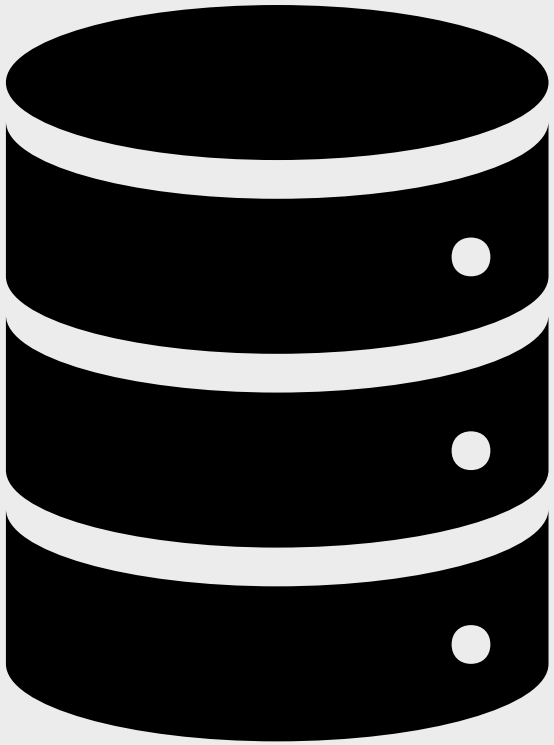
DATA VISUALIZATION

mit Python



- **Alter**: Alter der Person
- **Anämie**: Abnahme der roten Blutkörperchen oder des Hämoglobins
- **Kreatinphosphokinase**: Der Spiegel des CPK-Enzyms im Blut (mcg/l)
- **Diabetes**: Wenn der Patient Diabetes hat
- **ejection_fraction**: Der Prozentsatz (Prozent) des Bluts, das bei jeder Kontraktion aus dem Herzen austritt
- **high_blood_pressure**: Wenn der Patient an Bluthochdruck leidet
- **Blutplättchen**: Blutplättchen im Blut (Kiloplättchen/ml)
- **serum_creatinin**: Serumkreatininspiegel im Blut (mg/dl)
- **serum_sodium**: Serumnatriumspiegel im Blut (mEq/L)
- **Geschlecht**: weiblich oder männlich
- **Rauchen**: Wenn der Patient raucht oder nicht raucht
- **Dauer**: Nachbeobachtungszeit (Tage)

```python
# Pandas ist eine der Open-Source-Python-Bibliotheken,
# die benutzerfreundliche Datenstrukturen bieten und Datenanalysen ermöglichen.
import pandas as pd

# NumPy (Numerical Python) ist eine Mathematikbibliothek,
# mit der wir schnell wissenschaftliche Berechnungen durchführen können.
import numpy as np

# Python Matplotlib; matplotlib.pyplot ist eine Python-Bibliothek, die für 2D- oder 3D-Grafiken verwendet wird.
import matplotlib.pyplot as plt

# Seaborn ist eine Bibliothek in Python, mit der interessante und informative statistische Grafiken erstellt werden.
import seaborn as sns

import warnings

# Es ist ein Grafik-Erstellungsmodul.
import plotly.express as px

import plotly.figure_factory as ff

import plotly.graph_objs as go
```

```
# Wir importieren unseren Datensatz.
dataset = pd.read_csv(r"C:\Users\ceren\Desktop\datasets_727551_1263738_heart_failure_clinical_records_dataset.csv")

# Wir drucken die ersten fünf Zeilen des Datensatzes.
dataset.head()
```

Out[4]:

ıge	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_I
5.0	0	582	0	20	1	265000.00	1.9	130	1	0	4	
5.0	0	7861	0	38	0	263358.03	1.1	136	1	0	6	
5.0	0	146	0	20	0	162000.00	1.3	129	1	1	7	
0.0	1	111	0	20	0	210000.00	1.9	137	1	0	7	
5.0	1	160	1	20	0	327000.00	2.7	116	0	0	8	

In [5]:
```
# Wir betrachten die Größe des Datensatzes.
dataset.shape
```

Out[5]:
```
(299, 13)
```

In [6]:
```
# Wir prüfen, ob unser Datensatz Nullwerte enthält und kontrollieren unser Datentype.
dataset.info()
```

```
<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: 30.4 KB
```

In [7]:
```
to_update={'age':float, 'anaemia':bool, 'creatinine_phosphokinase':int, 'diabetes':bool,
       'ejection_fraction':int, 'high_blood_pressure':bool, 'platelets':float,
       'serum_creatinine':float, 'serum_sodium':float, 'sex':bool, 'smoking':bool, 'time':int,
       'DEATH_EVENT':bool}

dataset=dataset.astype(to_update)
dataset.info()
```

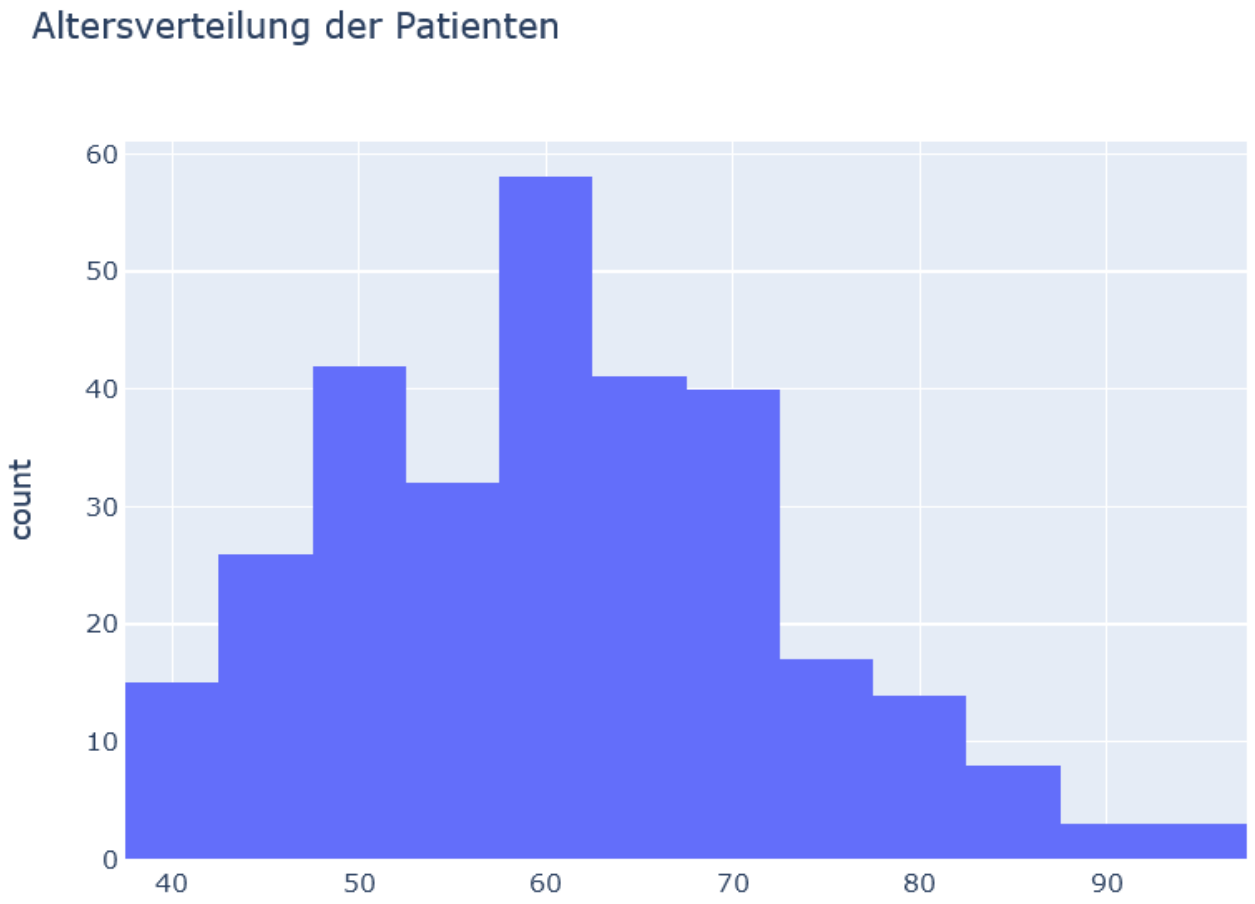
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 299 entries, 0 to 298
Data columns (total 13 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   age                       299 non-null    float64
 1   anaemia                   299 non-null    bool
 2   creatinine_phosphokinase  299 non-null    int32
 3   diabetes                  299 non-null    bool
 4   ejection_fraction         299 non-null    int32
 5   high_blood_pressure       299 non-null    bool
 6   platelets                 299 non-null    float64
 7   serum_creatinine          299 non-null    float64
 8   serum_sodium              299 non-null    float64
 9   sex                       299 non-null    bool
 10  smoking                   299 non-null    bool
 11  time                      299 non-null    int32
 12  DEATH_EVENT               299 non-null    bool
dtypes: bool(6), float64(4), int32(3)
memory usage: 14.7 KB
```

In [8]:
```
dataset.describe()
```

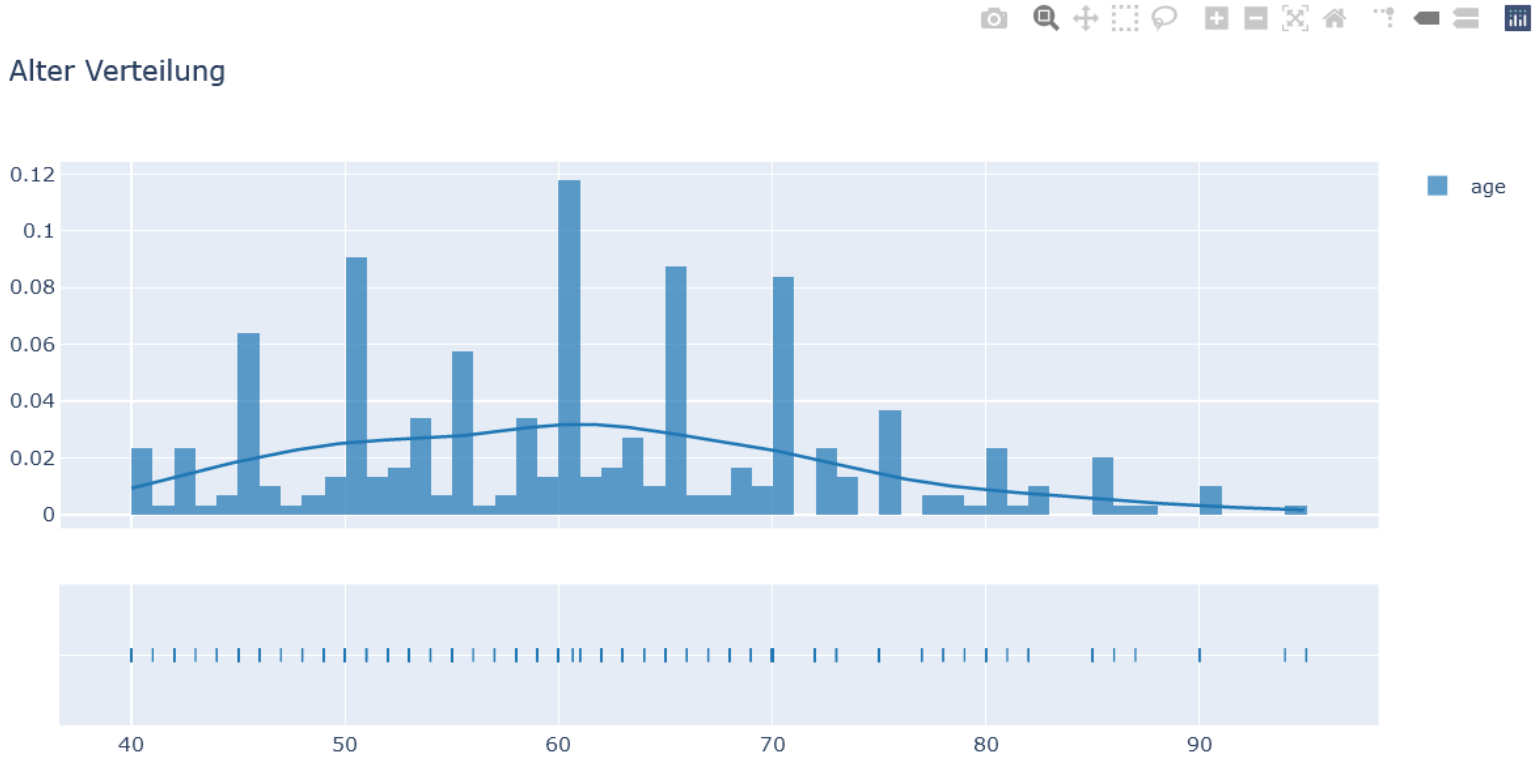
Out[8]:

	age	creatinine_phosphokinase	ejection_fraction	platelets	serum_creatinine	serum_sodium	time
count	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.000000
mean	60.833893	581.839465	38.083612	263358.029264	1.39388	136.625418	130.260870
std	11.894809	970.287881	11.834841	97804.236869	1.03451	4.412477	77.614208
min	40.000000	23.000000	14.000000	25100.000000	0.50000	113.000000	4.000000
25%	51.000000	116.500000	30.000000	212500.000000	0.90000	134.000000	73.000000
50%	60.000000	250.000000	38.000000	262000.000000	1.10000	137.000000	115.000000
75%	70.000000	582.000000	45.000000	303500.000000	1.40000	140.000000	203.000000
max	95.000000	7861.000000	80.000000	850000.000000	9.40000	148.000000	285.000000

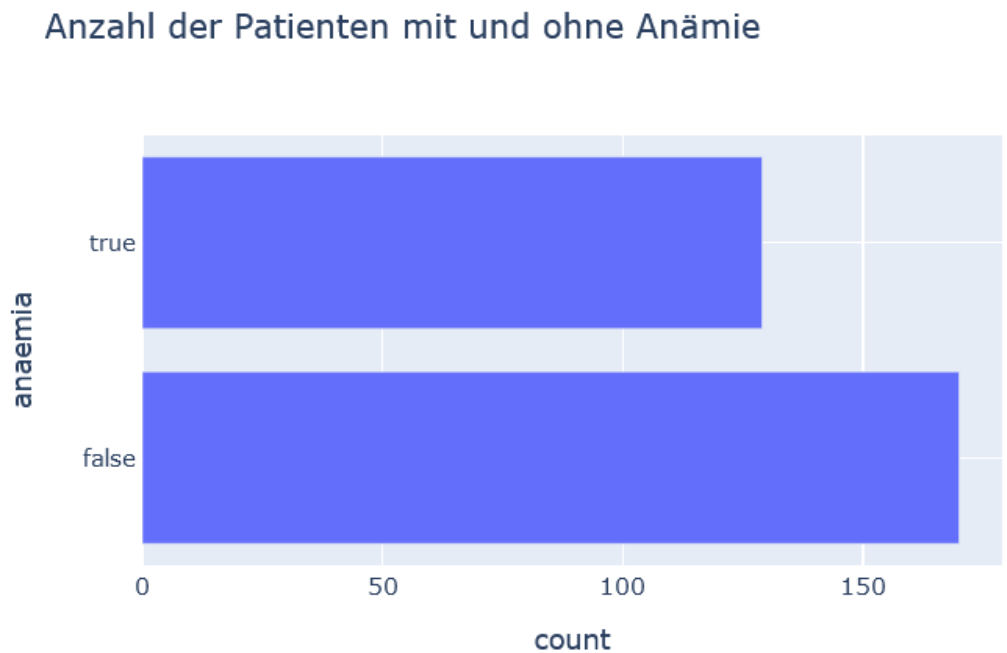
In [9]:
```
fig = px.histogram(dataset, "age", nbins=25, title='Altersverteilung der Patienten', width=700)
fig.show()
```



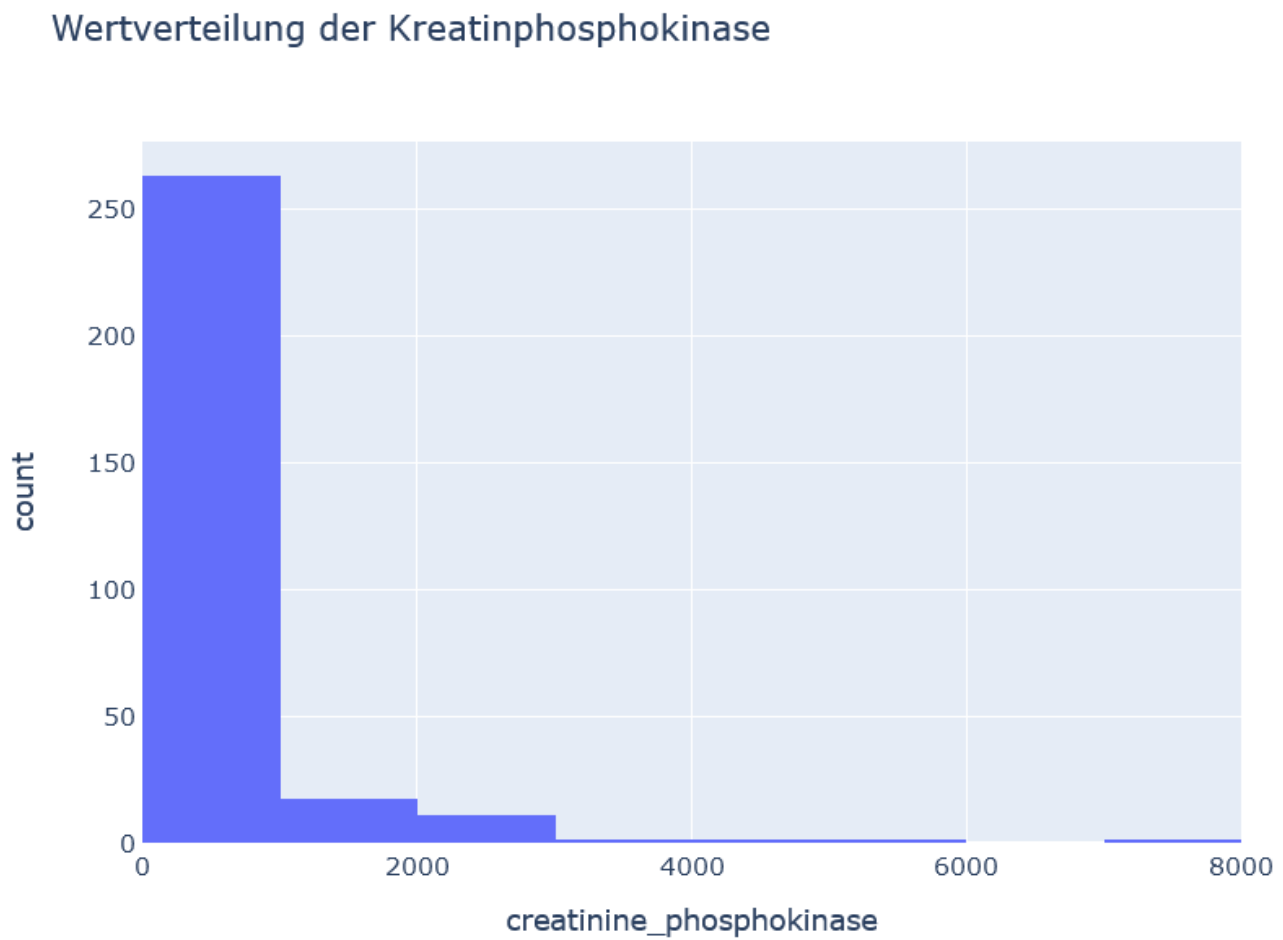
In [9]:
```
hist_data =[dataset["age"].values]
group_labels = ['age']
fig = ff.create_distplot(hist_data, group_labels)
fig.update_layout(title_text='Alter Verteilung')
fig.show()
```



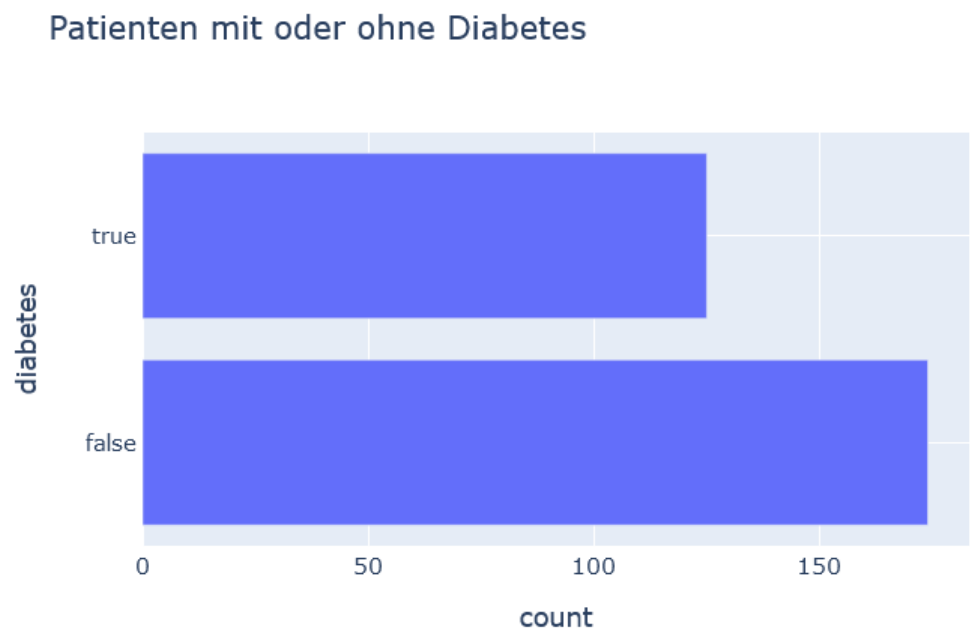
In [10]:
```
ds = dataset['anaemia'].value_counts().reset_index()
ds.columns = ['anaemia', 'count']
fig = px.bar(ds, x='count', y="anaemia", orientation='h',
             title='Anzahl der Patienten mit und ohne Anämie', width=600, height=400)
fig.show()
```



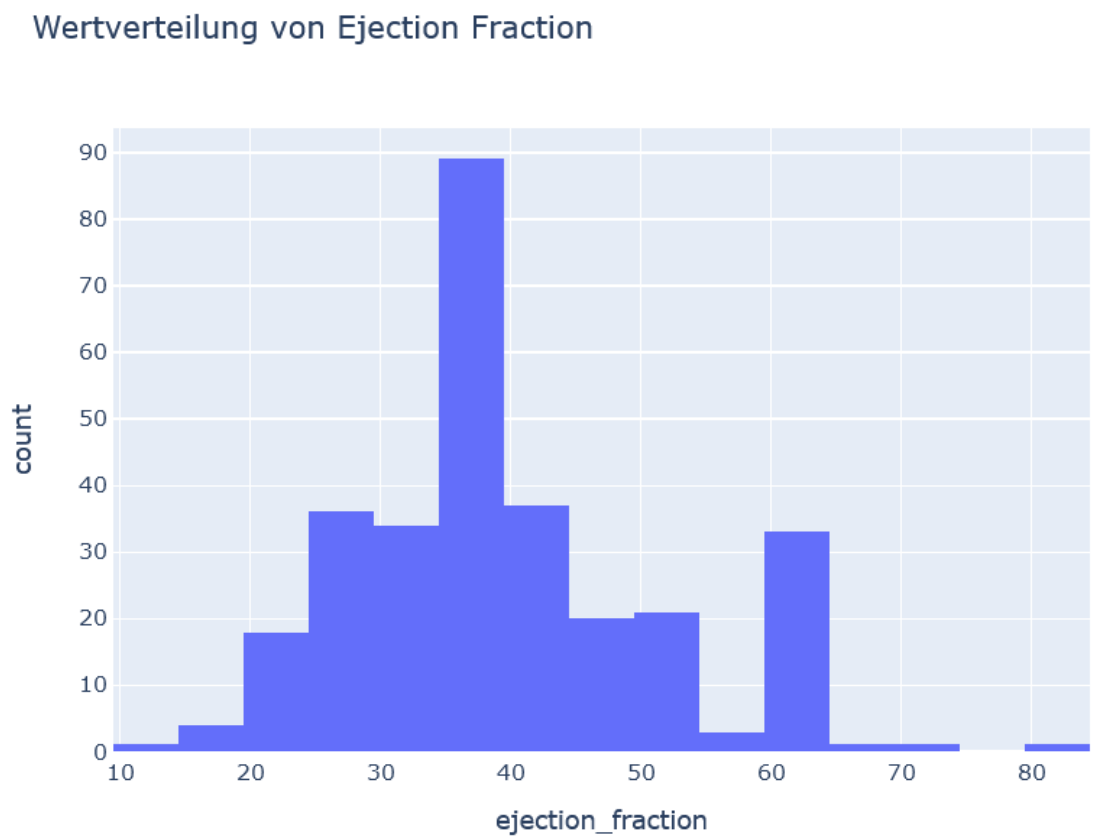
In [11]:
```
fig = px.histogram(dataset, "creatinine_phosphokinase", nbins=15,
                   title='Wertverteilung der Kreatinphosphokinase', width=700)
fig.show()
```



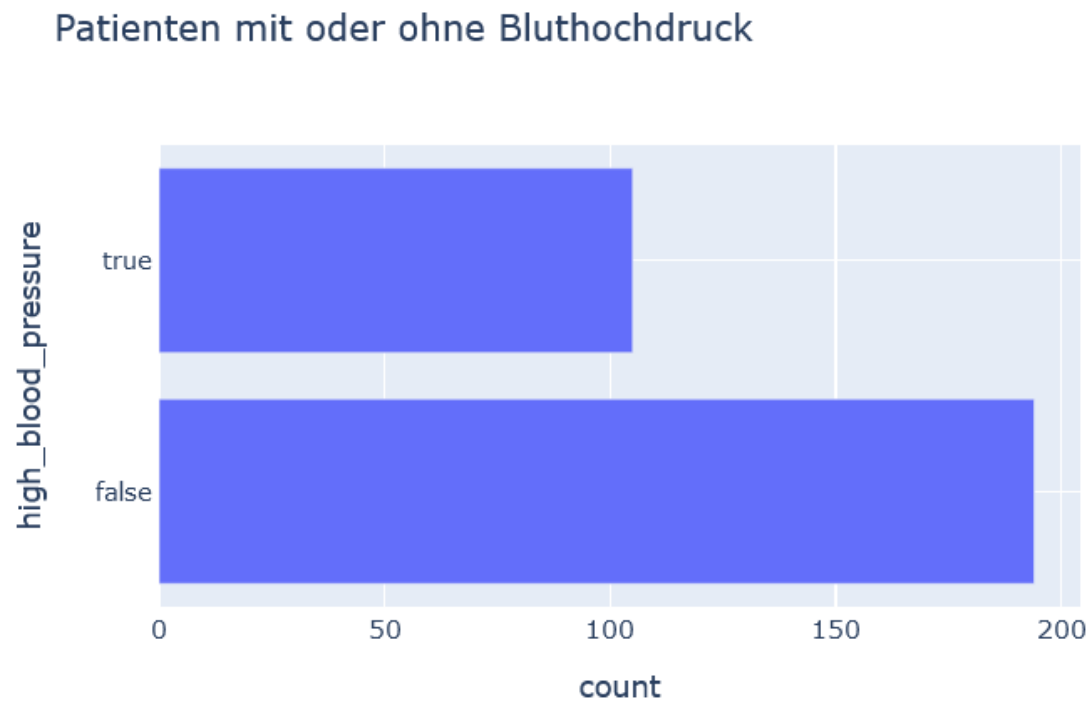
In [12]:
```
ds = dataset['diabetes'].value_counts().reset_index()
ds.columns = ['diabetes', 'count']
fig = px.bar(ds, x='count', y="diabetes", orientation='h',
             title='Patienten mit oder ohne Diabetes', width=600, height=400)
fig.show()
```



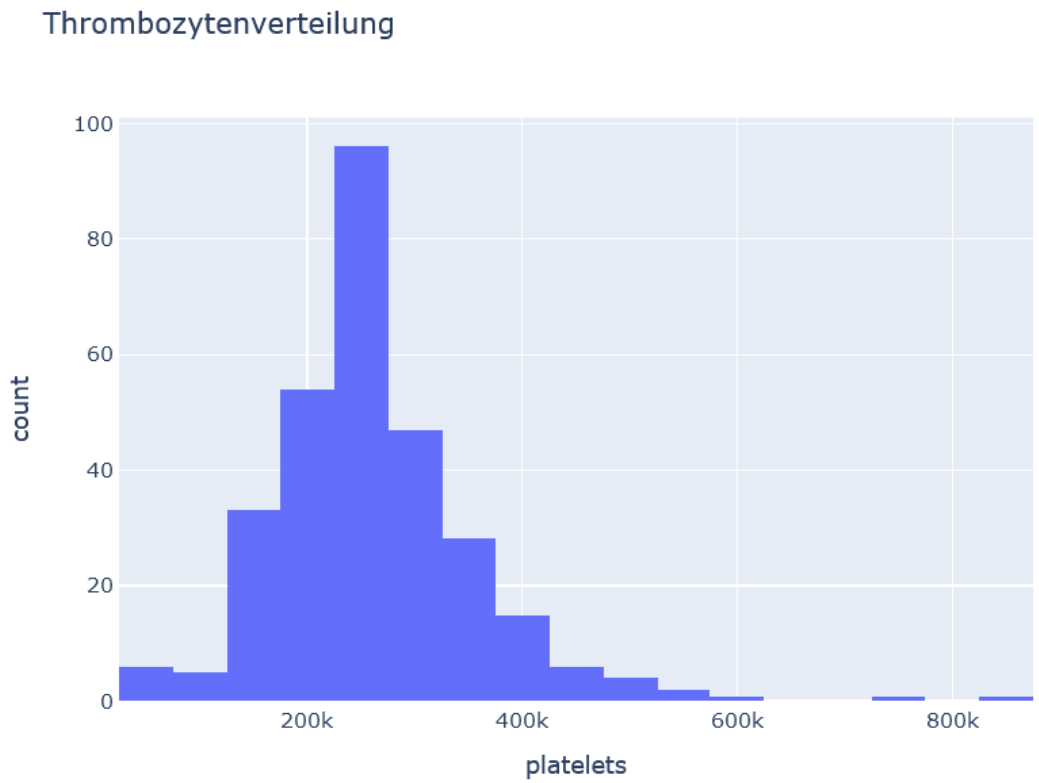
In [13]:
```
fig = px.histogram(dataset, "ejection_fraction", nbins=15, title='Wertverteilung von Ejection Fraction', width=700)
fig.show()

#50-70 Normalbereich, weniger als 40 Herzinsuffizienz
```



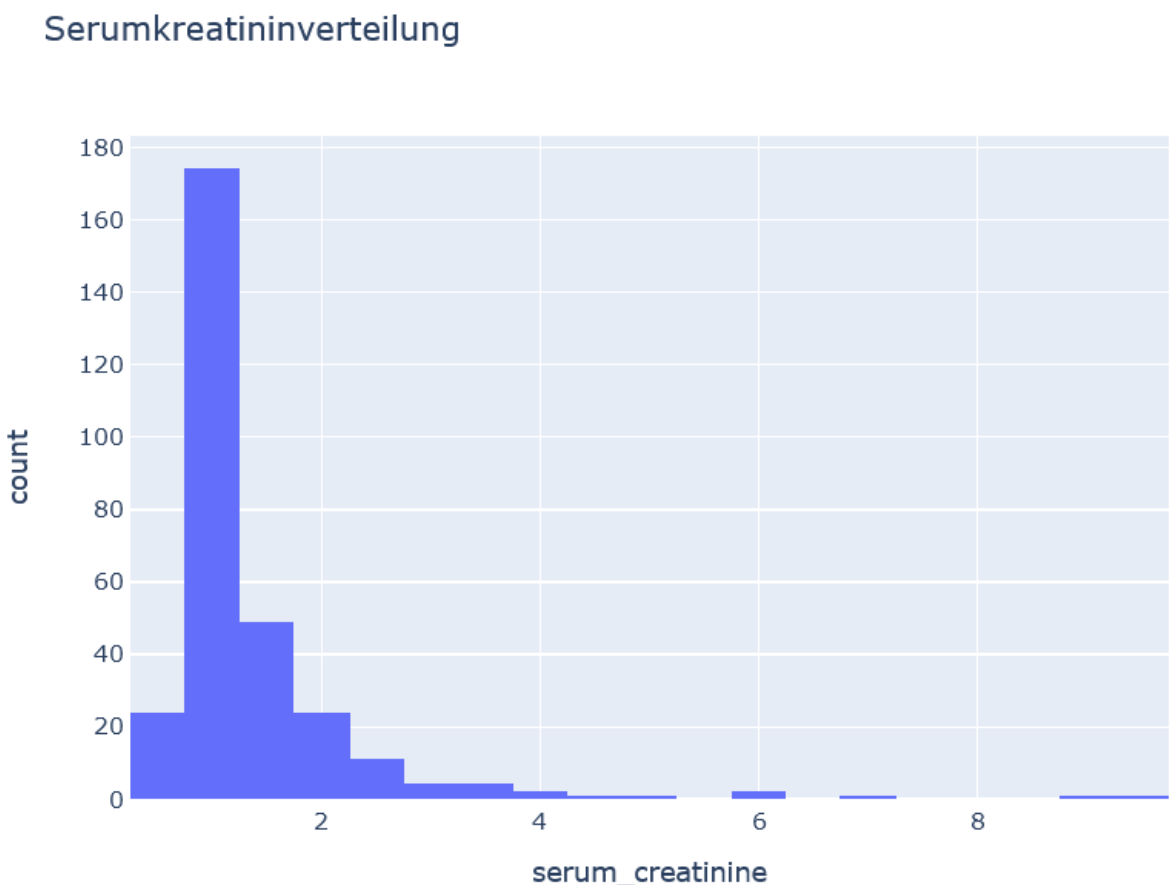
In [14]:
```
ds = dataset['high_blood_pressure'].value_counts().reset_index()
ds.columns = ['high_blood_pressure', 'count']
fig = px.bar(ds, x='count', y="high_blood_pressure", orientation='h',
             title='Patienten mit oder ohne Bluthochdruck', width=600, height=400)
fig.show()
```



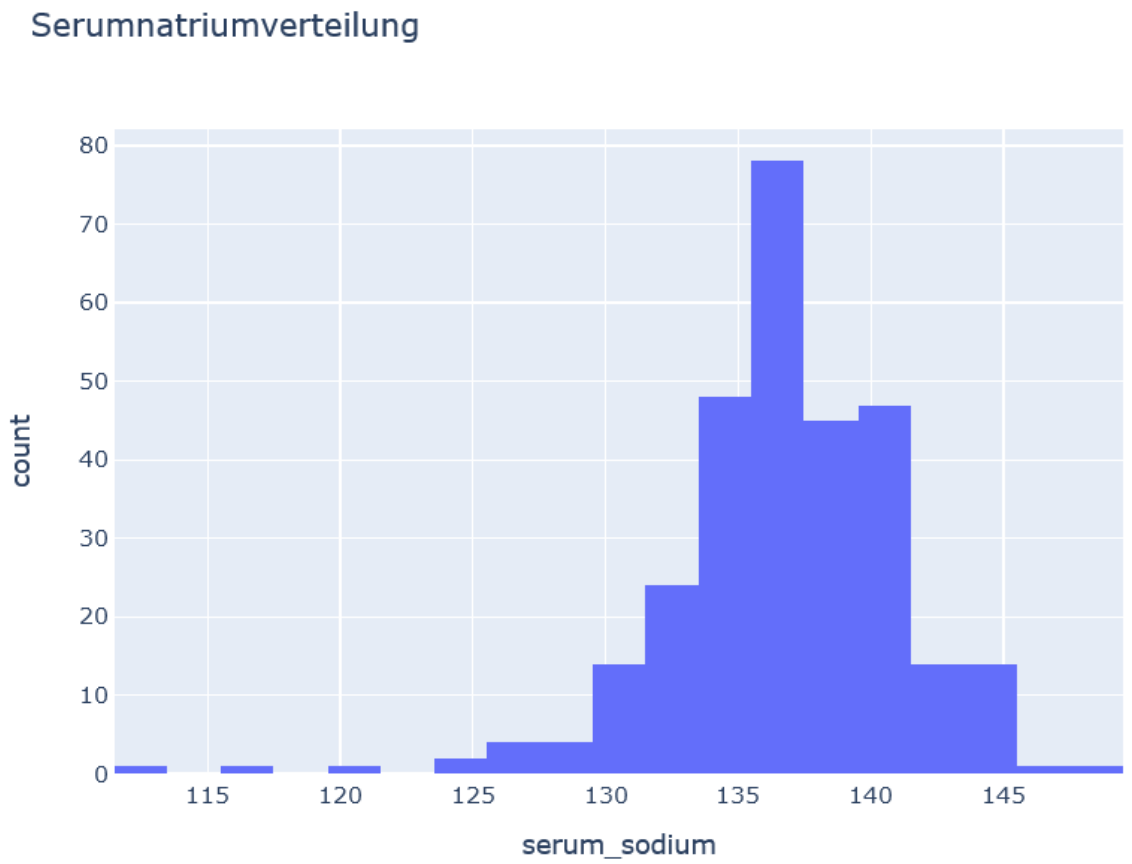
In [16]:
```
fig = px.histogram(dataset, "platelets", nbins=25, title='Thrombozytenverteilung', width=700)
fig.show()
# Der Normalwert von Thrombozyten im Blut liegt zwischen 150.000 und 450.000 in einem mm3 Blut.
```



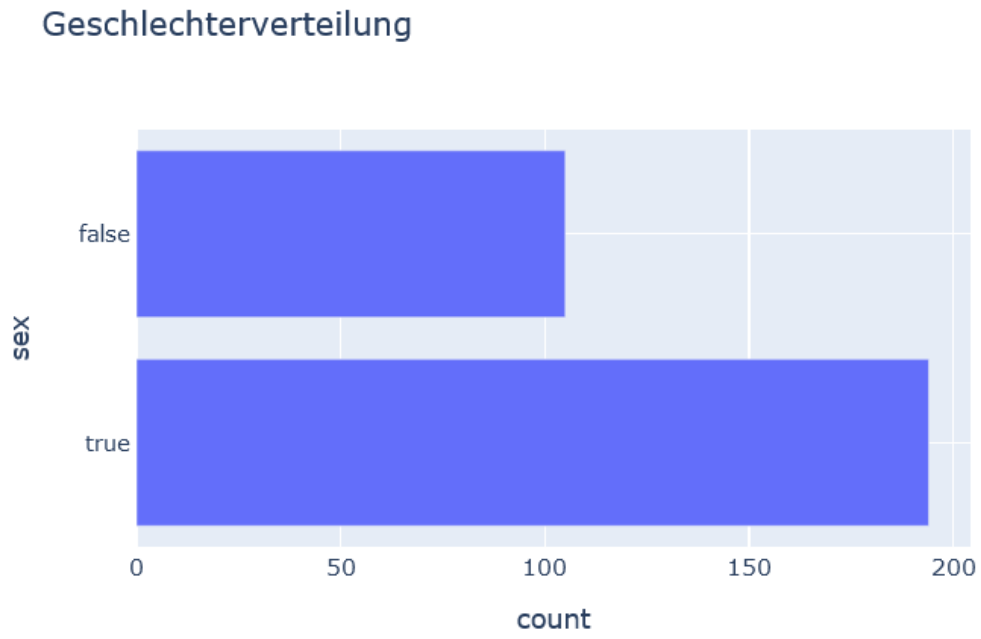
In [17]:
```
fig = px.histogram(dataset, "serum_creatinine", nbins=25, title='Serumkreatininverteilung', width=700)
fig.show()
# Der Normalwert für Kreatinin liegt bei im Normalbereich von 0,5 bis 1,2 mg / dl.
```



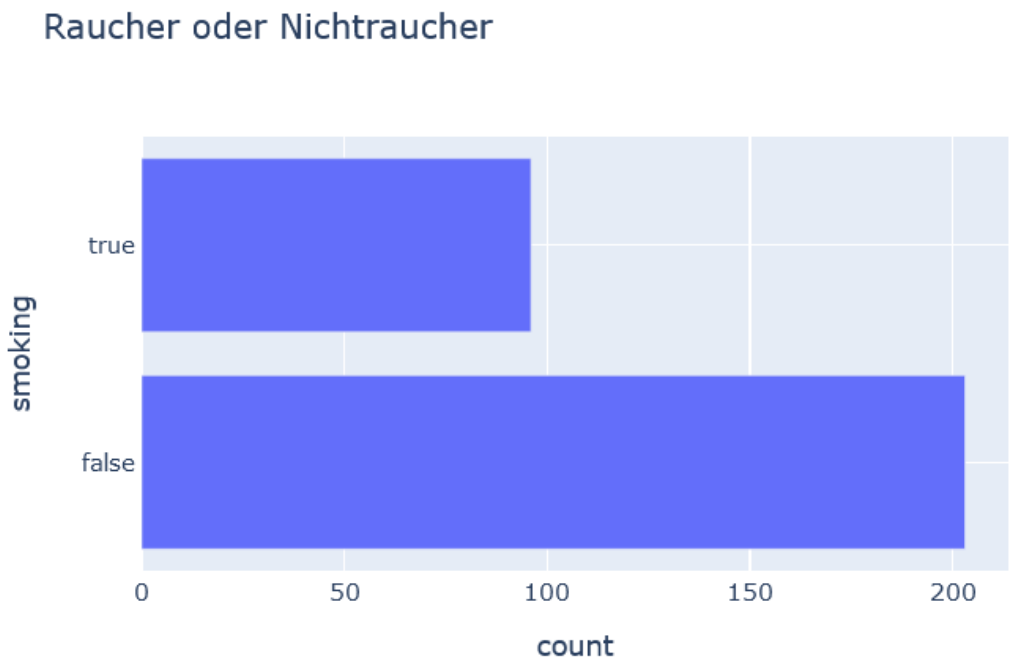
In [18]:
```
fig = px.histogram(dataset, "serum_sodium", nbins=25, title='Serumnatriumverteilung', width=700)
fig.show()
# Serumnitriumun da Der normale Bereich für Natrium im Blut beträgt 135-145 mÄq / l.
```



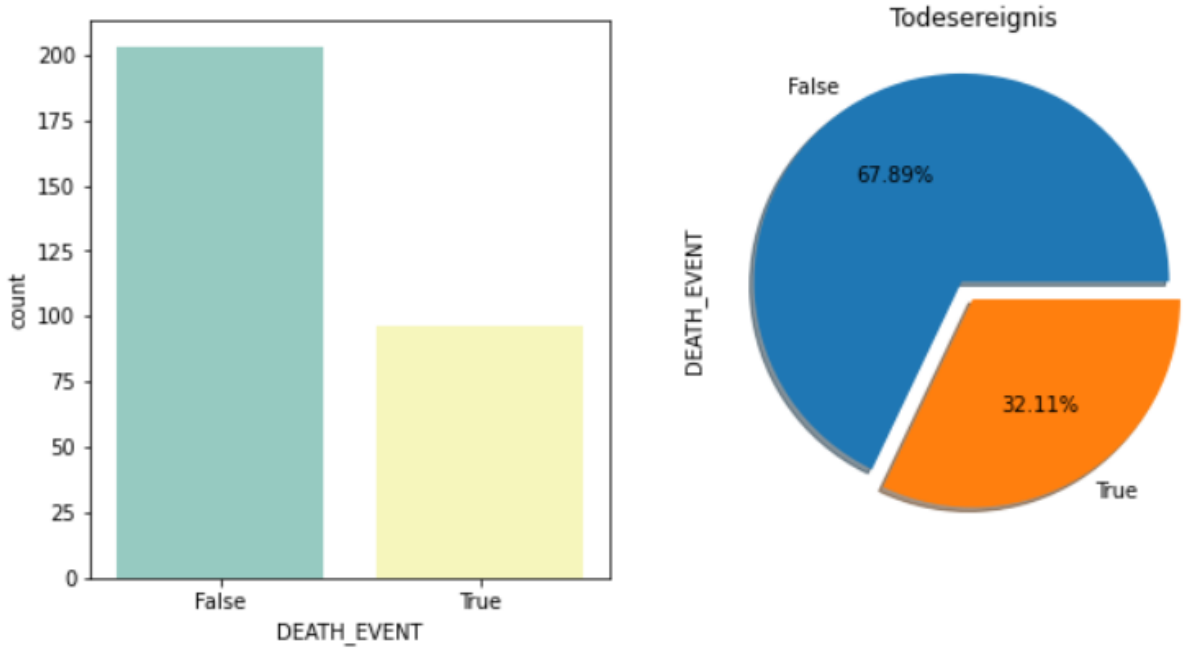
In [18]:
```
ds = dataset['sex'].value_counts().reset_index()
ds.columns = ['sex', 'count']
fig = px.bar(ds, x='count', y="sex", orientation='h', title='Geschlechterverteilung', width=600, height=400)
fig.show()
# 1 männlich
# 0 weiblich
```



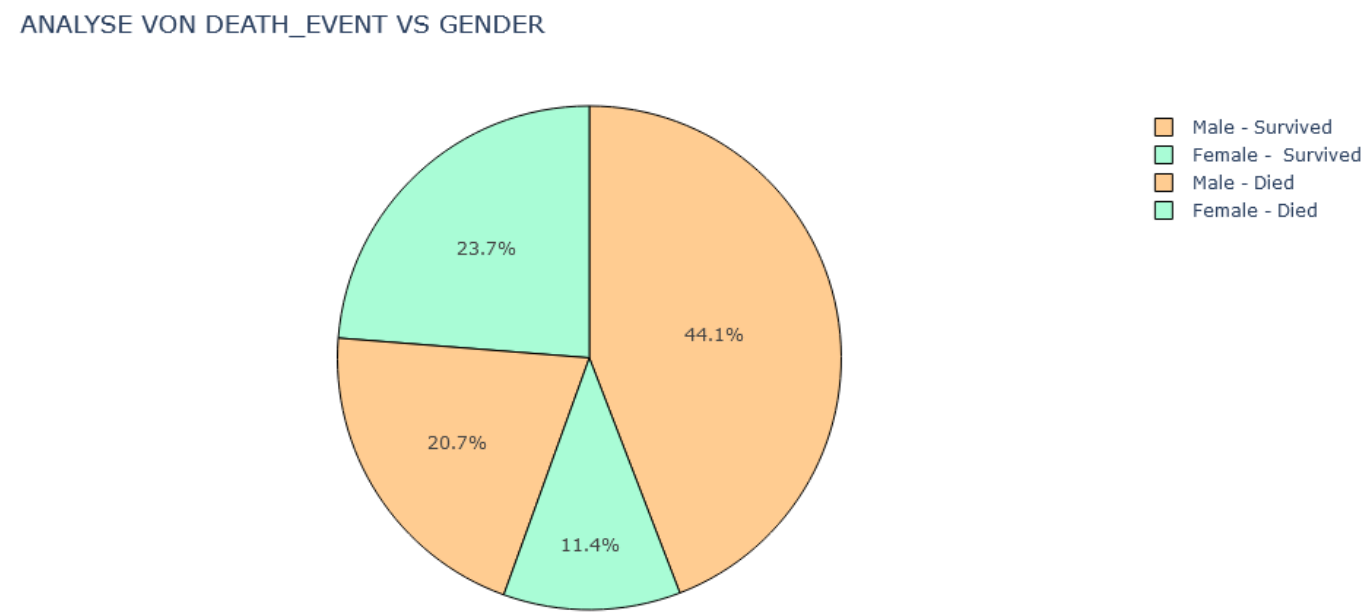
In [19]:
```
ds = dataset['smoking'].value_counts().reset_index()
ds.columns = ['smoking', 'count']
fig = px.bar(ds, x='count', y="smoking", orientation='h', title='Raucher oder Nichtraucher', width=600, height=400)
fig.show()
```



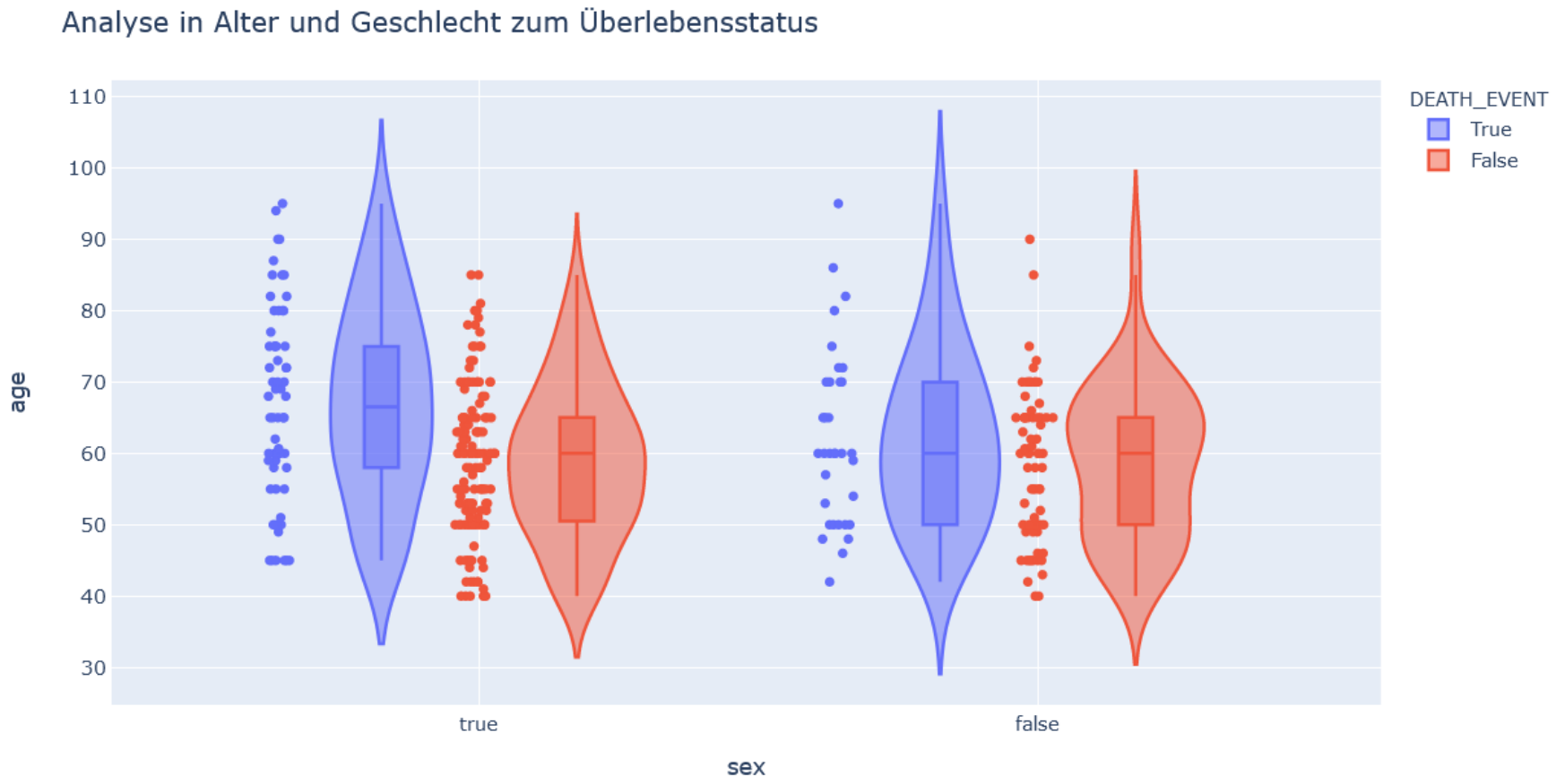
In [20]:
```
fig,ax = plt.subplots(1,2,figsize=(10,5))
sns.countplot(data = dataset , x= "DEATH_EVENT" ,palette = "Set3" ,ax=ax[0])
plt.title("Todesereignis")
dataset.DEATH_EVENT.value_counts().plot.pie(explode =[0.1,0] , autopct = "%0.2f%%" ,shadow = True ,ax = ax[1])
plt.show()
```



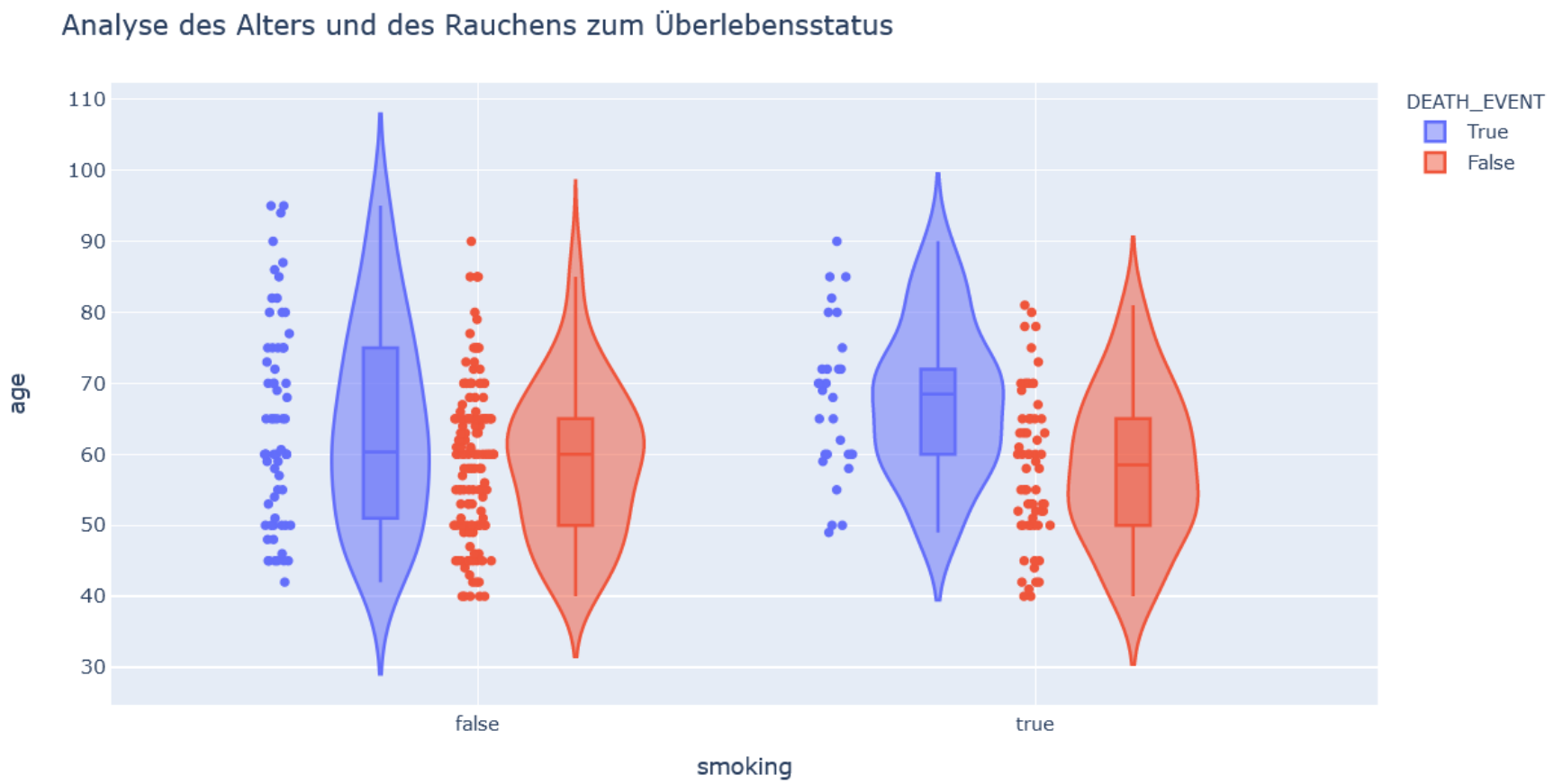
In [24]:
```
d1 = dataset[(dataset["DEATH_EVENT"]==0) & (dataset["sex"]==1)]
d2 = dataset[(dataset["DEATH_EVENT"]==1) & (dataset["sex"]==1)]
d3 = dataset[(dataset["DEATH_EVENT"]==0) & (dataset["sex"]==0)]
d4 = dataset[(dataset["DEATH_EVENT"]==1) & (dataset["sex"]==0)]

labels = ['Male - Survived','Male - Died', "Female -  Survived", "Female - Died"]
values = [len(d1),len(d2),len(d3),len(d4)]

colors = ['#ffcc91', '#ffcc91', '#a9fcd6', '#a9fcd6']

fig = go.Figure(data=[go.Pie(labels=labels, values=values)])
fig.update_layout(title_text="ANALYSE VON DEATH_EVENT VS GENDER")
fig.update_traces(marker=dict(colors=colors, line=dict(color='#000000', width=1)))
fig.show()
```



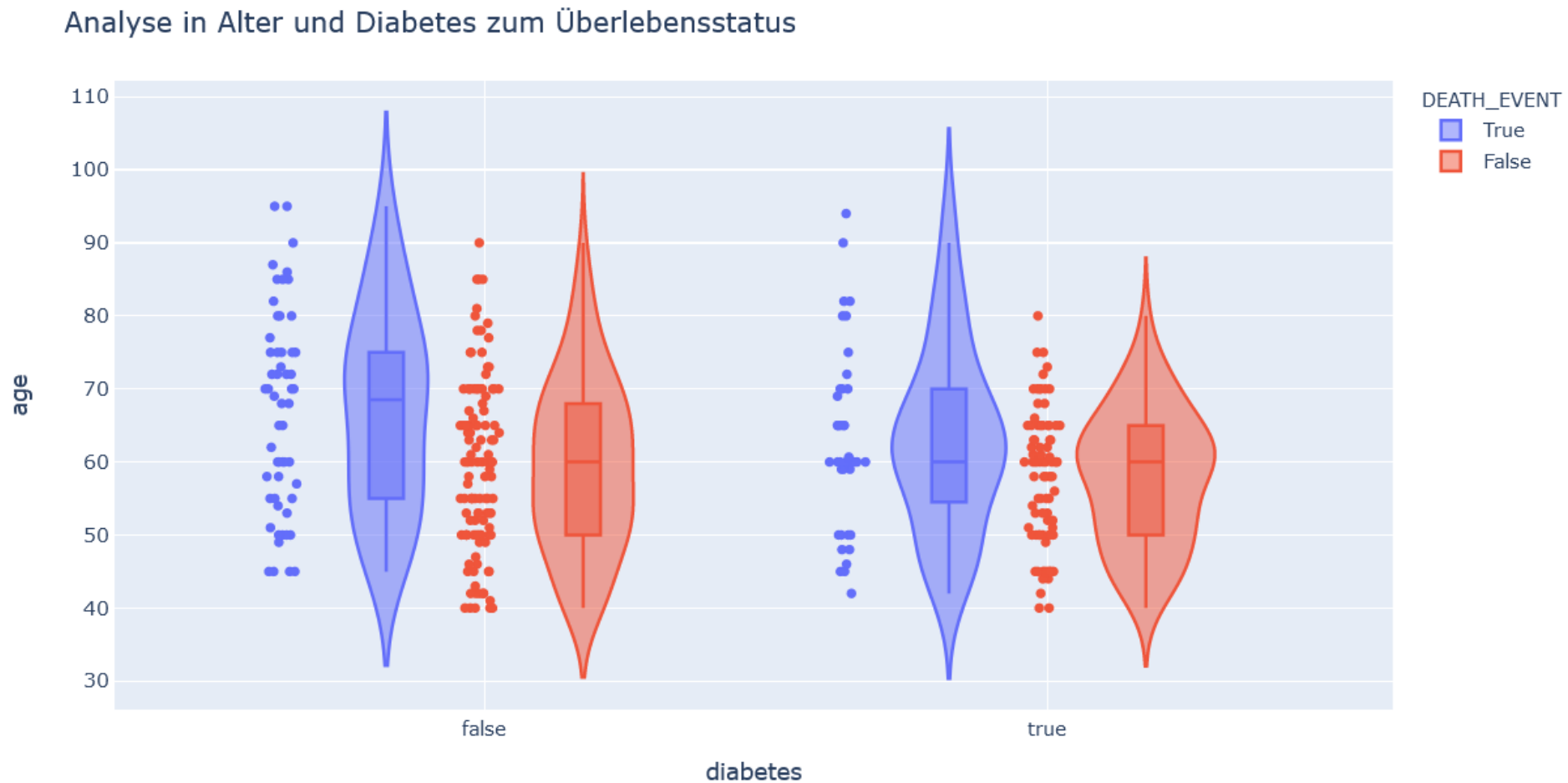
In [24]:
```
fig = px.violin(dataset, y="age", x="sex", color="DEATH_EVENT", box=True, points="all", hover_data=dataset.columns)
fig.update_layout(title_text="Analyse in Alter und Geschlecht zum Überlebensstatus")
fig.show()
# 1 männlich
# 0 weiblich
```



In [25]:
```
fig = px.violin(dataset, y="age", x="smoking", color="DEATH_EVENT", box=True, points="all", hover_data=dataset.columns)
fig.update_layout(title_text="Analyse des Alters und des Rauchens zum Überlebensstatus")
fig.show()

# Rechte Seite: Rauchens 1
# Linke Seite: Rauchens 0
```



In [26]:
```
fig = px.violin(dataset, y="age", x="diabetes", color="DEATH_EVENT", box=True, points="all", hover_data=dataset.column
s)
fig.update_layout(title_text="Analyse in Alter und Diabetes zum Überlebensstatus")
fig.show()

# Rechte Seite: Diabetes 1
# Linke Seite: Diabetes 0
```



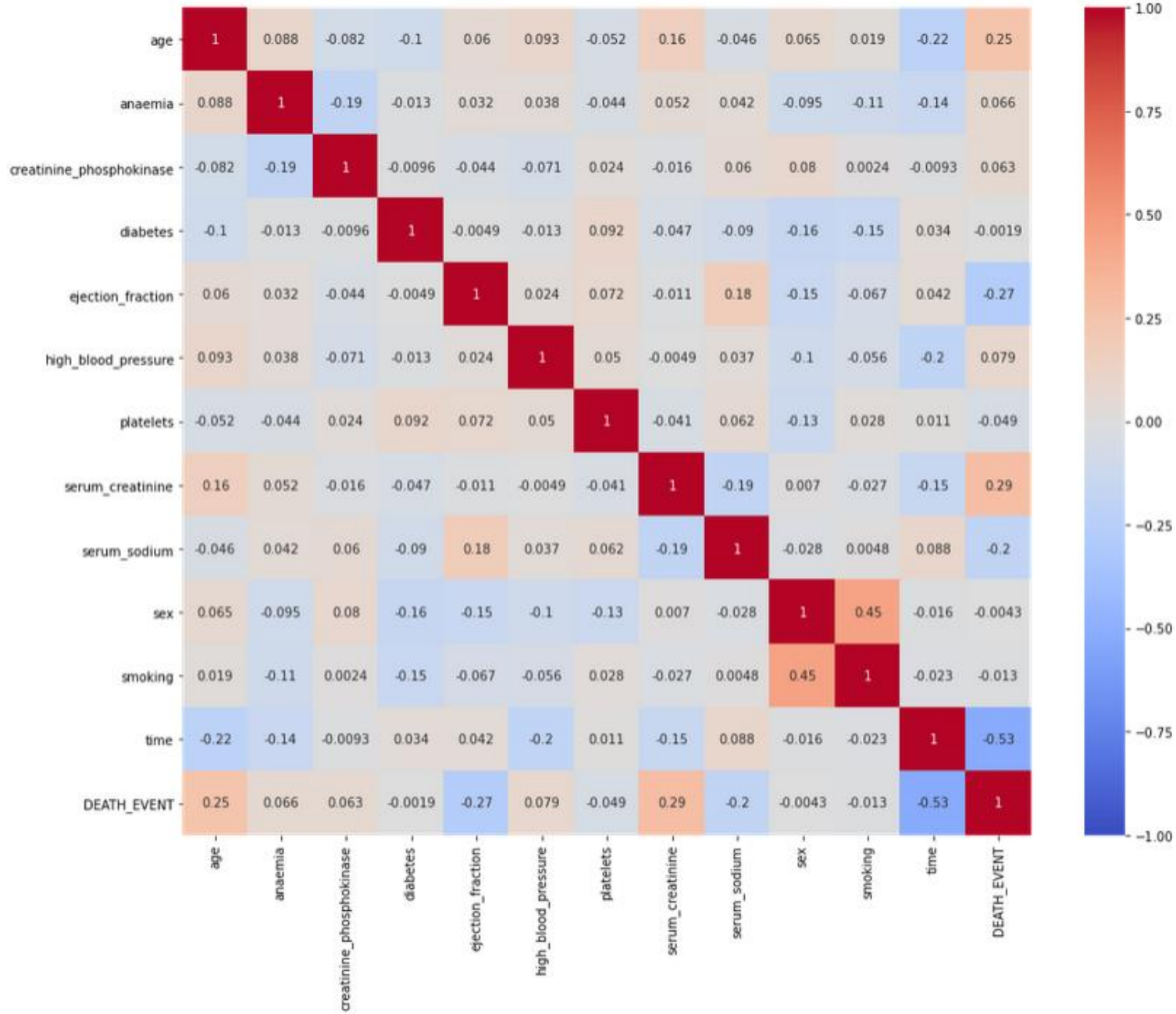
KORRELATIONSANALYSE UND RANDOM FOREST

mit Python

In [45]:

```
# Pandas ist eine der Open-Source-Python-Bibliotheken,
# die benutzerfreundliche Datenstrukturen bieten und Datenanalysen ermöglichen.
import pandas as pd

# NumPy (Numerical Python) ist eine Mathematikbibliothek,
# mit der wir schnell wissenschaftliche Berechnungen durchführen können.
import numpy as np

# Python Matplotlib; matplotlib.pyplot ist eine Python-Bibliothek, die für 2D- oder 3D-Grafiken verwendet wird.
import matplotlib.pyplot as plt

# Seaborn ist eine Bibliothek in Python, mit der interessante und informative statistische Grafiken erstellt werden.
import seaborn as sns

import warnings

# Es ist ein Grafik-Erstellungsmodul.
import plotly.express as px

import plotly.figure_factory as ff

import plotly.graph_objs as go

from sklearn.model_selection import KFold, cross_val_score, train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import confusion_matrix, accuracy_score

from mlxtend.plotting import plot_confusion_matrix

from sklearn.metrics import roc_curve,accuracy_score,plot_confusion_matrix


# Wir importieren unseren Datensatz.
dataset = pd.read_csv(r"C:\Users\ceren\Desktop\datasets_727551_1263738_heart_failure_clinical_records_dataset.csv")
```

In [3]:
```
plt.figure(figsize=(15,12))
sns.heatmap(dataset.corr(), vmin=-1, cmap='coolwarm', annot=True);
```



In [4]:
```
dataset.corr()['DEATH_EVENT'].apply(np.abs).sort_values(ascending=False)
```

Out[4]:
```
DEATH_EVENT                1.000000
time                       0.526964
serum_creatinine           0.294278
ejection_fraction          0.268603
age                        0.253729
serum_sodium               0.195204
high_blood_pressure        0.079351
anaemia                    0.066270
creatinine_phosphokinase   0.062728
platelets                  0.049139
smoking                    0.012623
sex                        0.004316
diabetes                   0.001943
Name: DEATH_EVENT, dtype: float64
```

In [49]:
```
Features = ['ejection_fraction','serum_creatinine','age','serum_sodium','high_blood_pressure']
x = dataset[Features]
y = dataset["DEATH_EVENT"]
x_train,x_test,y_train,y_test = train_test_split(x,y, test_size=0.2, random_state=2698)
```

In [50]:
```
print("x_train Shape : ", x_train.shape)
print("X_test Shape  : ", x_test.shape)
print("y_train Shape : ", y_train.shape)
print("y_test Shape  : ", y_test.shape)
```

```
x_train Shape :  (239, 5)
X_test Shape  :  (60, 5)
y_train Shape :  (239,)
y_test Shape  :  (60,)
```

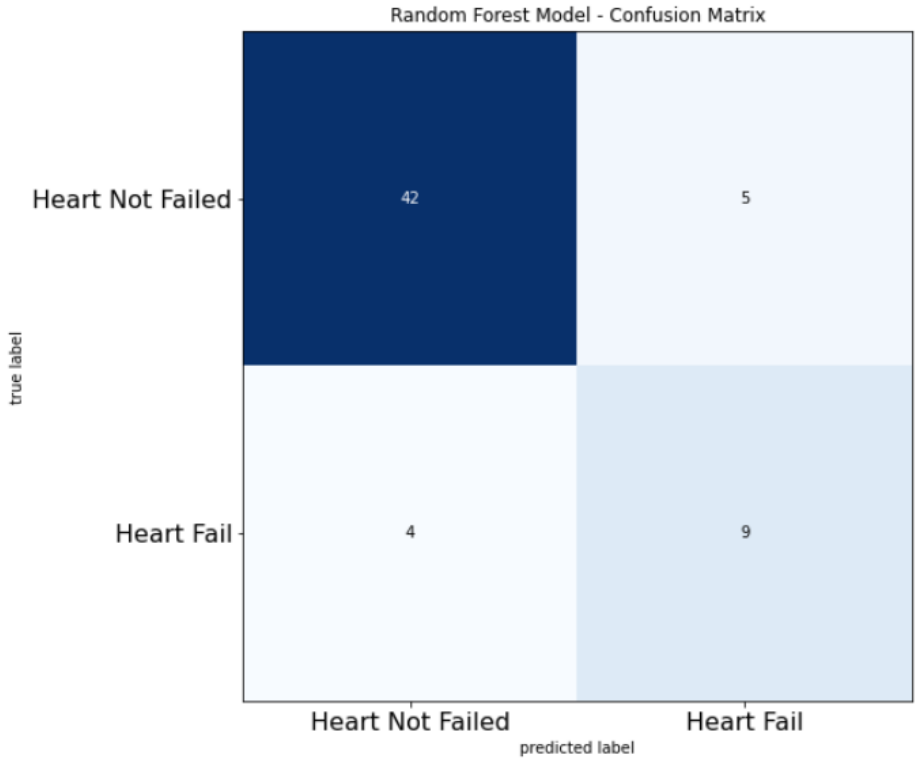
In [51]:
```
clf = RandomForestClassifier(max_features=0.5, max_depth=15, random_state=1)
clf.fit(x_train, y_train)
pred=clf.predict(x_test)
```

In [52]:
```
print("Accuracy of RandomForestClassifier is /Train set: ",clf.score(x_train,y_train))
print("Accuracy of RandomForestClassifier is /Test set : ",clf.score(x_test,y_test))
```

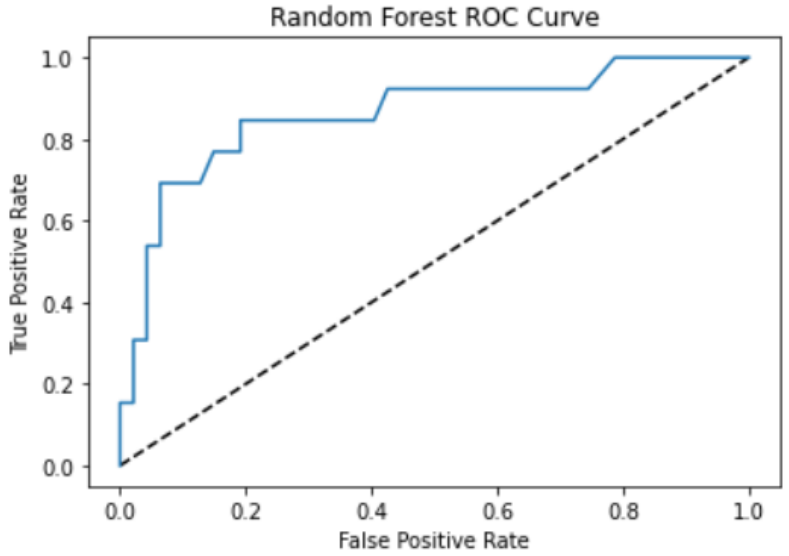
```
Accuracy of RandomForestClassifier is /Train set:  1.0
Accuracy of RandomForestClassifier is /Test set :  0.85
```

In [53]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
from mlxtend.plotting import plot_confusion_matrix
cm = confusion_matrix(y_test, pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Random Forest Model - Confusion Matrix")
plt.xticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.yticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [54]:
```
model = RandomForestClassifier(min_samples_split=2, class_weight={0:2,1:7}, random_state=13)
model.fit(x_train, y_train)
y_pred_prob = model.predict_proba(x_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='RF')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve')
plt.show()
```



mit Python

LOGISTISCHE REGRESSION

In [34]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objs as go
from sklearn.model_selection import KFold, cross_val_score, train_test_split
from sklearn import svm, datasets
from sklearn.model_selection import train_test_split
from sklearn.metrics import plot_confusion_matrix
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.metrics import r2_score, roc_auc_score, roc_curve, classification_report
```

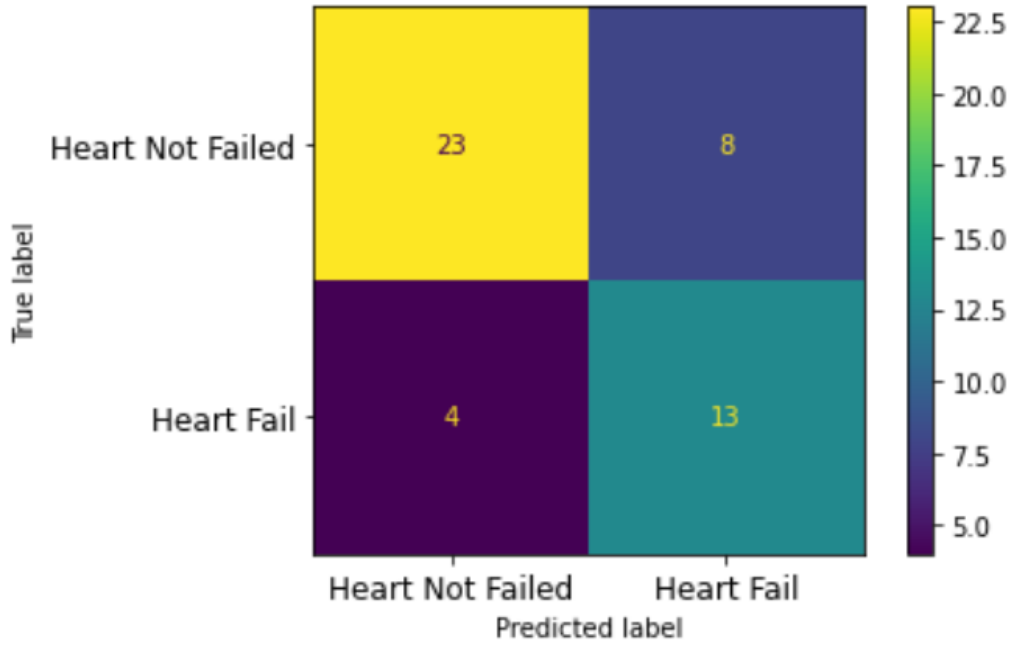
In [35]:
```
dataset = pd.read_csv(r"C:\Users\ceren\Desktop\datasets_727551_1263738_heart_failure_clinical_records_dataset.csv")
```

In [36]:
```
X = dataset.copy()
y = X['DEATH_EVENT']
X = X.drop(['DEATH_EVENT'], axis=1)

X, X_test, y, y_test = train_test_split(X, y, random_state=0, test_size=0.2, shuffle=True)
```

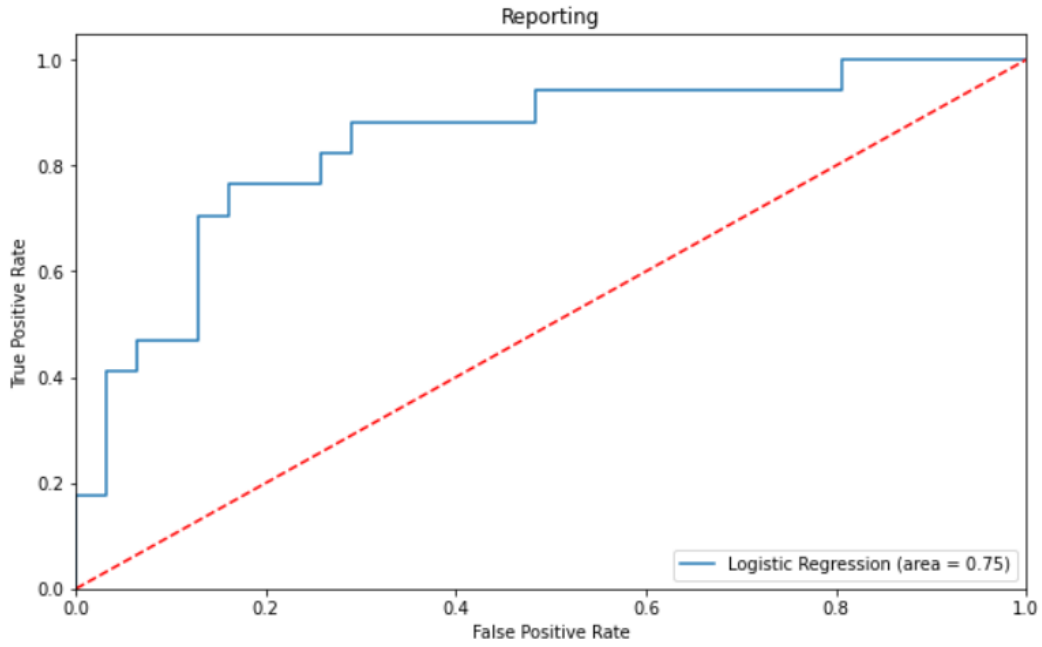
In [37]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on train set: {:.2f}'.format(logreg.score(X_train, y_train)))
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
```

```
Accuracy of logistic regression classifier on train set: 0.85
Accuracy of logistic regression classifier on test set: 0.75
```

In [38]:
```
plot_confusion_matrix(logreg, X_test, y_test)
plt.xticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=12)
plt.yticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=12)
plt.show()
```



In [39]:
```
logit_roc_auc = roc_auc_score(y_test, logreg.predict(X_test))
fpr, tpr, thresholds = roc_curve(y_test, logreg.predict_proba(X_test)[:,1])
plt.figure(figsize=(10,6))
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' % logit_roc_auc)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Reporting')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```



CHI-QUADRAT-TEST

mit Python

In [37]:

```
# Pandas ist eine der Open-Source-Python-Bibliotheken,
# die benutzerfreundliche Datenstrukturen bieten und Datenanalysen ermöglichen.
import pandas as pd

# NumPy (Numerical Python) ist eine Mathematikbibliothek,
# mit der wir schnell wissenschaftliche Berechnungen durchführen können.
import numpy as np

# Python Matplotlib; matplotlib.pyplot ist eine Python-Bibliothek, die für 2D- oder 3D-Grafiken verwendet wird.
import matplotlib.pyplot as plt

# Seaborn ist eine Bibliothek in Python, mit der interessante und informative statistische Grafiken erstellt werden.
import seaborn as sns

import warnings

# Es ist ein Grafik-Erstellungsmodul.
import plotly.express as px

import plotly.figure_factory as ff

import plotly.graph_objs as go

from sklearn.model_selection import KFold, cross_val_score, train_test_split

from sklearn.ensemble import RandomForestClassifier


#Feature Selection
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2

# Model Selection
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV
# For parameterization and splitting data
from sklearn.metrics import confusion_matrix
from sklearn import metrics # For Accuracy


from sklearn.metrics import confusion_matrix, accuracy_score
from mlxtend.plotting import plot_confusion_matrix

from sklearn.metrics import roc_curve,accuracy_score,plot_confusion_matrix
```

In [38]:
```
df = pd.read_csv(r"C:\Users\ceren\Desktop\datasets_727551_1263738_heart_failure_clinical_records_dataset.csv")
```

In [39]:
```
#Separating the data to asses with feature selection
X_feat=df[['age', 'anaemia', 'creatinine_phosphokinase', 'diabetes',
       'ejection_fraction', 'high_blood_pressure', 'platelets',
       'serum_creatinine', 'serum_sodium', 'sex', 'smoking', 'time']]
y_feat=df['DEATH_EVENT']
```

In [40]:
```
#Feature Selection
bestfeatures = SelectKBest(score_func=chi2, k=5)
fit = bestfeatures.fit(X_feat,y_feat)
dfscores = pd.DataFrame(fit.scores_)
dfcolumns = pd.DataFrame(X_feat.columns)
#concat two dataframes for better visualization
featureScores = pd.concat([dfcolumns,dfscores],axis=1)
featureScores.columns = ['Factors','Score']  #naming the dataframe columns
print(featureScores.nlargest(6,'Score'))  #print 5 best features
```

```
                     Factors         Score
6                  platelets  26135.771990
11                      time   3826.892661
2   creatinine_phosphokinase   1897.314839
4          ejection_fraction     79.072541
0                        age     44.619455
7           serum_creatinine     19.814118
```

In [41]:
```
train_accuracy= []
accuracy_list = []
algorithm = []

X_train,X_test,y_train,y_test = train_test_split(df[['platelets','serum_creatinine','creatinine_phosphokinase','ejection_fraction','age']]
                                                 ,df['DEATH_EVENT'],test_size=0.2, random_state=0)
print("X_train shape :",X_train.shape)
print("Y_train shape :",y_train.shape)
print("X_test shape :",X_test.shape)
print("Y_test shape :",y_test.shape)
```

```
X_train shape : (239, 5)
Y_train shape : (239,)
X_test shape : (60, 5)
Y_test shape : (60,)
```

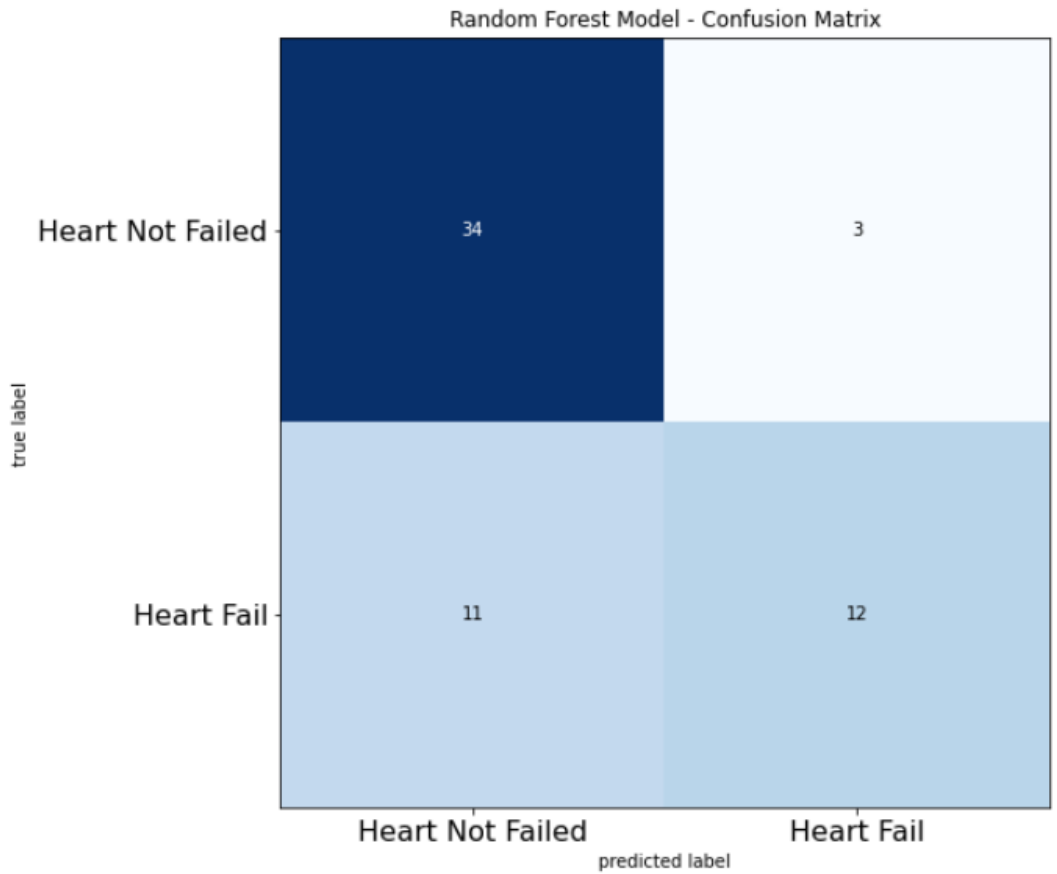
In [42]:
```
clf = RandomForestClassifier(max_features=0.5, max_depth=15, random_state=1)
clf.fit(X_train, y_train)
pred=clf.predict(X_test)
```

In [43]:
```
print("Accuracy of RandomForestClassifier is /Train set: ",clf.score(X_train,y_train))
print("Accuracy of RandomForestClassifier is /Test set : ",clf.score(X_test,y_test))
```

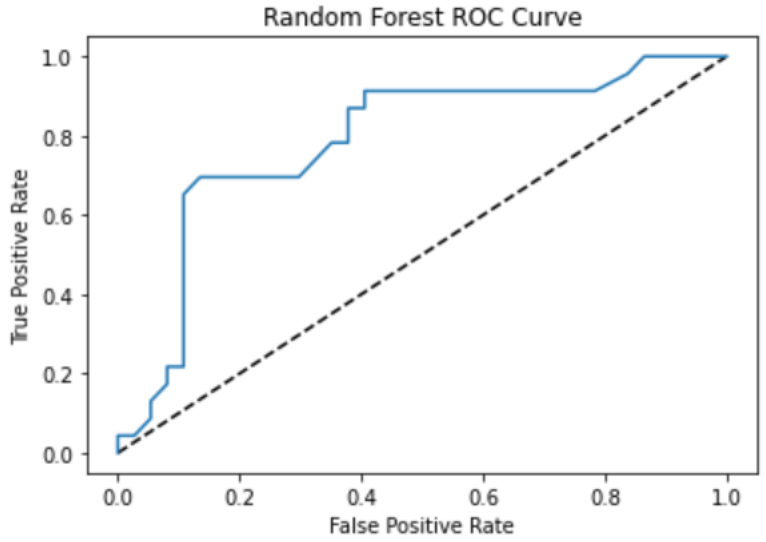
```
Accuracy of RandomForestClassifier is /Train set:  1.0
Accuracy of RandomForestClassifier is /Test set :  0.7666666666666667
```

In [44]:
```
from sklearn.metrics import confusion_matrix, accuracy_score
from mlxtend.plotting import plot_confusion_matrix
cm = confusion_matrix(y_test, pred)
plt.figure()
plot_confusion_matrix(cm, figsize=(12,8), hide_ticks=True, cmap=plt.cm.Blues)
plt.title("Random Forest Model - Confusion Matrix")
plt.xticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.yticks(range(2), ["Heart Not Failed","Heart Fail"], fontsize=16)
plt.show()
```

```
<Figure size 432x288 with 0 Axes>
```



In [45]:
```
model = RandomForestClassifier(min_samples_split=2, class_weight={0:2,1:7}, random_state=13)
model.fit(X_train, y_train)
y_pred_prob = model.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_prob)
plt.plot([0, 1], [0, 1], 'k--')
plt.plot(fpr, tpr, label='RF')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Random Forest ROC Curve')
plt.show()
```



Accuracy in feature selection based on Correlation Analysis

```
Accuracy of RandomForestClassifier is /Test set :  0.85
```

Accuracy in feature selection based on Chi2 Test

```
Accuracy of RandomForestClassifier is /Test set :  0.7666666666666667
```

```
Accuracy of logistic regression classifier on test set: 0.75
```


VIELEN DANK FÜR IHRE
AUFMERKSAMKEIT