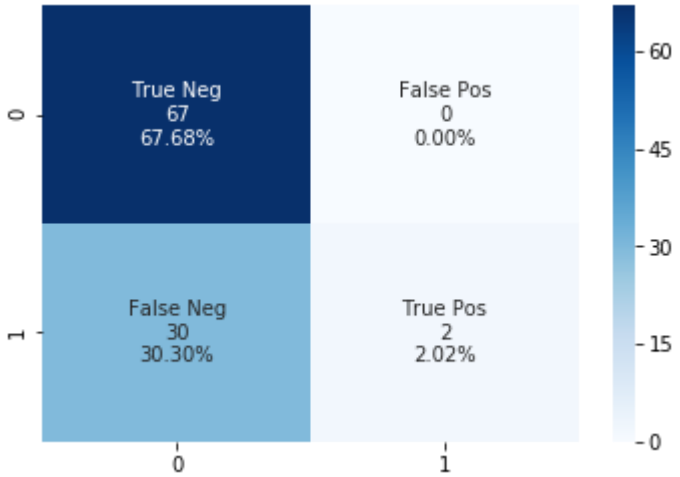
Decision Tree Classifier



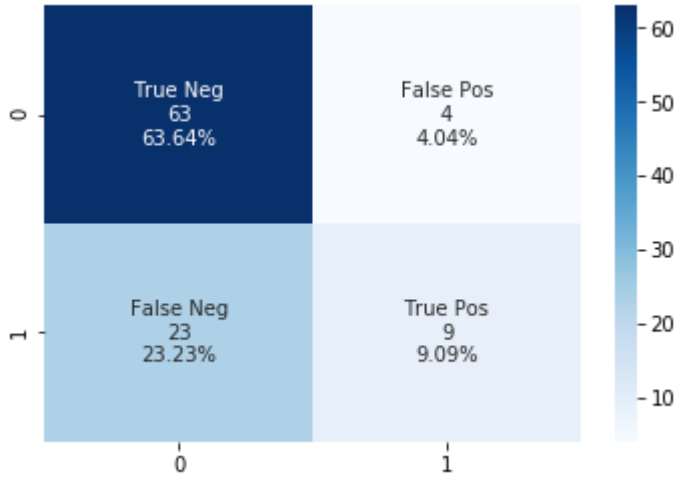
Logistic Regression Classifier

Findings

Confusion Matrix results of Support Vector and K Neighbors algorithms:



Support Vector Classifier



K Neighbors Classifier

Limitations

Data limitation : This dataset does not contain geography details of patients, which is a significant factor in analyzing any disease. It would have helped a great deal to explore the geography specific patterns and this feature could have been included in model building.

Conclusions

The death of a patient due to heart failure seems to depend predominantly on age and clinical features of ejection fraction, serum creatinine and serum sodium. Blood pressure does not have a strong effect on its own but may influence in combination with other factors. The best machine learning model for predicting heart failure is the Decision Tree classifier. In summary, it is possible to predict heart failure using a machine learning model upto an accuracy of around 76%.

Acknowledgements

The dataset used was created by Davide Chicco, Giuseppe Jurman and published on Kaggle: https://www.kaggle.com/andrewmvd/heart-failure-clinical-data

References

None – I did the work entirely on my own.

Prediction of Death: due to Heart Failure

Project Goals

Every year significant number of people across the globe lose their loved ones suddenly due to heart failure. Cardiovascular diseases (CVDs) are the number one cause of death globally, taking an estimated 17.9 million lives each year, which accounts for 31% of all deaths worldwide. Heart failure is a common event caused by CVDs and can be predicted by analysing vital clinical parameters of a patient. Early detection can help in taking appropriate corrective action on time and thus save lives. Most cardiovascular diseases can be prevented by addressing behavioural risk factors such as tobacco use, unhealthy diet and obesity, physical inactivity and harmful use of alcohol using population-wide strategies. In this project, Machine Learning has been used to develop a model that can predict heart failure in a patient, given his clinical features. This project aims to explore the following areas:

1. Explore the key factors that can lead to death due to heart failure (Using EDA)

2. Build a machine learning model to predict whether a particular person will die due to heart failure

In [1]:
```
#!pip install plotly - this library should be installed if not already present. In this case it is already installed.
```

In [2]:
```
#Importing the Necessary Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
from sklearn.metrics import confusion_matrix,accuracy_score,f1_score, recall_score, precision_score, classification_report
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
```

In [3]:
```
patient_data = pd.read_csv('heart_failure_clinical_records_dataset.csv')
```

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

Out[4]:

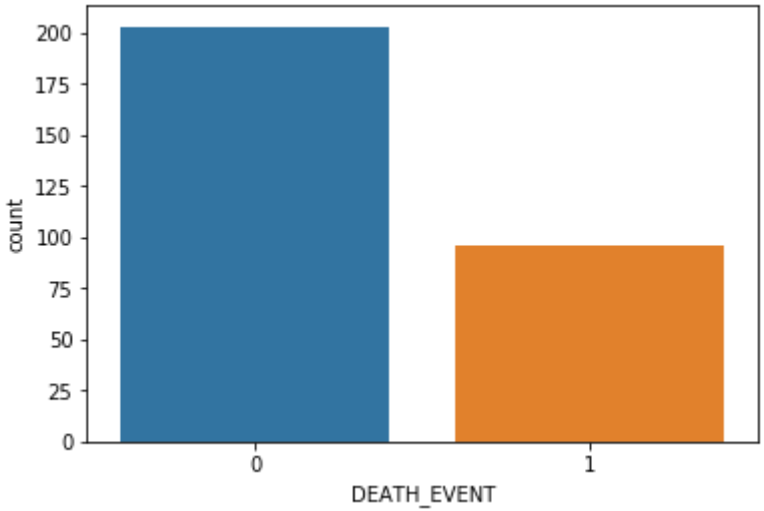
	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time
0	75.0	0	582	0	20	1	265000.00	1.9	130	1	0	4
1	55.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6
2	65.0	0	146	0	20	0	162000.00	1.3	129	1	1	7
3	50.0	1	111	0	20	0	210000.00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

Exploratory Data Analysis (EDA)

In [5]:
```
# Classification Count
print(patient_data['DEATH_EVENT'].value_counts())
sns.countplot(x='DEATH_EVENT', data=patient_data)
```

```
0    203
1     96
Name: DEATH_EVENT, dtype: int64
```

Out[5]: <matplotlib.axes._subplots.AxesSubplot at 0x21b86bd4390>



DEATH_EVENT

- 1: Person Died
- 0: Person Survived

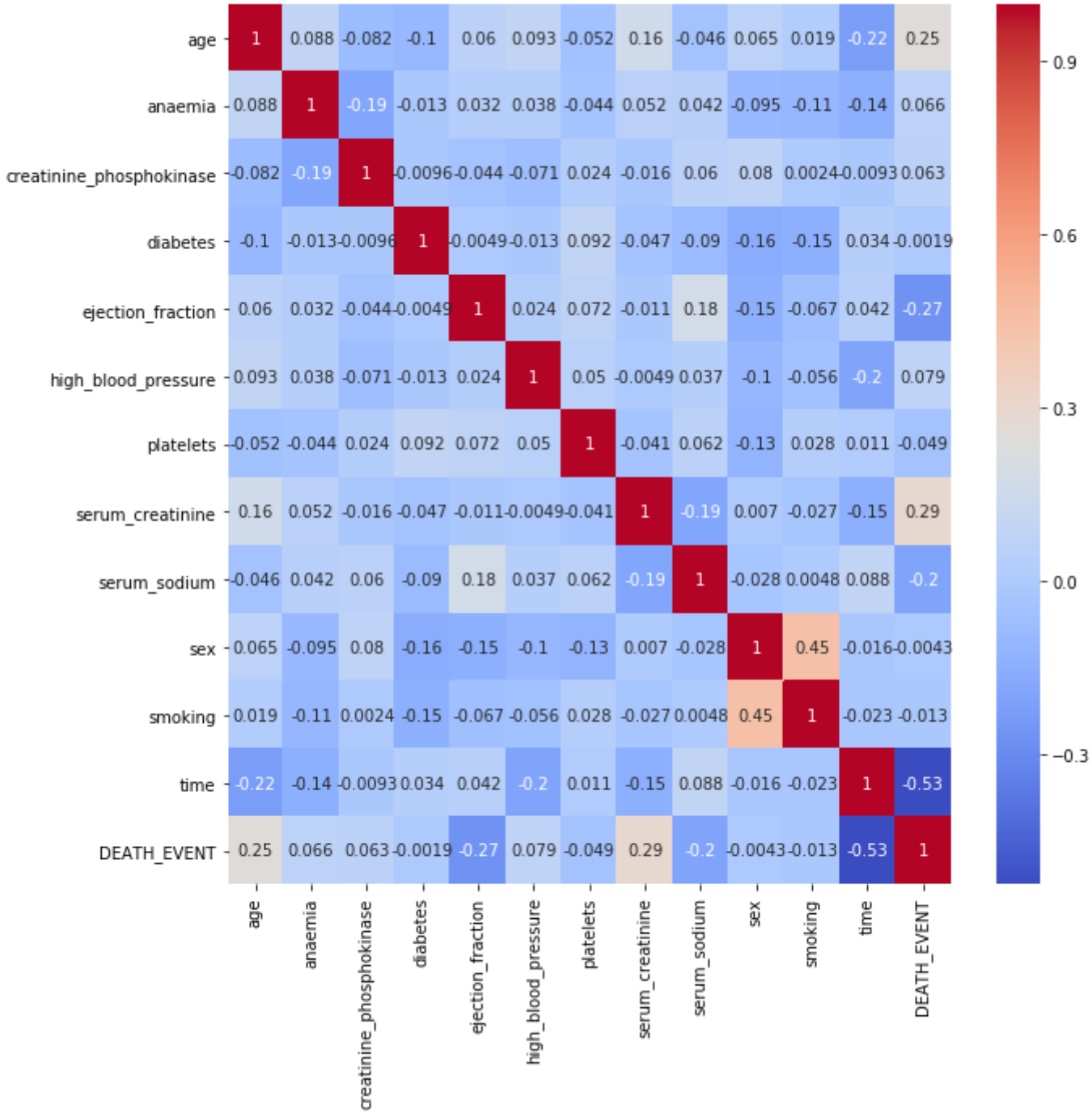
In [6]:
```
#Explore correlations in the dataframe
patient_data.corr()
```

Out[6]:

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	seru
age	1.000000	0.088006	-0.081584	-0.101012	0.060098	0.093289	-0.052354	0.159187	
anaemia	0.088006	1.000000	-0.190741	-0.012729	0.031557	0.038182	-0.043786	0.052174	
creatinine_phosphokinase	-0.081584	-0.190741	1.000000	-0.009639	-0.044080	-0.070590	0.024463	-0.016408	
diabetes	-0.101012	-0.012729	-0.009639	1.000000	-0.004850	-0.012732	0.092193	-0.046975	
ejection_fraction	0.060098	0.031557	-0.044080	-0.004850	1.000000	0.024445	0.072177	-0.011302	
high_blood_pressure	0.093289	0.038182	-0.070590	-0.012732	0.024445	1.000000	0.049963	-0.004935	
platelets	-0.052354	-0.043786	0.024463	0.092193	0.072177	0.049963	1.000000	-0.041198	
serum_creatinine	0.159187	0.052174	-0.016408	-0.046975	-0.011302	-0.004935	-0.041198	1.000000	
serum_sodium	-0.045966	0.041882	0.059550	-0.089551	0.175902	0.037109	0.062125	-0.189095	
sex	0.065430	-0.094769	0.079791	-0.157730	-0.148386	-0.104615	-0.125120	0.006970	
smoking	0.018668	-0.107290	0.002421	-0.147173	-0.067315	-0.055711	0.028234	-0.027414	
time	-0.224068	-0.141414	-0.009346	0.033726	0.041729	-0.196439	0.010514	-0.149315	
DEATH_EVENT	0.253729	0.066270	0.062728	-0.001943	-0.268603	0.079351	-0.049139	0.294278	

In [7]:
```
fig, ax= plt.subplots(figsize=(10,10))
sns.heatmap(patient_data.corr(),
        xticklabels=patient_data.corr().columns,
        yticklabels=patient_data.corr().columns, cmap= 'coolwarm', annot = True)
```

Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x21b86c7f710>



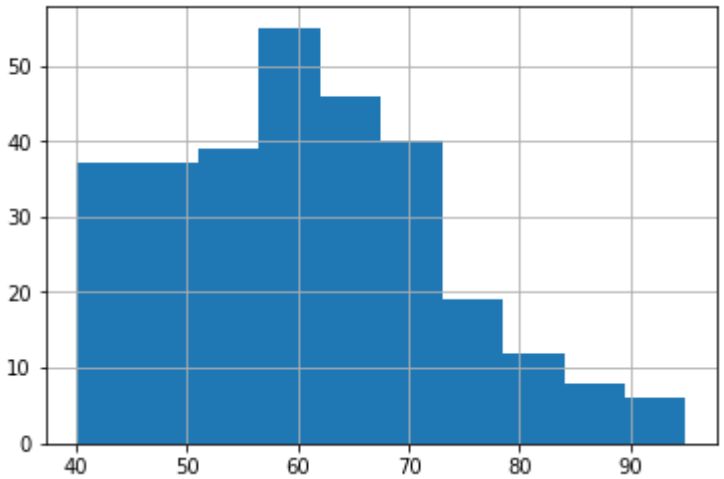
Based on the above results, correlations of following features with DEATH EVENT will be further explored using Data Visualizations:

- Age
- High Blood Pressure
- Ejection Fraction
- Serum Creatinine
- Serum Sodium

Age Spread

In [8]:
```
#Plot the age spread given in the dataset
patient_data['age'].hist()
```

Out[8]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b870a6d68>
```

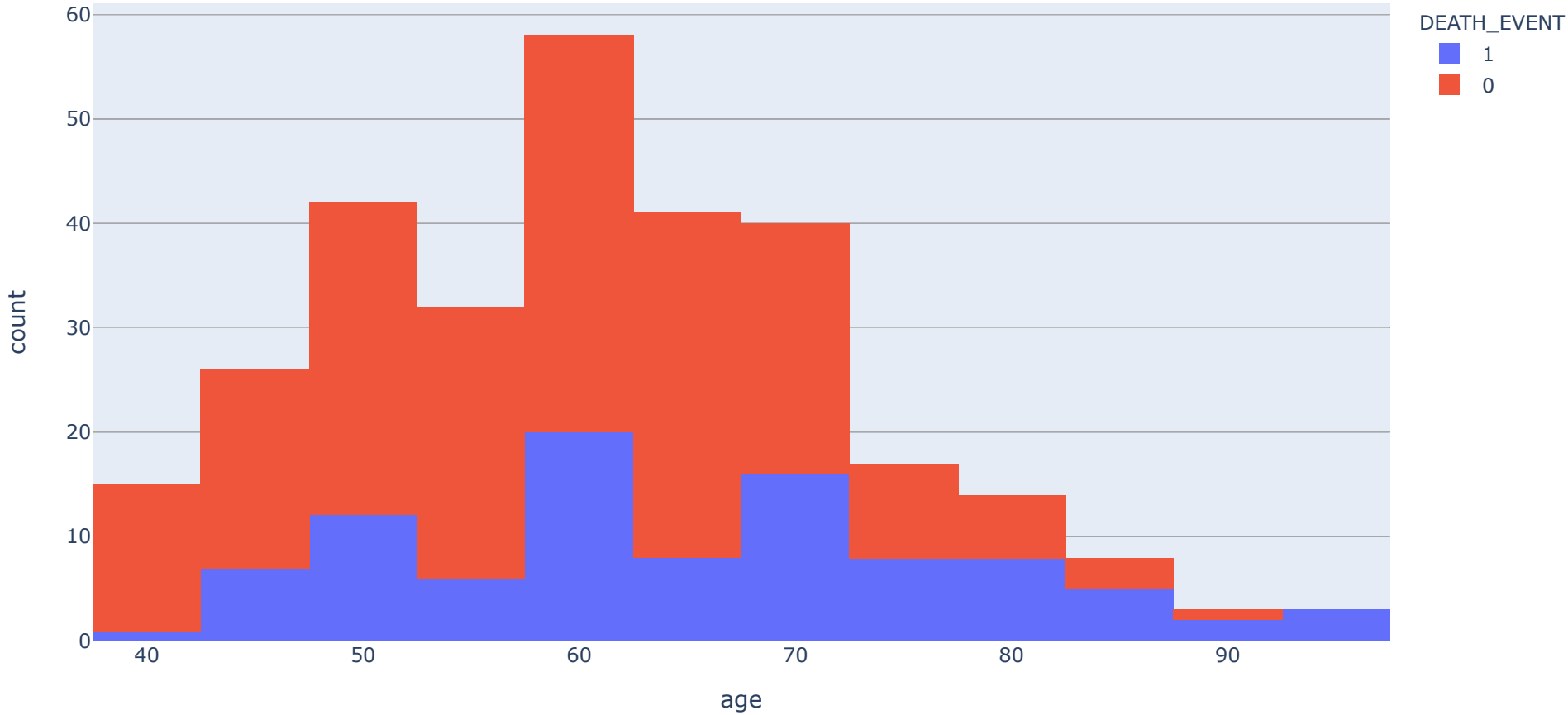


Age Spread Analysis:

- Age distribution is between 40 to 95 years
- The count is maximum for people close to 60 years of age, followed by 70 years
- The count of people is very low for age greater than 80

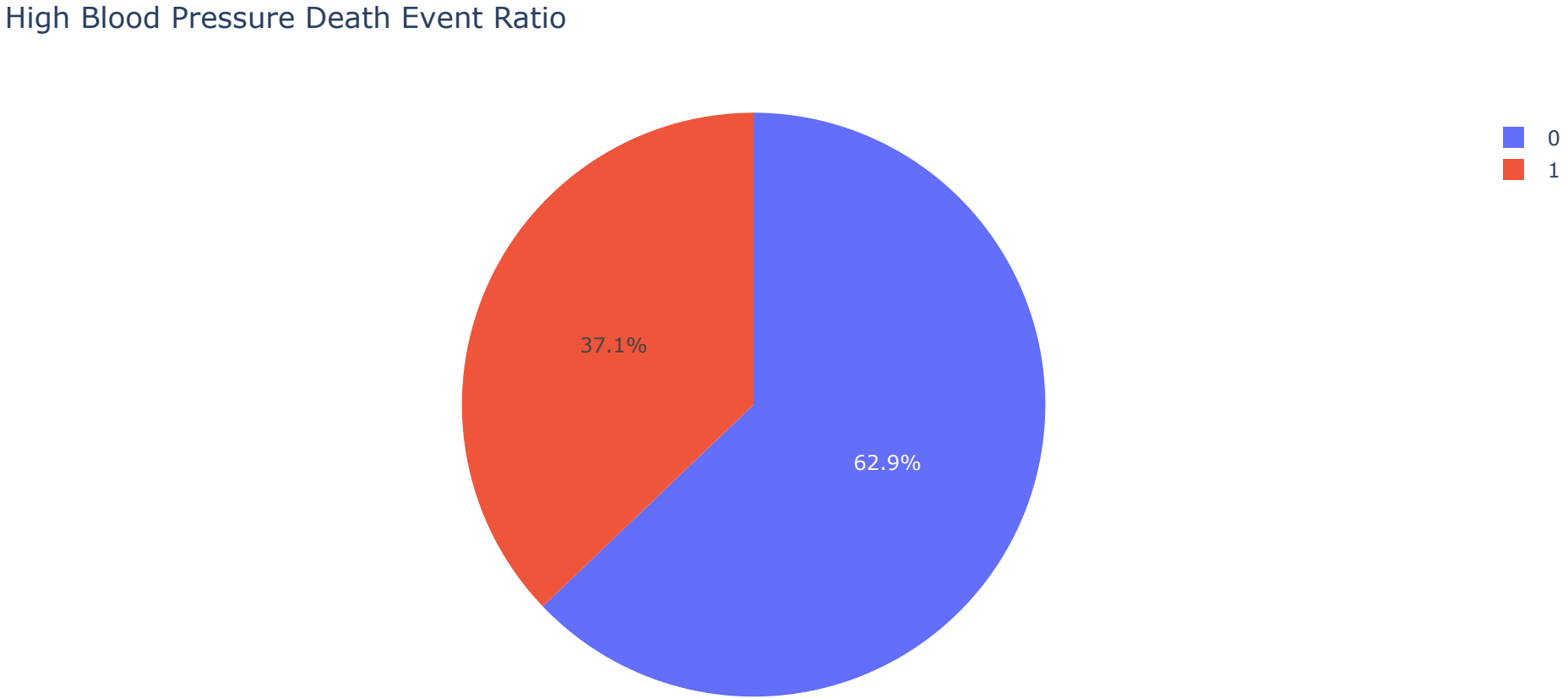
Analysis - Age Vs. Death Event

In [9]:
```
# Plot the analysis of impact of age on death event
fig = px.histogram(patient_data, x="age", color="DEATH_EVENT",  hover_data=patient_data.columns)
fig.show()
```



Analysis - High Blood Pressure Vs. Death Event

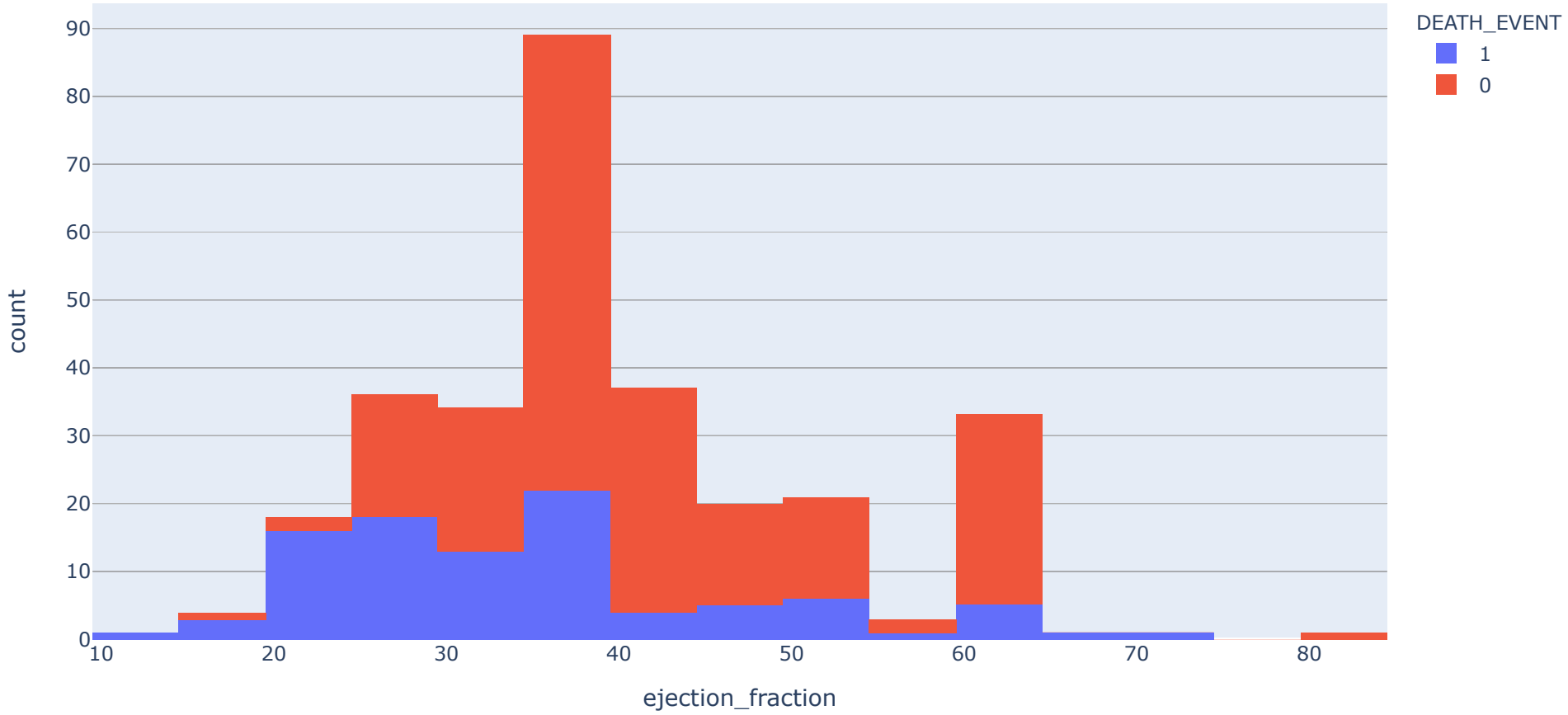
In [10]:
```
# Plot the analysis of impact of High Blood Pressure on death event
fig = px.pie(patient_data, values='high_blood_pressure',names='DEATH_EVENT', title='High Blood Pressure Death Event Ratio')
fig.show()
```



Only 37% of people with High Blood Pressure have died and hence this feature should not be selected for model building.

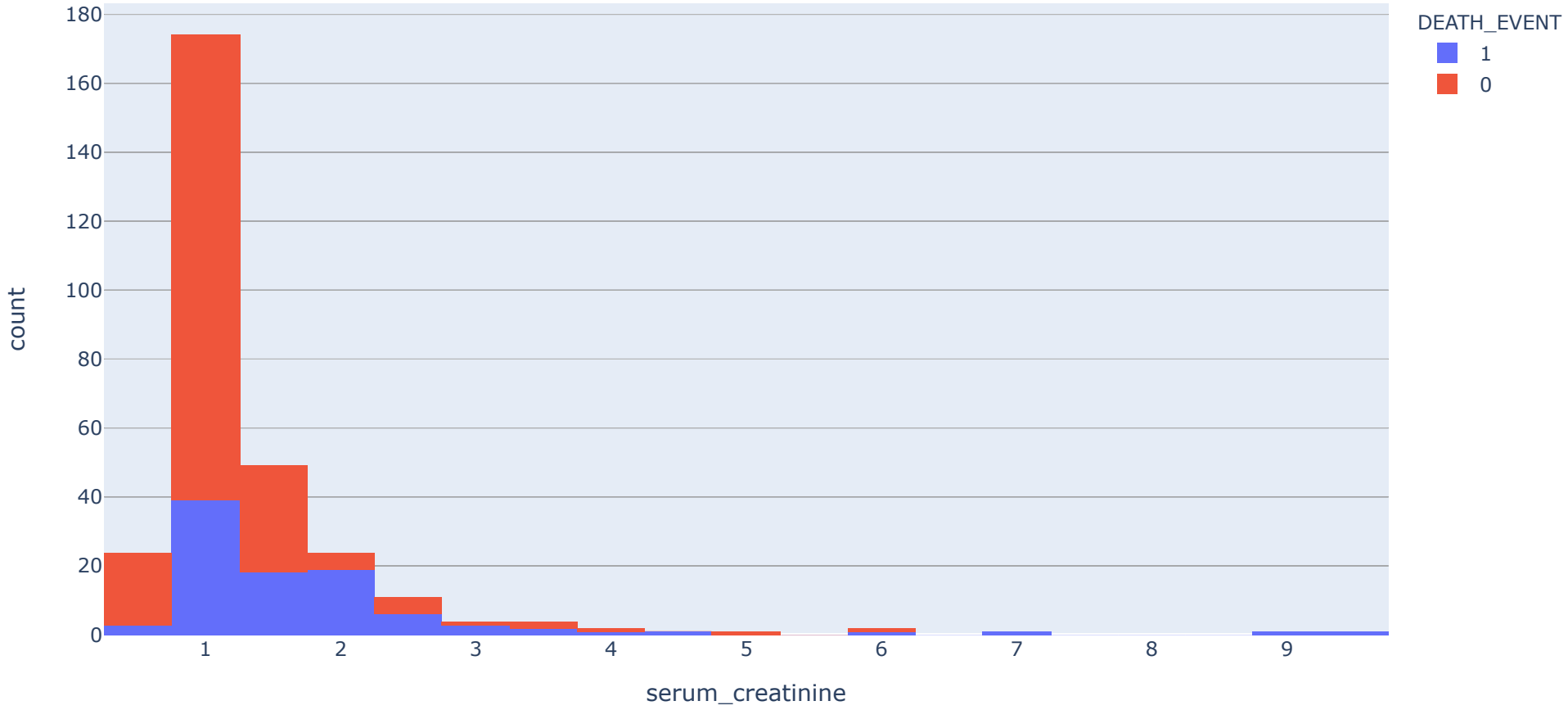
Analysis - Ejection Fraction Vs. Death Event

In [11]:
```
# Plot the analysis of impact of Ejection Fraction on death event
fig = px.histogram(patient_data, x="ejection_fraction", color="DEATH_EVENT",  hover_data=patient_data.columns)
fig.show()
```



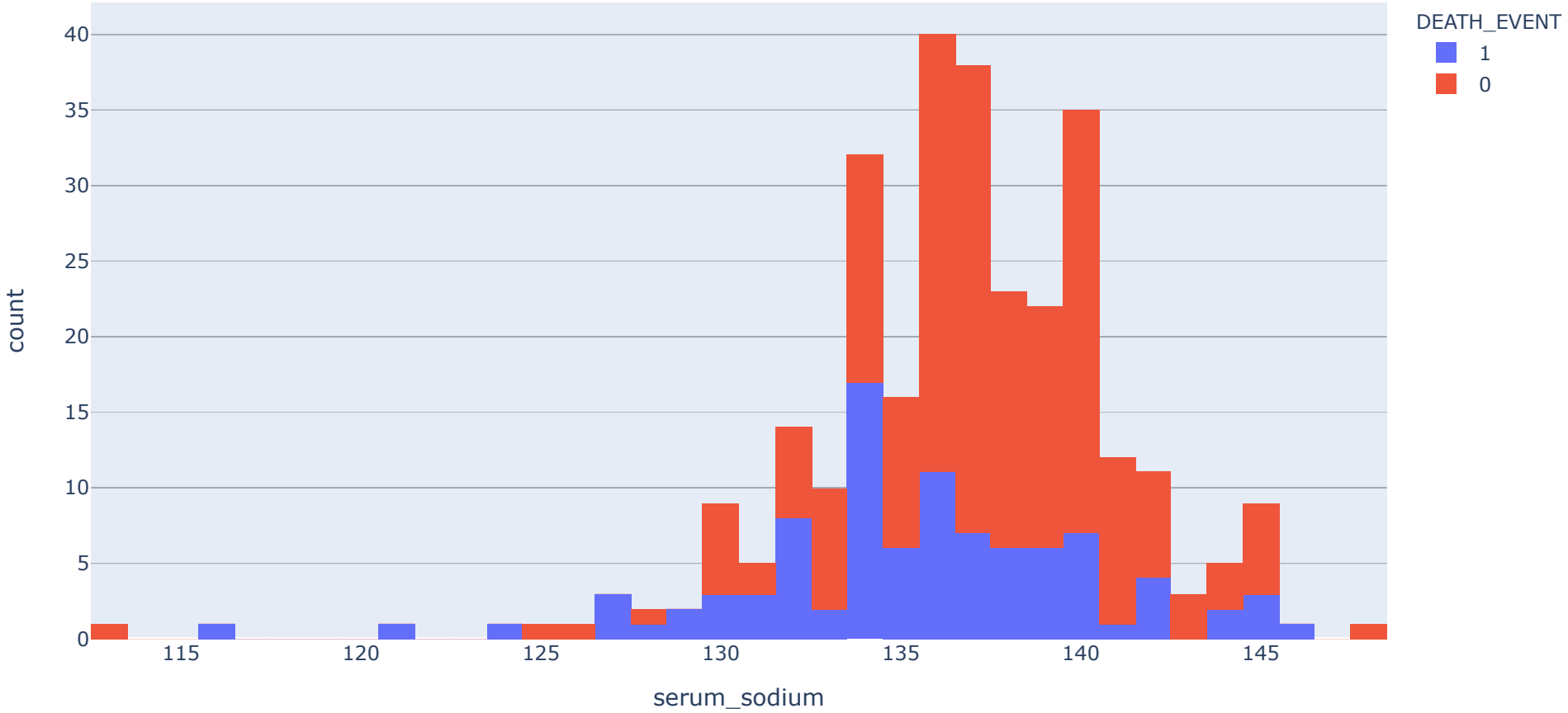
Analysis - Serum Creatinine Vs. Death Event

In [12]:
```
# Plot the analysis of impact of Serum Creatinine on death event
fig = px.histogram(patient_data, x="serum_creatinine", color="DEATH_EVENT", hover_data=patient_data.columns)
fig.show()
```



Analysis - Serum Sodium Vs. Death Event

In [13]:
```
# Plot the analysis of impact of Serum Sodium on death event
fig = px.histogram(patient_data, x="serum_sodium", color="DEATH_EVENT", hover_data=patient_data.columns)
fig.show()
```



Data Cleaning

In [14]:
```
# remove rows that have NULL values
new_patient_data = patient_data.dropna()
print("CLEANING DATAFRAME")
print("Old dataframe length:", len(patient_data))
print("New dataframe length:", len(new_patient_data))
print("Number of rows with at least 1 NA value in movies dataframe: ", (len(patient_data)-len(new_patient_data)))

#remove rows that have negative values of age
new_patient_data = new_patient_data[new_patient_data['age']>0]

#rest the index of the new dataframe with NULL values excluded
new_patient_data.reset_index(inplace = True)
```

```
CLEANING DATAFRAME
Old dataframe length: 299
New dataframe length: 299
Number of rows with at least 1 NA value in movies dataframe:  0
```

Data Modelling

Target is stored in variable 'y'

In [15]:
```
y=new_patient_data[['DEATH_EVENT']].copy()

y.head()
```

Out[15]:

	DEATH_EVENT
0	1
1	1
2	1
3	1
4	1

Feature Selection

In [16]:
```
features = ['age','ejection_fraction','serum_creatinine','serum_sodium']
```

In [17]:
```
X = new_patient_data[features].copy()
```

In [18]:
```
X.columns
```

Out[18]:
```
Index(['age', 'ejection_fraction', 'serum_creatinine', 'serum_sodium'], dtype='object')
```

Split training and test data

In [19]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=324)
```

Decision Tree Classifier

Fit on Train Set

In [20]:
```
DT_death_classifier = DecisionTreeClassifier(max_leaf_nodes=15, max_depth=5, random_state=0)
DT_death_classifier.fit(X_train, y_train)
```

Out[20]:
```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
            max_features=None, max_leaf_nodes=15,
            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 [21]:
```
type(DT_death_classifier)
```

Out[21]:
```
sklearn.tree.tree.DecisionTreeClassifier
```

Predict on Test Set

In [26]:
```
predictions = DT_death_classifier.predict(X_test)
```

In [27]:
```
predictions[:10]
```

Out[27]:
```
array([0, 0, 1, 0, 1, 0, 1, 1, 0, 0], dtype=int64)
```

In [28]:
```
y_test['DEATH_EVENT'][:10]
```

Out[28]:
```
147    0
235    0
11     1
271    0
20     0
27     1
258    0
126    1
296    0
8      1
Name: DEATH_EVENT, dtype: int64
```

In [29]:
```
print("PREDICTION ON TEST DATA")
print("")
print("Accuracy:", accuracy_score(y_true = y_test, y_pred = predictions))
print("Precision:", precision_score(y_test, predictions))
print("Recall:", recall_score(y_test, predictions))
print("F1 Score:", f1_score(y_test, predictions))
```

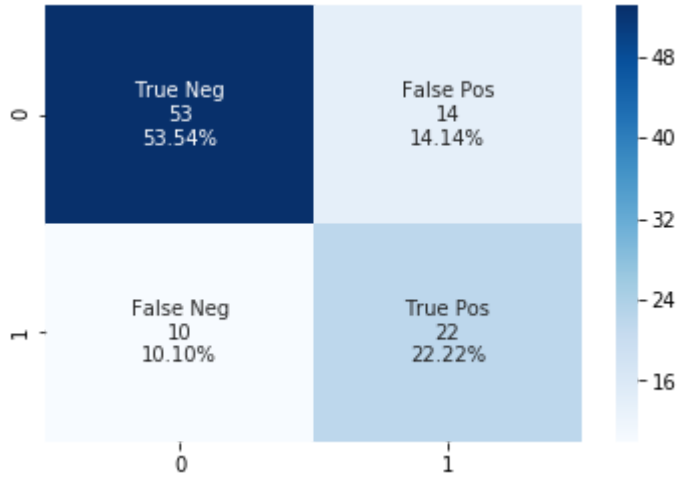
```
PREDICTION ON TEST DATA

Accuracy: 0.7575757575757576
Precision: 0.6111111111111112
Recall: 0.6875
F1 Score: 0.6470588235294118
```

In [30]:
```
conf=confusion_matrix(y_test, predictions)
```

In [31]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                conf.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     conf.flatten()/np.sum(conf)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(conf, annot=labels, fmt='', cmap='Blues')
```

Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x21b8916f390>



Logistic Regression Classifier

Fit on Train Set

In [32]:
```
type(y_train['DEATH_EVENT'])
```

Out[32]: pandas.core.series.Series

In [33]:
```
LR_death_classifier = LogisticRegression()
#LR_death_classifier.fit(X_train, y_train)
LR_death_classifier.fit(X_train, y_train['DEATH_EVENT'])
```

Out[33]:
```
LogisticRegression(C=1.0, class_weight=None, dual=False, fit_intercept=True,
          intercept_scaling=1, max_iter=100, multi_class='ovr', n_jobs=1,
          penalty='l2', random_state=None, solver='liblinear', tol=0.0001,
          verbose=0, warm_start=False)
```

In [34]:
```
type(LR_death_classifier)
```

Out[34]:
```
sklearn.linear_model.logistic.LogisticRegression
```

Predict on Test Set

In [39]:
```
predictions = LR_death_classifier.predict(X_test)
```

In [40]:
```
predictions[:10]
```

Out[40]:
```
array([0, 0, 0, 0, 1, 0, 0, 1, 0, 0], dtype=int64)
```

In [41]:
```
y_test['DEATH_EVENT'][:10]
```

Out[41]:
```
147    0
235    0
11     1
271    0
20     0
27     1
258    0
126    1
296    0
8      1
Name: DEATH_EVENT, dtype: int64
```

In [42]:
```
print("PREDICTION ON TEST DATA")
print("")
print("Accuracy:", accuracy_score(y_true = y_test, y_pred = predictions))
print("Precision:", precision_score(y_test, predictions))
print("Recall:", recall_score(y_test, predictions))
print("F1 Score:", f1_score(y_test, predictions))
```

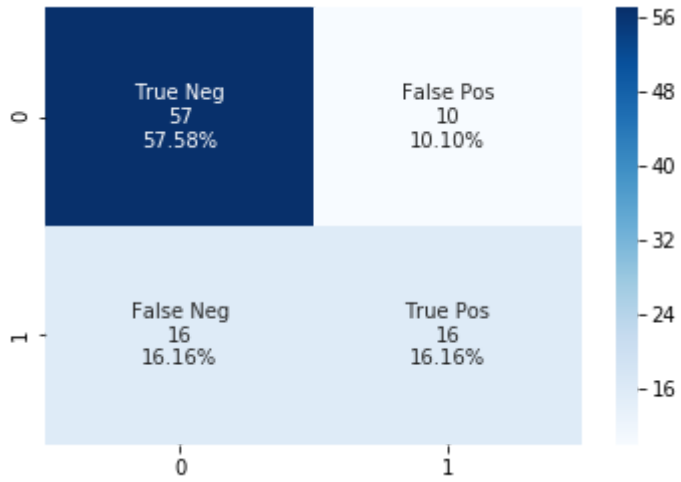
```
PREDICTION ON TEST DATA

Accuracy: 0.7373737373737373
Precision: 0.6153846153846154
Recall: 0.5
F1 Score: 0.5517241379310345
```

In [43]:
```
conf=confusion_matrix(y_test, predictions)
```

In [44]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                conf.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     conf.flatten()/np.sum(conf)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(conf, annot=labels, fmt='', cmap='Blues')
```

Out[44]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b890c17b8>
```



Support Vector Classifier

Fit on Train Set

In [45]:
```
SV_death_classifier = SVC()
SV_death_classifier.fit(X_train, y_train['DEATH_EVENT'])
```

Out[45]:
```
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto', kernel='rbf',
  max_iter=-1, probability=False, random_state=None, shrinking=True,
  tol=0.001, verbose=False)
```

In [46]:
```
type(SV_death_classifier)
```

Out[46]:
```
sklearn.svm.classes.SVC
```

Predict on Test Set

In [51]:
```
predictions = SV_death_classifier.predict(X_test)
```

In [52]:
```
predictions[:10]
```

Out[52]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0], dtype=int64)
```

In [53]:
```
y_test['DEATH_EVENT'][:10]
```

Out[53]:
```
147    0
235    0
11     1
271    0
20     0
27     1
258    0
126    1
296    0
8      1
Name: DEATH_EVENT, dtype: int64
```

In [54]:
```
print("PREDICTION ON TEST DATA")
print("")
print("Accuracy:", accuracy_score(y_true = y_test, y_pred = predictions))
print("Precision:", precision_score(y_test, predictions))
print("Recall:", recall_score(y_test, predictions))
print("F1 Score:", f1_score(y_test, predictions))
```

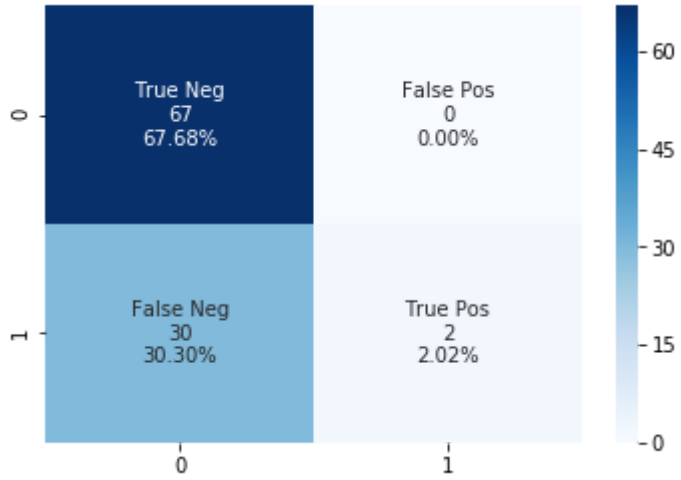
```
PREDICTION ON TEST DATA

Accuracy: 0.696969696969697
Precision: 1.0
Recall: 0.0625
F1 Score: 0.11764705882352941
```

In [55]:
```
conf=confusion_matrix(y_test, predictions)
```

In [56]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                conf.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     conf.flatten()/np.sum(conf)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(conf, annot=labels, fmt='', cmap='Blues')
```

Out[56]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b89211d30>
```



K Neighbors Classifier

Fit on Train Set

In [57]:
```
KN_death_classifier = KNeighborsClassifier(n_neighbors=6)
KN_death_classifier.fit(X_train, y_train['DEATH_EVENT'])
```

Out[57]:
```
KNeighborsClassifier(algorithm='auto', leaf_size=30, metric='minkowski',
           metric_params=None, n_jobs=1, n_neighbors=6, p=2,
           weights='uniform')
```

In [58]:
```
type(KN_death_classifier)
```

Out[58]:
```
sklearn.neighbors.classification.KNeighborsClassifier
```

Predict on Test Set

In [67]:
```
predictions = KN_death_classifier.predict(X_test)
```

In [ ]:
```
predictions[:10]
```

In [68]:
```
y_test['DEATH_EVENT'][:10]
```

Out[68]:
```
147    0
235    0
11     1
271    0
20     0
27     1
258    0
126    1
296    0
8      1
Name: DEATH_EVENT, dtype: int64
```

In [69]:
```
print("PREDICTION ON TEST DATA")
print("")
print("Accuracy:", accuracy_score(y_true = y_test, y_pred = predictions))
print("Precision:", precision_score(y_test, predictions))
print("Recall:", recall_score(y_test, predictions))
print("F1 Score:", f1_score(y_test, predictions))
```

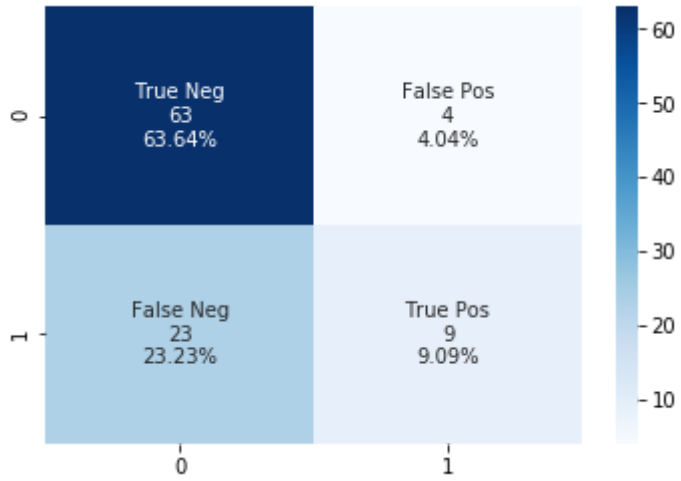
```
PREDICTION ON TEST DATA

Accuracy: 0.7272727272727273
Precision: 0.6923076923076923
Recall: 0.28125
F1 Score: 0.4
```

In [70]:
```
conf=confusion_matrix(y_test, predictions)
```

In [71]:
```
group_names = ['True Neg','False Pos','False Neg','True Pos']
group_counts = ["{0:0.0f}".format(value) for value in
                conf.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                     conf.flatten()/np.sum(conf)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
          zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(conf, annot=labels, fmt='', cmap='Blues')
```

Out[71]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x21b89609ba8>
```



Conclusion

The death of a patient due to heart failure seems to depend predominantly on age and clinical features of ejection fraction, serum creatinine and serum sodium. Blood pressure does not have a strong effect on its own but may influence in combination with other factors. The best machine learning model for predicting heart failure is the Decision Tree classifier. In summary, it is possible to predict heart failure using a machine learning model upto an accuracy of around 76%.