I. Problem Statement

According to the CDC, heart failure in the United States affects about 6.2 million people, and almost 1.4 million are under 60 years of age. It is a serious condition that costs billions of dollars a year in treatment. More than half of those who develop congestive heart failure die within 5 years of diagnosis. There are factors that can increase risk of heart failure, most notably, smoking tobacco and poor diet and exercise.

The goal of this project would be to create a model that could help predict heart failure based on a number of features that can help with early detection. Furthermore, the model will also help determine which features are more determinant.

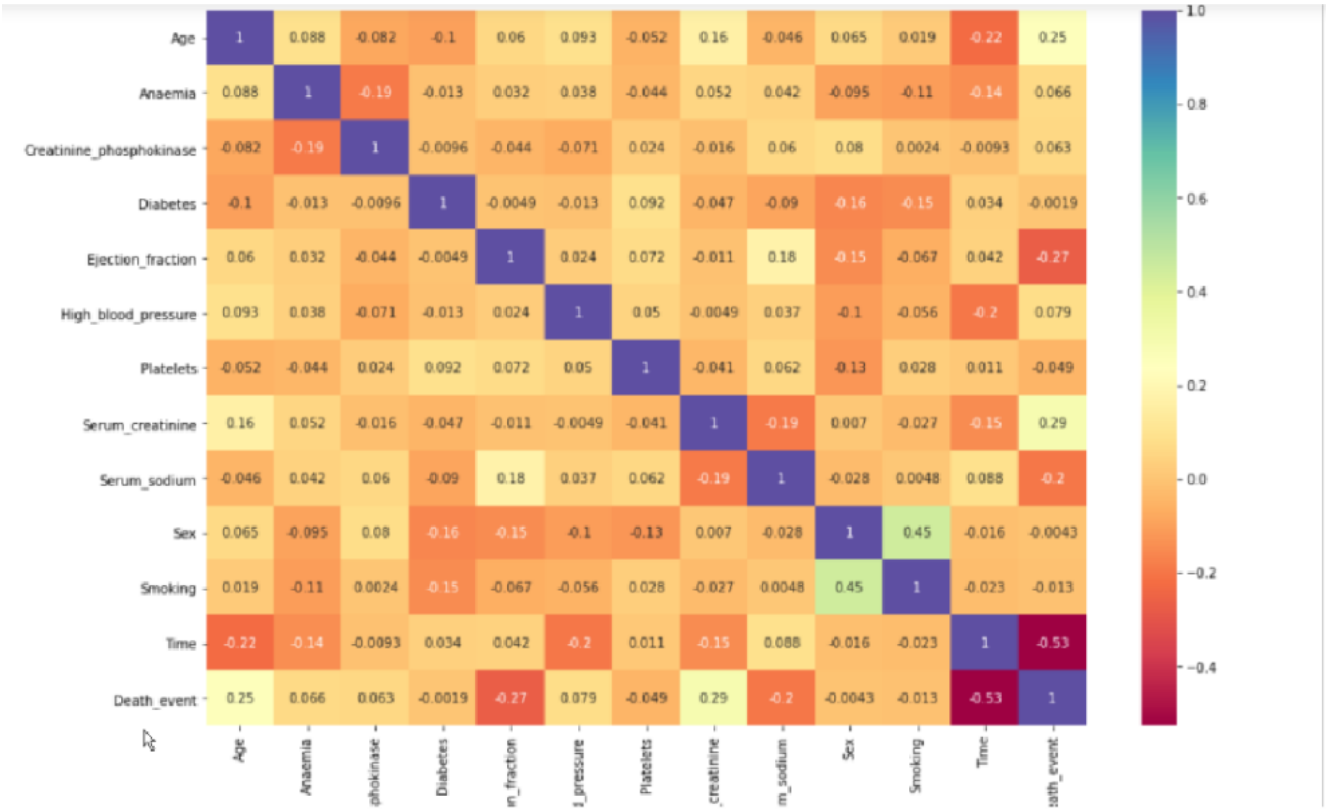
II. Data Wrangling

The dataset was retrieved from the website Kaggle.com. The dataset is free to use and can be downloaded in cvs format. The dataset itself is very simple. It only contains 299 entries and 13 columns. This is one of the reasons why this dataset was chosen, as it would be quick to look through the rows. However, this comes at the cost of effectively predicting with our model

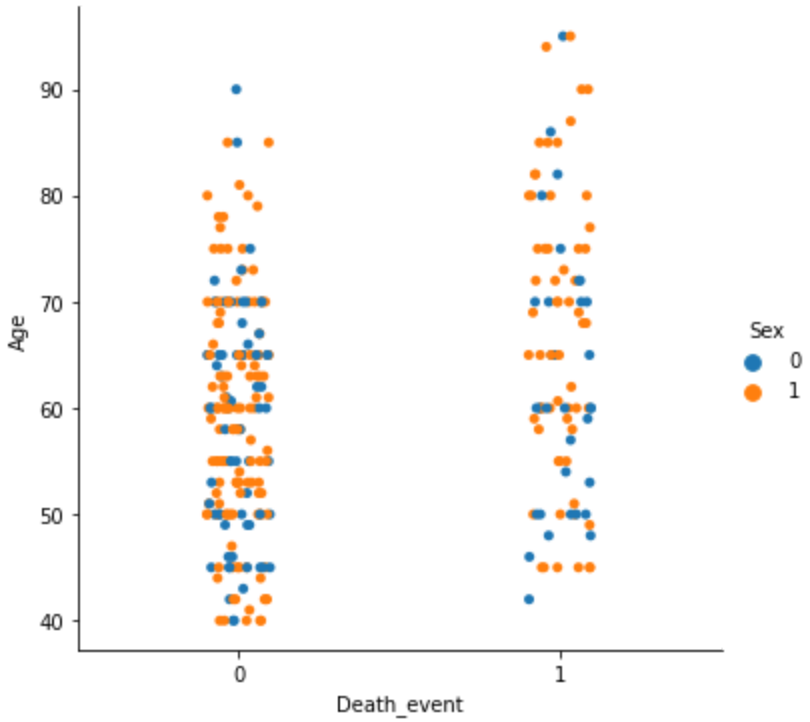
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    int64
 2   creatinine_phosphokinase  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    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    int64
 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
dtypes: float64(3), int64(10)
memory usage: 30.5 KB
```

Another advantage to this dataset is that there did not exist any null values and no duplicates, so I did not have to omit any rows. The only change I made was rewriting the columns names for readability.

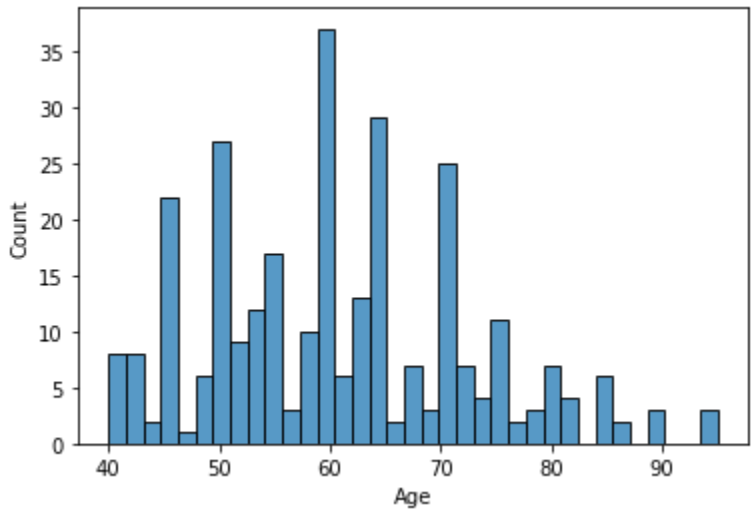
III. Exploratory Data Analysis



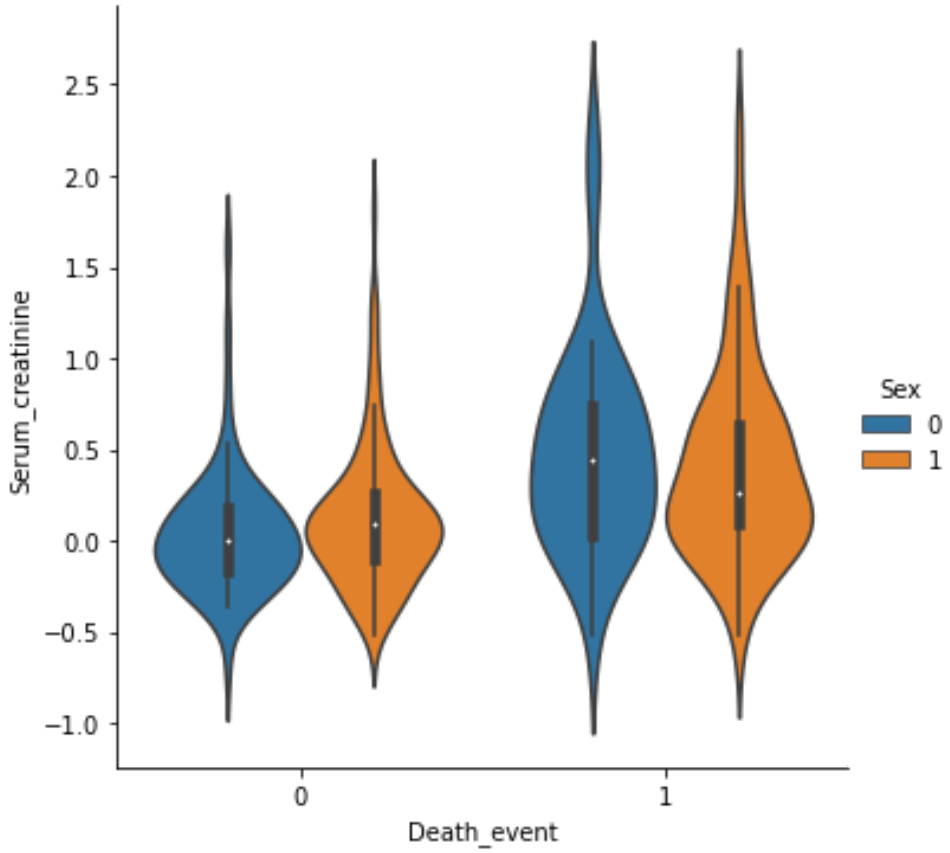
The first for my EDA was to create a heatmap to view the correlation between the features. The scale on the right shows the measurement scale. The higher the number, the stronger the correlation. The noticeable correlations include Death_event - Serum_creatinine, Death_event - Age, Death_event - high_blood_pressure, and Smoking - Sex. Even though the Smoking - Sex has a high correlation, we are concerned with the occurrence of death, so it will not be looked into.



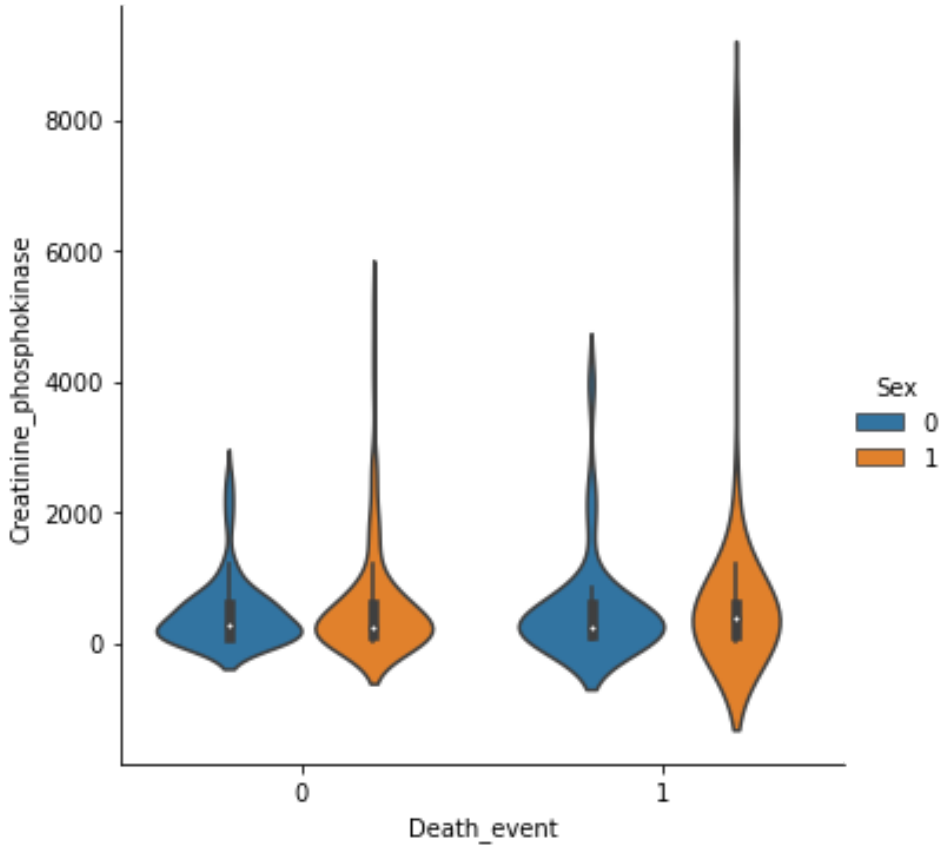
The first one viewed was the relationship between age and death_event. In this instance, a death event of 1 is associated with heart failure. This swarmplot is also showing the distribution of male or female. 1 is associated with male and 0 is associated with female. Although the data is fewer for death event of 1, one can see a shift up with the age spread.



This histogram helps get a better understanding of the distribution for Age.



The following shows the relationship between Death event and Serum creatinine. It seems that that the majority of people who experienced heart failure had a level of Serum creatinine just above 0 in their blood.



The last correlation mentioned is the relationship between Creatinine phosphokinase and Death event. Like Serum creatinine, the plot shows that the Level of the CPK enzyme in the blood for heart failure is close to zero. It is notable to mention however that there exists a skewness. For the purposes of the models I chose, this is not a concern.

IV. Model Selection

The first step was to scale the data to prepare for the models. This was done using StandardScaler. For my model selection process, I began by using the library LazyClassifier to get an idea of how a number of machine learning models would perform without any tuning.

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
LinearSVC	0.79	0.76	0.76	0.78	0.01
LogisticRegression	0.79	0.76	0.76	0.78	0.01
NearestCentroid	0.77	0.75	0.75	0.76	0.01
XGBClassifier	0.77	0.74	0.74	0.76	1.16
CalibratedClassifierCV	0.78	0.74	0.74	0.76	0.03
RandomForestClassifier	0.77	0.74	0.74	0.76	0.11
ExtraTreesClassifier	0.77	0.72	0.72	0.75	0.09
LinearDiscriminantAnalysis	0.76	0.72	0.72	0.74	0.02
LGBMClassifier	0.74	0.72	0.72	0.74	0.03
RidgeClassifierCV	0.76	0.72	0.72	0.74	0.01
BaggingClassifier	0.74	0.72	0.72	0.74	0.02
RidgeClassifier	0.74	0.71	0.71	0.73	0.02
GaussianNB	0.73	0.70	0.70	0.72	0.01
SGDClassifier	0.72	0.70	0.70	0.72	0.01
SVC	0.73	0.70	0.70	0.72	0.01
AdaBoostClassifier	0.71	0.67	0.67	0.69	0.07
BernoulliNB	0.71	0.67	0.67	0.69	0.01
NuSVC	0.71	0.67	0.67	0.69	0.01
PassiveAggressiveClassifier	0.69	0.66	0.66	0.68	0.01
ExtraTreeClassifier	0.68	0.66	0.66	0.67	0.01
QuadraticDiscriminantAnalysis	0.69	0.65	0.65	0.67	0.01
DecisionTreeClassifier	0.68	0.65	0.65	0.67	0.01
Perceptron	0.67	0.63	0.63	0.65	0.01
KNeighborsClassifier	0.68	0.62	0.62	0.63	0.01
LabelPropagation	0.64	0.59	0.59	0.61	0.01
LabelSpreading	0.64	0.59	0.59	0.61	0.01
DummyClassifier	0.62	0.59	0.59	0.60	0.01

The goal here would be to see if there are any big differences in the F1 score to omit those models. I decided to not consider any models that are under .72. The models above should be able to achieve similar F1 scores after tuning. The three machine learning classification models used: Logistic Regression, Random Forest Classifier, XGBoost. The metric I focused on was the F1 score. Furthermore, hyperparameter tuning for Random Forest Classifier and XGBoost was used with RandomizedSearchCV to find the best parameters. The following shows the classification reports for the Random Forest, Logistic Regression, and XGBoost models respectively.

In [31]:
```
from sklearn.metrics import classification_report
print(classification_report(y_pred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.94      0.72      0.82        69
           1       0.49      0.86      0.62        21

    accuracy                           0.76        90
   macro avg       0.71      0.79      0.72        90
weighted avg       0.84      0.76      0.77        90
```

	precision	recall	f1-score	support
0	0.94	0.76	0.84	66
1	0.57	0.88	0.69	24
accuracy			0.79	90
macro avg	0.76	0.82	0.76	90
weighted avg	0.84	0.79	0.80	90

	precision	recall	f1-score	support
0	0.89	0.77	0.82	61
1	0.62	0.79	0.70	29
accuracy			0.78	90
macro avg	0.75	0.78	0.76	90
weighted avg	0.80	0.78	0.78	90

V. Conclusion

According to these reports, the F1 score shows that Linear regression performed the best, even though it required no tuning. Also, both Random Forest and XGBoost performed the same. The takeaway from this is that the dataset is simple enough to predict with the scalar response (death_event) and the explanatory variables. Further research examined and concluded that Serum_creatinine and Time were the most predictive features for the RandomForestClassifier model, and time alone for XGBoost.

In [46]:
```
for feature_list_index in sfm.get_support(indices=True):
    print(feat_labels[feature_list_index])
```

```
Serum_creatinine
Time
```

```
In [48]: for feature_list_index in sfm.get_support(indices=True):
             print(feat_labels[feature_list_index])

Time
```

It is important to reiterate that this dataset was small, thus predictive power was limited. If more data was provided, it could provide a more complete picture. The models that I chose aren't necessarily optimized. The hyperparameter tuning could be more extensive at the cost of computational time. Other models could be explored as well. This research is useful for individuals in preventing the risk of heart failure by identifying what features contribute to it.