CREATING A SURVIVAL MODEL USING DATA FROM PATIENTS WITH CARDIOVASCULAR HEART DISEASE

A Project Presentation

PRESENTED TO PROFESSOR **WENDY LEE**

DEPARTMENT OF COMPUTER SCIENCE
SAN JOSÉ STATE UNIVERSITY

IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE CLASS **CS185C-02**

BY

SARA BELL

MAY 2021

Chicco and Jurman *BMC Medical Informatics and Decision Making* (2020) 20:16
https://doi.org/10.1186/s12911-020-1023-5

BMC Medical Informatics and Decision Making

RESEARCH ARTICLE Open Access

Machine learning can predict survival of patients with heart failure from serum creatinine and ejection fraction alone

Davide Chicco[1*] and Giuseppe Jurman[2]

PLOS ONE

RESEARCH ARTICLE

Gender based survival prediction models for heart failure patients: A case study in Pakistan

Faisal Maqbool Zahid[1], Shakeela Ramzan[2], Shahla Faisal[1]*, Ijaz Hussain[3]

1 Department of Statistics / Government College University, Faisalabad, Pakistan, **2** Faisalabad Medical University, Allied Hospital, Faisalabad, Pakistan, **3** Department of Statistics / Quaid-i-Azam University, Islamabad, Pakistan

* shahla_ramzan@yahoo.com

```
<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
```

	age	anaemia	creatinine_phosphokinase	diabetes	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	serum_sodium	sex	smoking	time	DEATH_EVENT
count	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.000000	299.00000	299.000000	299.000000	299.00000	299.000000	299.00000
mean	60.833893	0.431438	581.839465	0.418060	38.083612	0.351171	263358.029264	1.39388	136.625418	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	0.00000	73.000000	0.00000
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	1.000000	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

[7]
```
''' from 'Machine Learning can predict survival of patients with heart failure...'
Ejection fraction value, given as a percentage with physiological values
ranging between 50% and 75%

from 'Gender based survival prediction models for heart failure- a case study in Pakistan'
The values of EF were divided into three categories as normal (50 - 75), borderline
(40 - 49) and very low (<40). A very low EF value (<40%) may be an evidence of heart failure
or cardiomyopathy, but a too high value (>75%) in general is not related to heart failure
although it may cause hypertrophic cardiomyopathy.'''

# Ejection Fraction percent (%)
def EF_to_category(row):
  if row['ejection_fraction'] <= 39:
    return 'Very Low'
  if row['ejection_fraction'] >=40 and row['ejection_fraction'] <= 49:
    return 'Borderline'
  if row['ejection_fraction'] >=50 and row['ejection_fraction'] <=75:
    return 'Normal'
  if row['ejection_fraction'] >=76:
    return 'High'

cardio_copy['EF_category'] = cardio_copy.apply(lambda row: EF_to_category(row), axis='columns')
```

7 New Categorical Columns Created:

'gender'	Object
'age_group'	Object
'EF_category'	Object
'platelets_range'	Object
'CPK_range'	Object
'creatinine_range'	Object
'Outcome'	Object

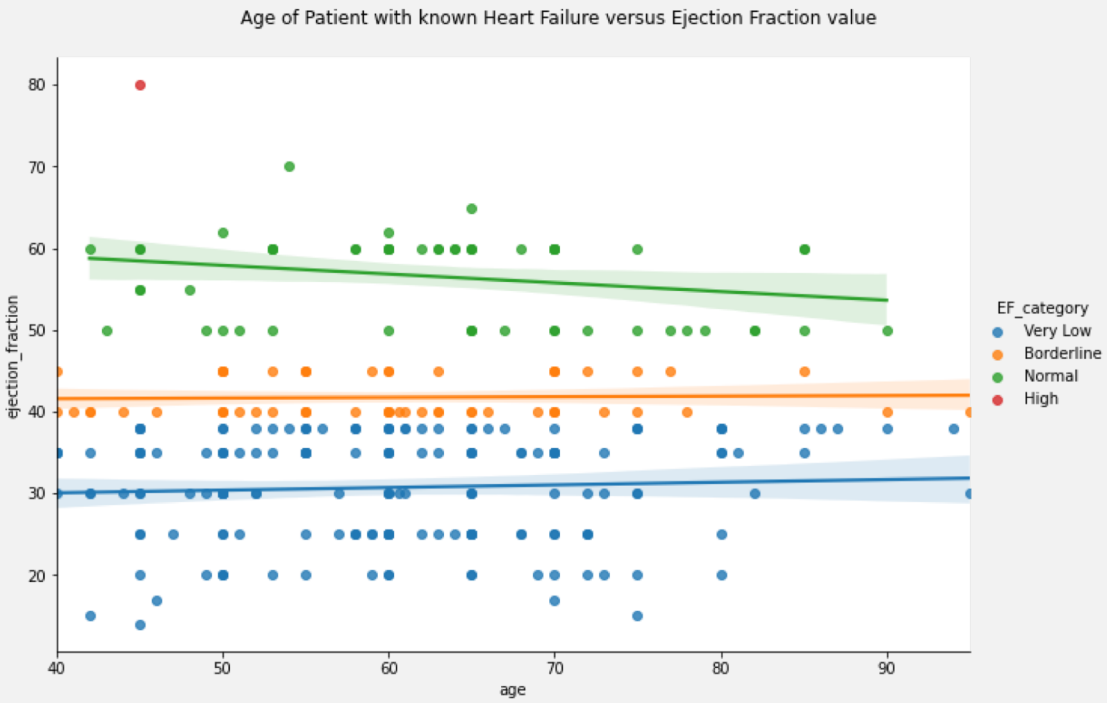
[9]
```
'''Normal Results
Total CPK normal values:
10 to 120 micrograms per liter (mcg/L)
from https://www.mountsinai.org/health-library/tests/creatine-phosphokinase-test#:~:text=Total%20CPK%20normal%20values%3A,per%20liter%20(mcg%2FL)'''

# CPK Levels enzyme in mcg/L
def CPK_to_range(row):
  if row['creatinine_phosphokinase'] <= 9:
    return 'Low'
  if row['creatinine_phosphokinase'] >=10 and row['creatinine_phosphokinase'] <= 120:
    return 'Normal'
  if row['creatinine_phosphokinase'] >=121:
    return 'High'
cardio_copy['CPK_range'] = cardio_copy.apply(lambda row: CPK_to_range(row), axis='columns')
```

[12]
```
# make Columns into 'Bool' type
# easier to create plots
cardio_copy = cardio_copy.astype({'anaemia':bool, 'diabetes': bool, 'high_blood_pressure':bool, 'smoking': bool})
cardio_copy
```

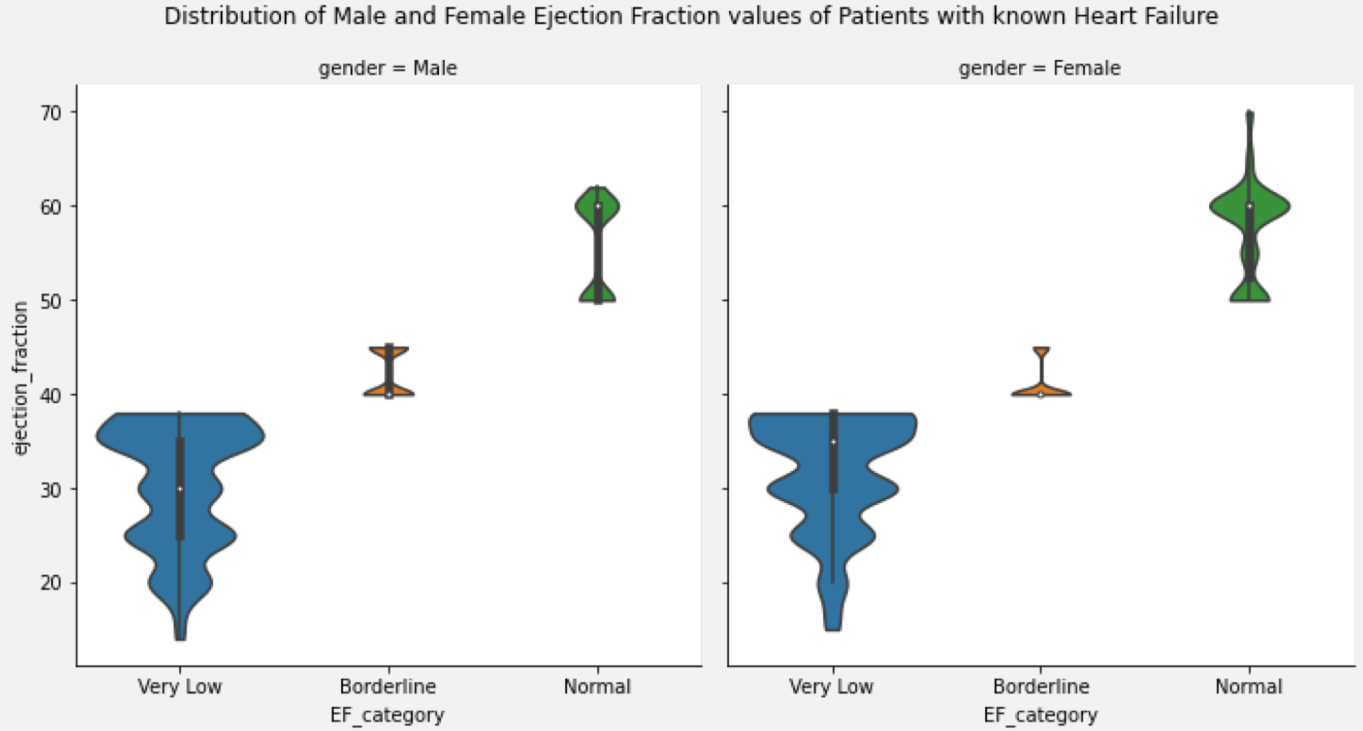
Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Ejection Fraction

Question #1.1
Is there a correlation between the patient's age and ejection fraction?



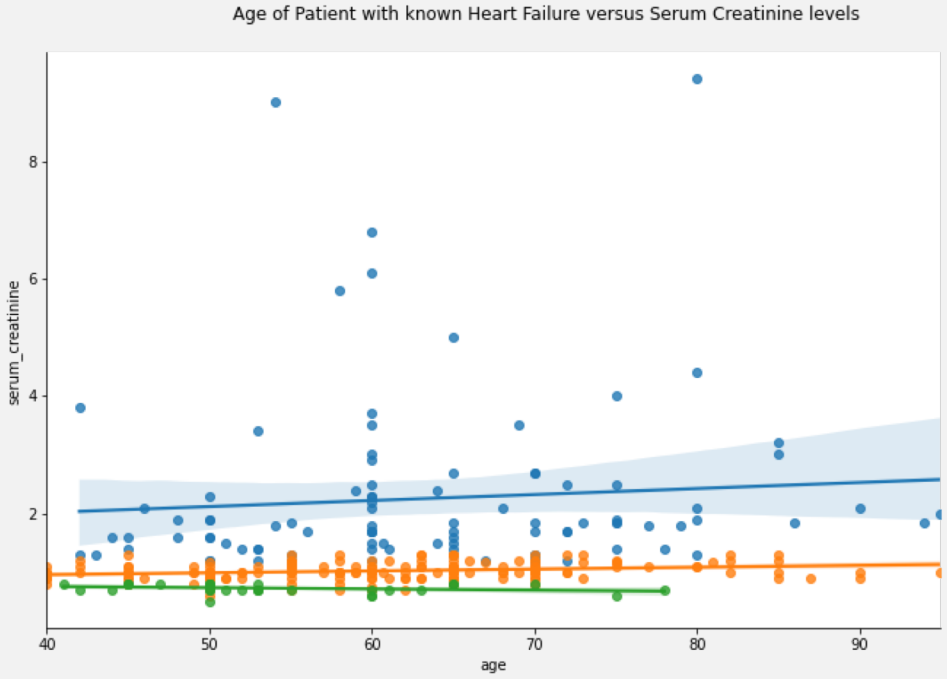
Question #1.2
Is the distribution of ejection fraction (percent) for Women and Men different?



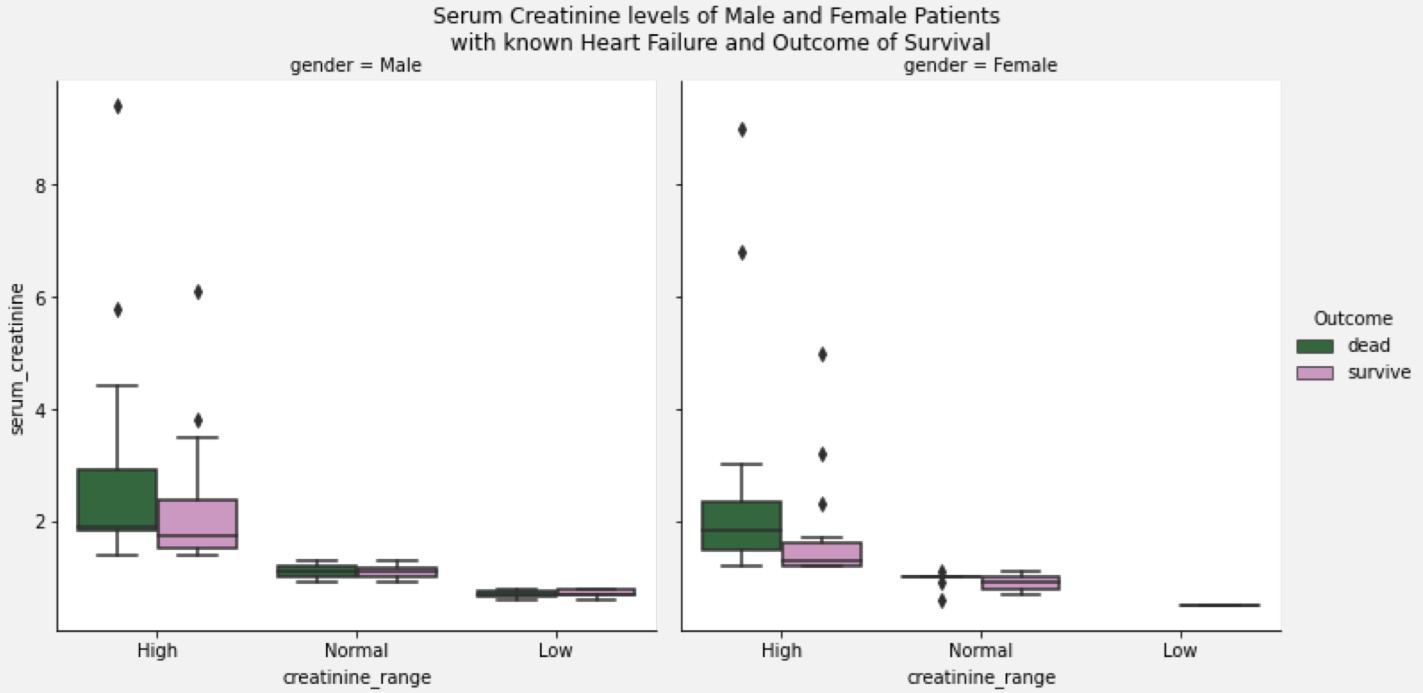
Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Serum Creatinine

Question #2.1
Is there a correlation between the patient's age and serum creatinine levels?



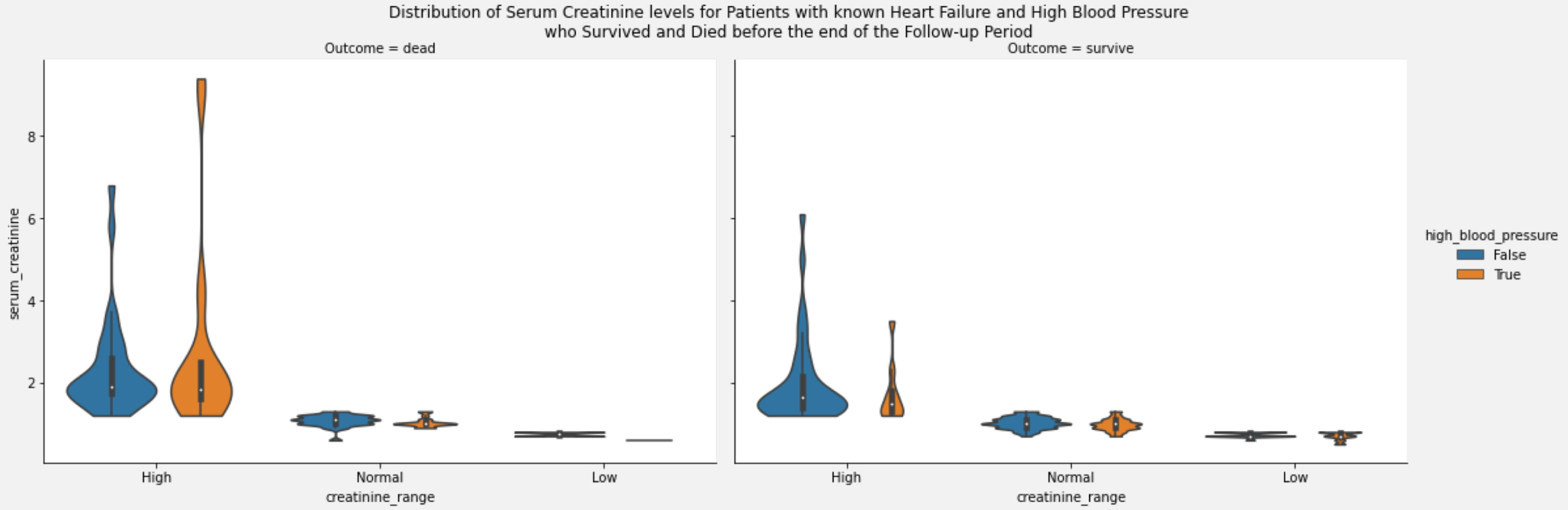
Question #2.2
Do Women or Men have higher serum creatinine levels? Did they survive or die before the end of the follow-up period?



Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Serum Creatinine, High Blood Pressure

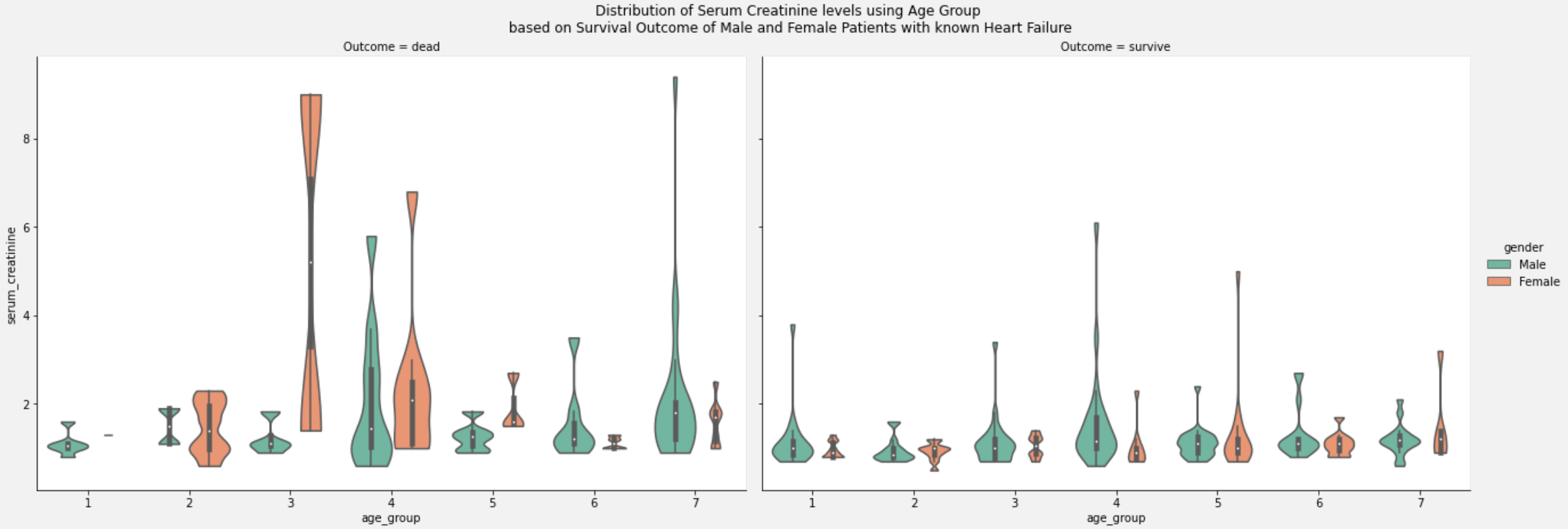
Question #2.3
Do patients with higher levels of serum creatinine also have high blood pressure? How do these values differ with patients who survived and patients who died before the end of the follow-up period?



Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Serum Creatinine, Age Group, Survival

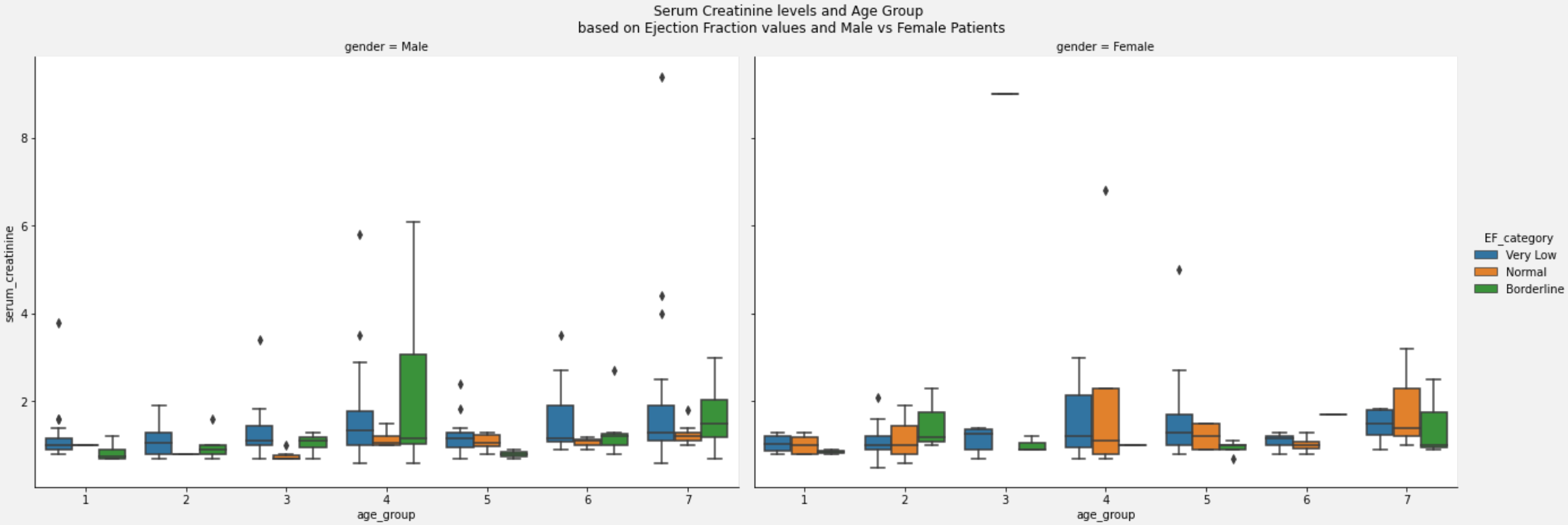
Question #2.4
When looking at the survival outcome of the patient and gender, is there a pattern with serum creatinine levels and age group?



Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Serum Creatinine, Ejection Fraction, Age Group, Gender

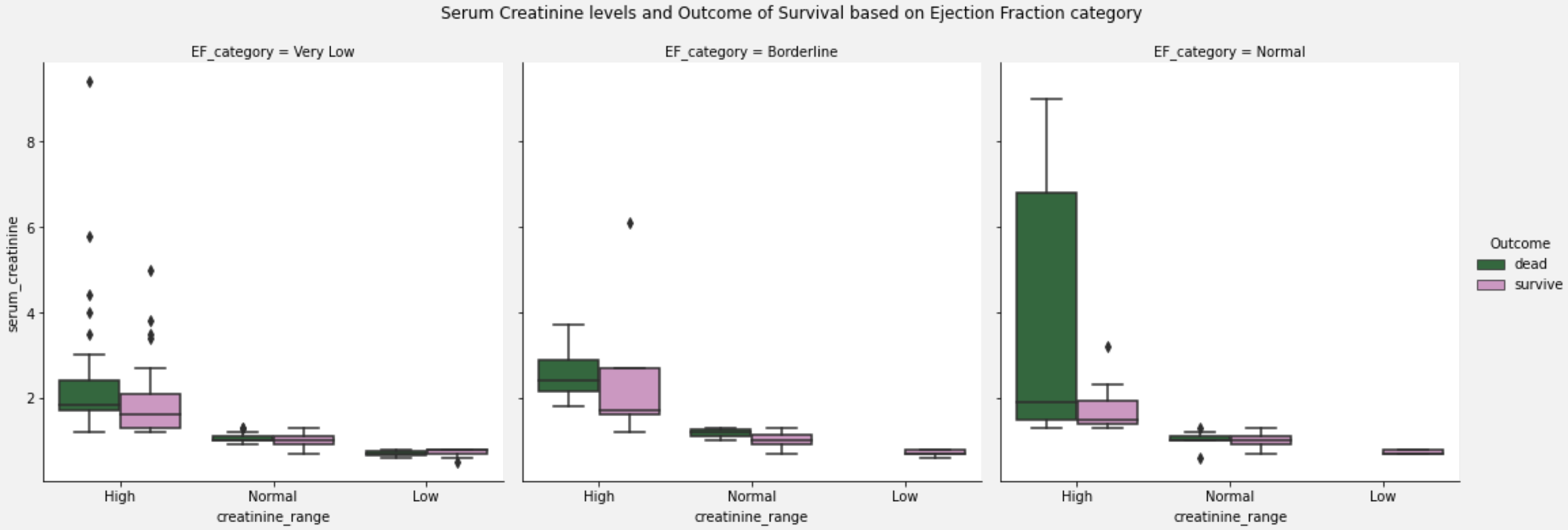
Question #3
When looking at each gender and ejection fraction, which age group has the highest serum creatinine levels?



Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Serum Creatinine, Ejection Fraction, Survival Outcome

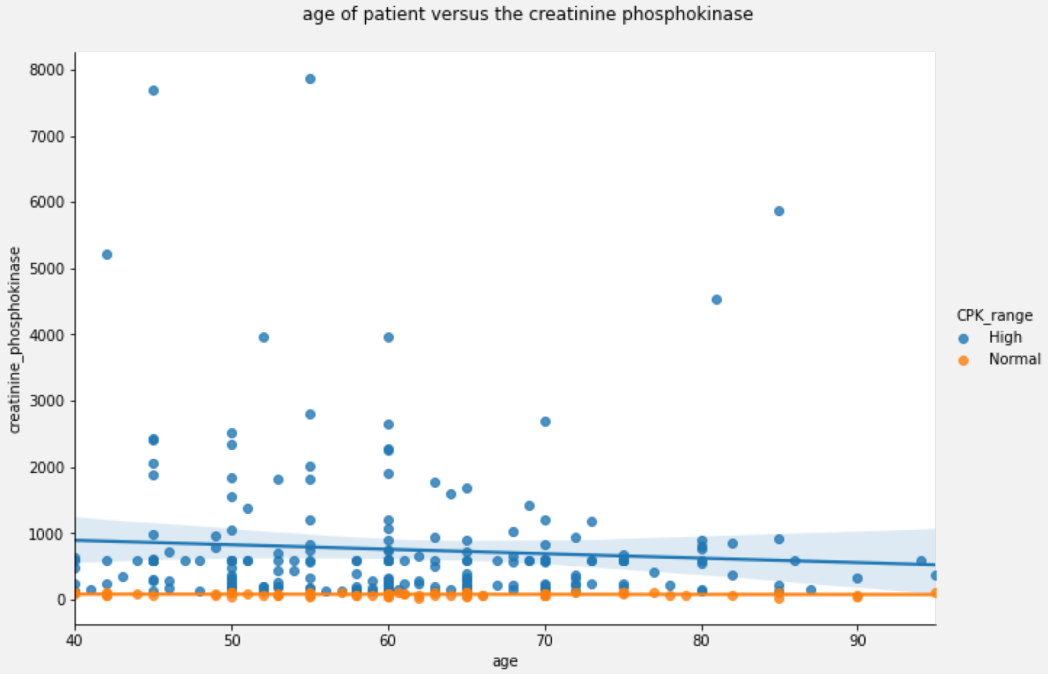
Question #4
Is there a difference in the patient's serum creatinine levels among the different ejection fraction values? Which provides more information on if the patient survived or died before the end of the follow-up period?



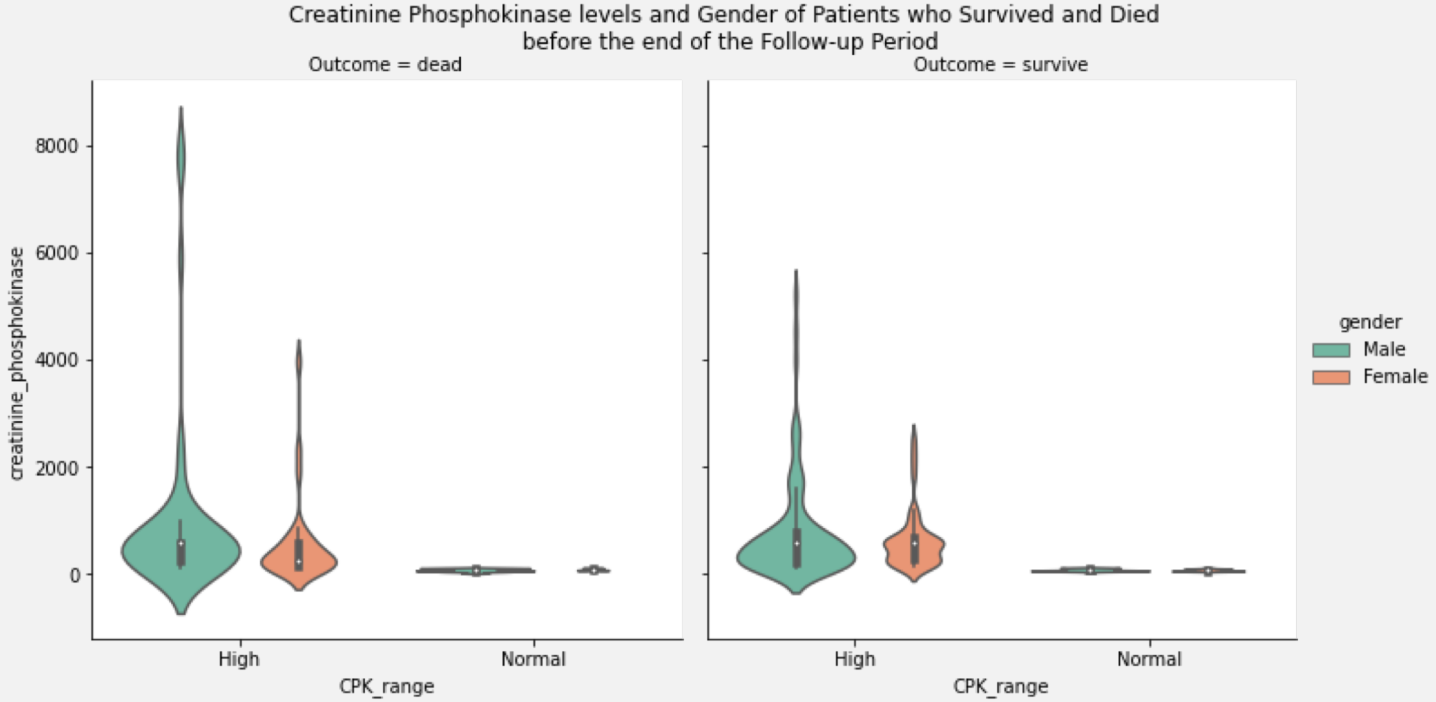
Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Creatinine Phosphokinase

Question #5.1
Is there a correlation between the patient's age and creatinine phosphokinase levels?



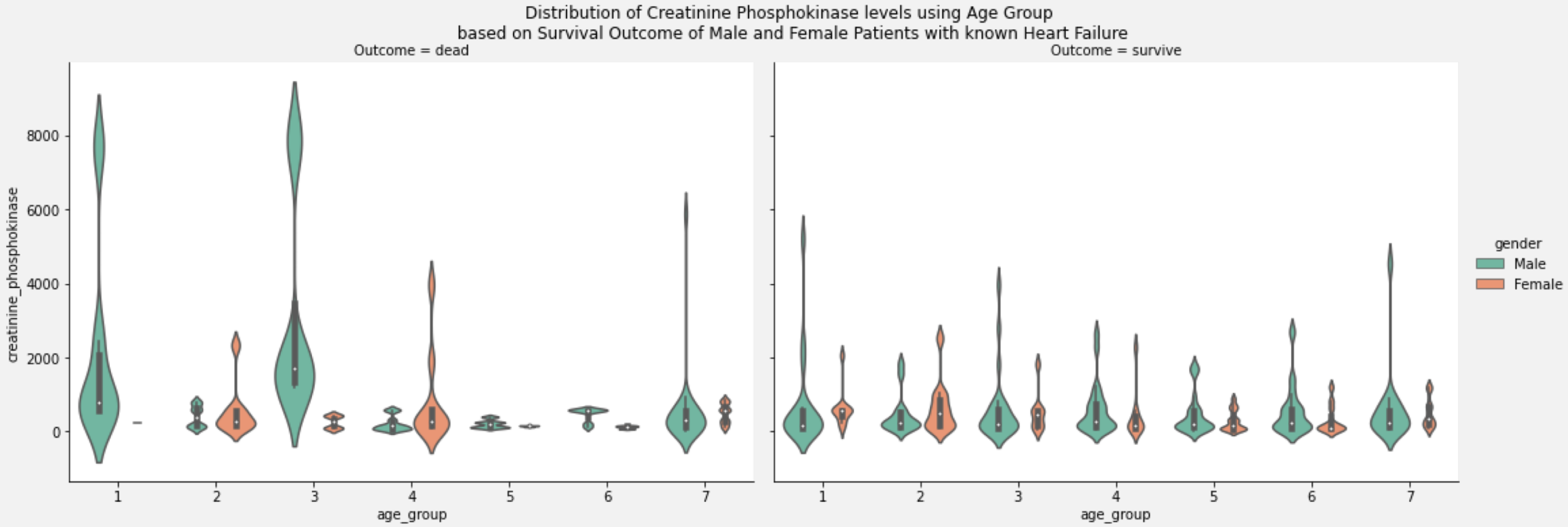
Question #5.2
Are patients with higher or lower creatinine phosphokinase levels more likely to survive or die before the end of the follow-up period? Is there a difference between men and women?



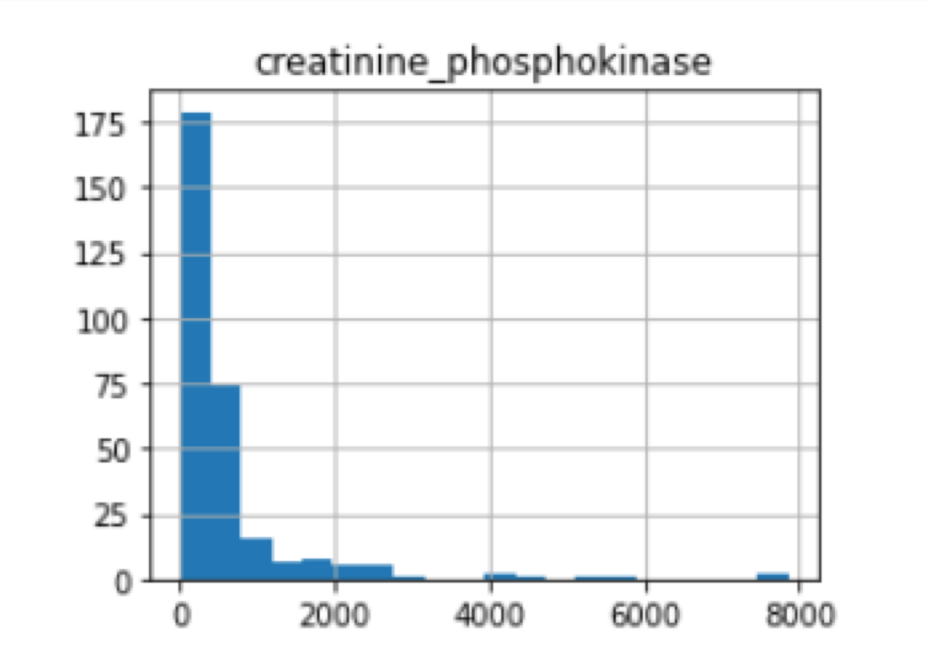
Part 1: Create and answer at least 5 unique questions using different types of plots to help you understand the data. You can create additional categorical columns or reshape your data to help you understand the data.

Creatinine Phosphokinase, Age Group, Survival

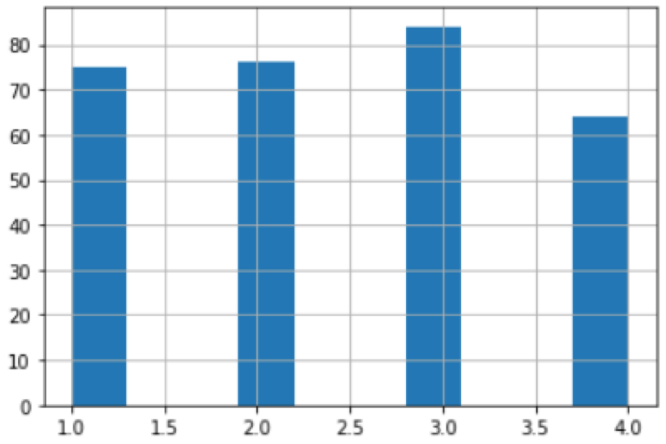
Question #5.3
When looking at the survival outcome of the patient and gender, is there a pattern with creatinine phosphokinase levels and age group?



Part 2: Create a test set and a training set using the original dataset.




```
[25]  1 # going to use attribute 'creatinine_phosphokinase' to 'cut()' and split data
      2 # min = 23, 25th = 116.5, 50th = 250, 75th = 582, max = 7861
      3 cardio_df["CPK_cat"] = pd.cut(cardio_df["creatinine_phosphokinase"],
      4                               bins=[0.0, 116.5, 250.0, 582.0, np.inf],
      5                               labels=[1, 2, 3, 4])
      6 cardio_df["CPK_cat"].hist();
```




```
12 # split(X, y, groups=None), where X is traning data and y is target variable
13 for train_index, test_index in split.split(cardio_df, cardio_df["CPK_cat"]):
14     strat_train_set = cardio_df.loc[train_index]
15     strat_test_set = cardio_df.loc[test_index]
```

```
34 # cardio_training will be a DataFrame
35 cardio_training = strat_train_set.drop("DEATH_EVENT", axis=1) # drop labels for training set
```

```
39 # cardio_target will be a Series
40 cardio_target = strat_train_set["DEATH_EVENT"].copy()
```

Part 3: Follow the steps that we use in Hands-ons 15 and 16 to prepare the data and pipeline for training a few ML classifiers that can predict a binary outcome (survive or dead). Use any strategy that you see fit. Use N-fold cross-validation to evaluate the performance of each classifier.

[28]
```
# create a copy to include numerical only df (droping binary attributes)
# in the full pipeline: num pipeline and cat pipeline
# num pipeline include: imputer (no diff because no missing data), custom tranformer, scaler
# cat pipeline inlcude: imputer (no diff because no missing data), custom transformer, but no scaler

cardio_training_num = cardio_training.drop(['anaemia', 'diabetes', 'high_blood_pressure', 'sex', 'smoking'], axis = 1)
cardio_training_cat = cardio_training.drop(['age', 'creatinine_phosphokinase', 'ejection_fraction', 'platelets', 'serum_creatinine', 'serum_sodium', 'time'], axis = 1)
#cardio_training_cat
```

[30]
```
# This is our custom transformer class
# how much serum_creatinine per CPK enzyme
creatinine_phosphokinase_idx, serum_creatinine_idx = [
    list(cardio_training_num.columns).index(col)
    for col in ('creatinine_phosphokinase', "serum_creatinine")]
print(creatinine_phosphokinase_idx, serum_creatinine_idx)
class CombinedAttributesAdder(BaseEstimator, TransformerMixin):
    def __init__(self, add_serum_creatinine_per_CPK= True):
        self.add_serum_creatinine_per_CPK = add_serum_creatinine_per_CPK

    def fit(self, X, y=None):
        return self  # nothing else to do

    def transform(self, X, y=None):
        #print(X)
        # converting units of 'serum_creatinine' (mg/dL) to units 'creatinine phosphokinase' (mcg/L)
        # creatinine_units_CPK = X[:, serum_creatinine_idx]  * (10000)
        if self.add_serum_creatinine_per_CPK:
            #print(X[:, creatinine_phosphokinase_idx])
            #print(X[:, serum_creatinine_idx])
            creatinine_units_CPK = (X[:, serum_creatinine_idx])  * (10000)
            #print(creatinine_units_CPK) # make sure indices line up
            serum_creatinine_per_CPK = (creatinine_units_CPK)  / ( X[:, creatinine_phosphokinase_idx] ) # now can divide, same units
            #print(serum_creatinine_per_CPK)
            # this only is used in num pipeline
            return np.c_[X, creatinine_units_CPK, serum_creatinine_per_CPK]
        else: # this when using cat pipeline
            return np.c_[X]
```

Part 3: Follow the steps that we use in Hands-ons 15 and 16 to prepare the data and pipeline for training a few ML classifiers that can predict a binary outcome (survive or dead). Use any strategy that you see fit. Use N-fold cross-validation to evaluate the performance of each classifier.

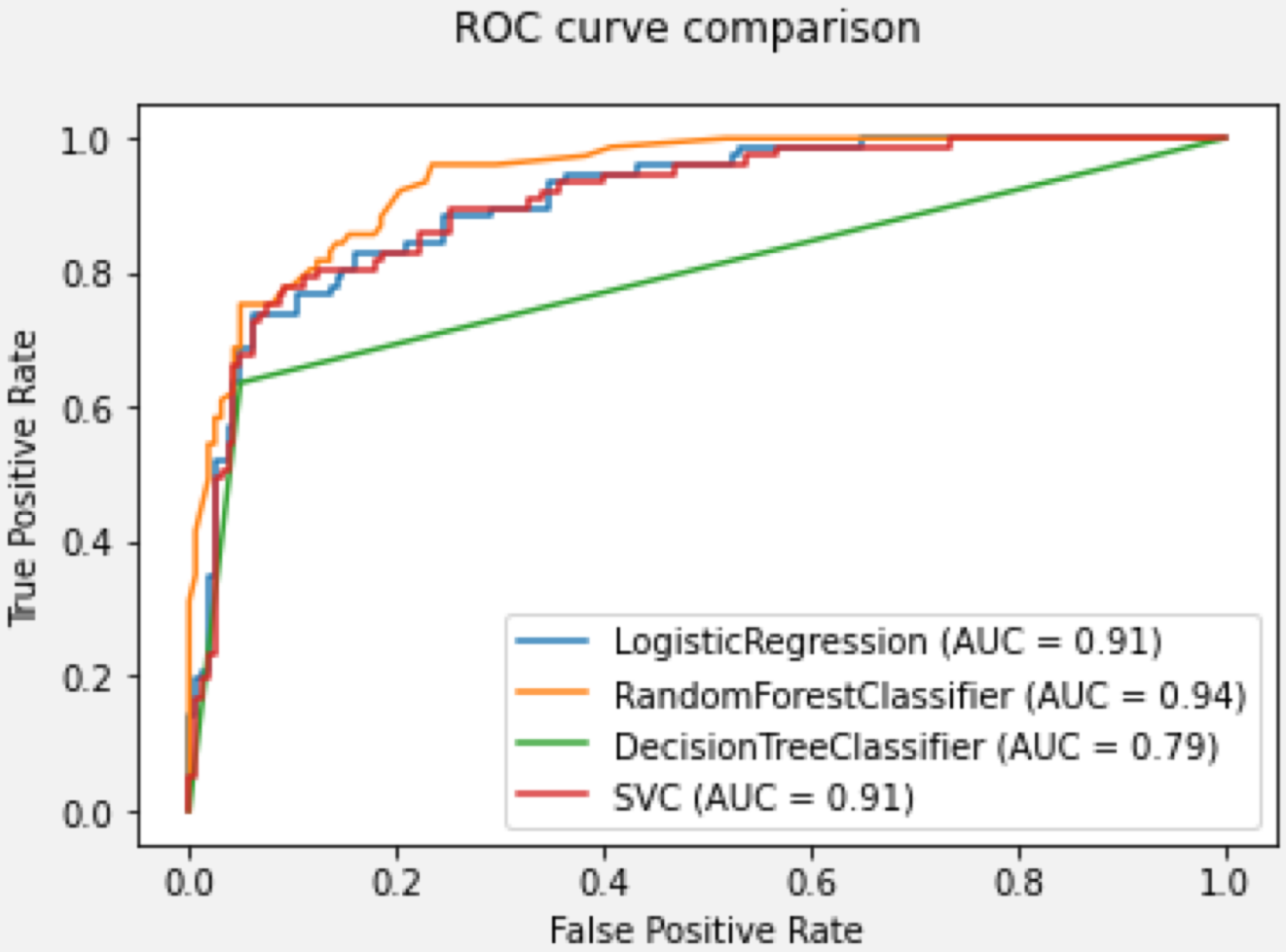
Attribute	Model	CV	Accuracy	Attribute	Model	CV	Accuracy
Serum Creatinine	Logistic Regression	10	0.82 (+/-0.12)	Creatinine Phosphokinase	Logistic Regression	10	0.83 (+/-0.10)
Serum Creatinine	Decision Tree Classifier	10	0.77 (+/-0.18)	Creatinine Phosphokinase	Decision Tree Classifier	10	0.80 (+/-0.14)
Serum Creatinine	Random Forest Classifier	10	0.81 (+/-0.08)	Creatinine Phosphokinase	Random Forest Classifier	10	0.84 (+/-0.16)
Serum Creatinine	GaussianNB	10	0.75 (+/-0.14)	Creatinine Phosphokinase	GaussianNB	10	0.74 (+/-0.13)
Serum Creatinine	SVM Classification	10	0.80 (+/-.10)	Creatinine Phosphokinase	SVM Classification	10	0.83 (+/-0.12)

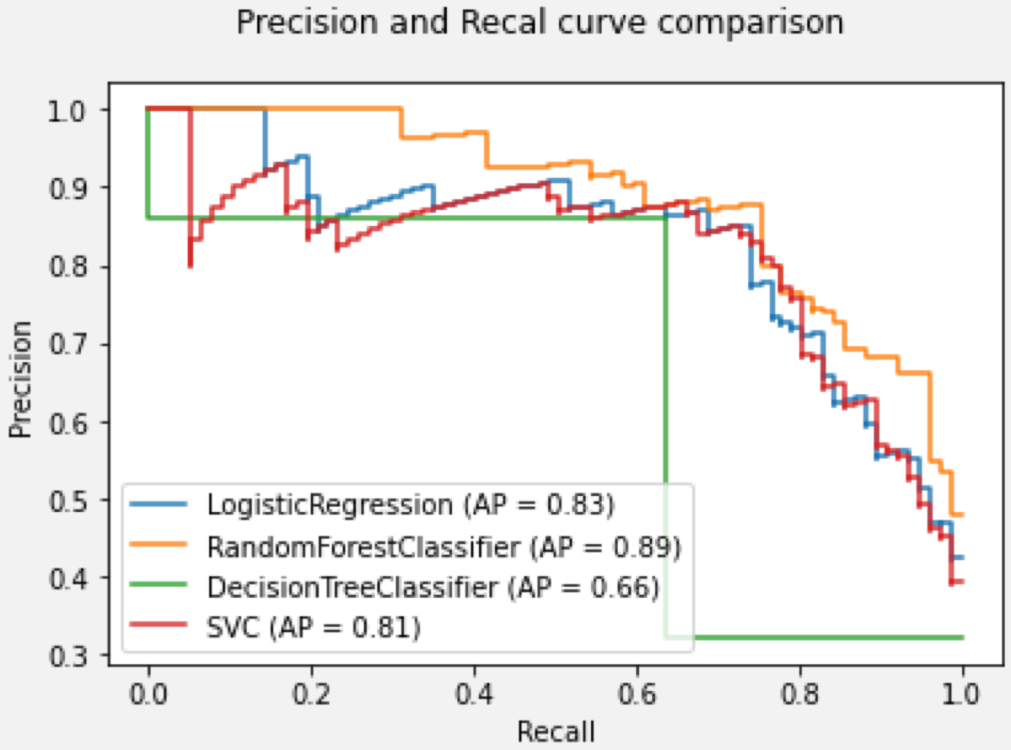
Part 3: Follow the steps that we use in Hands-ons 15 and 16 to prepare the data and pipeline for training a few ML classifiers that can predict a binary outcome (survive or dead). Use any strategy that you see fit. Use N-fold cross-validation to evaluate the performance of each classifier.

Fine-Tuning Models with GridSearch using Creatinine Phosphokinase: Obtaining 'best_params' for each model

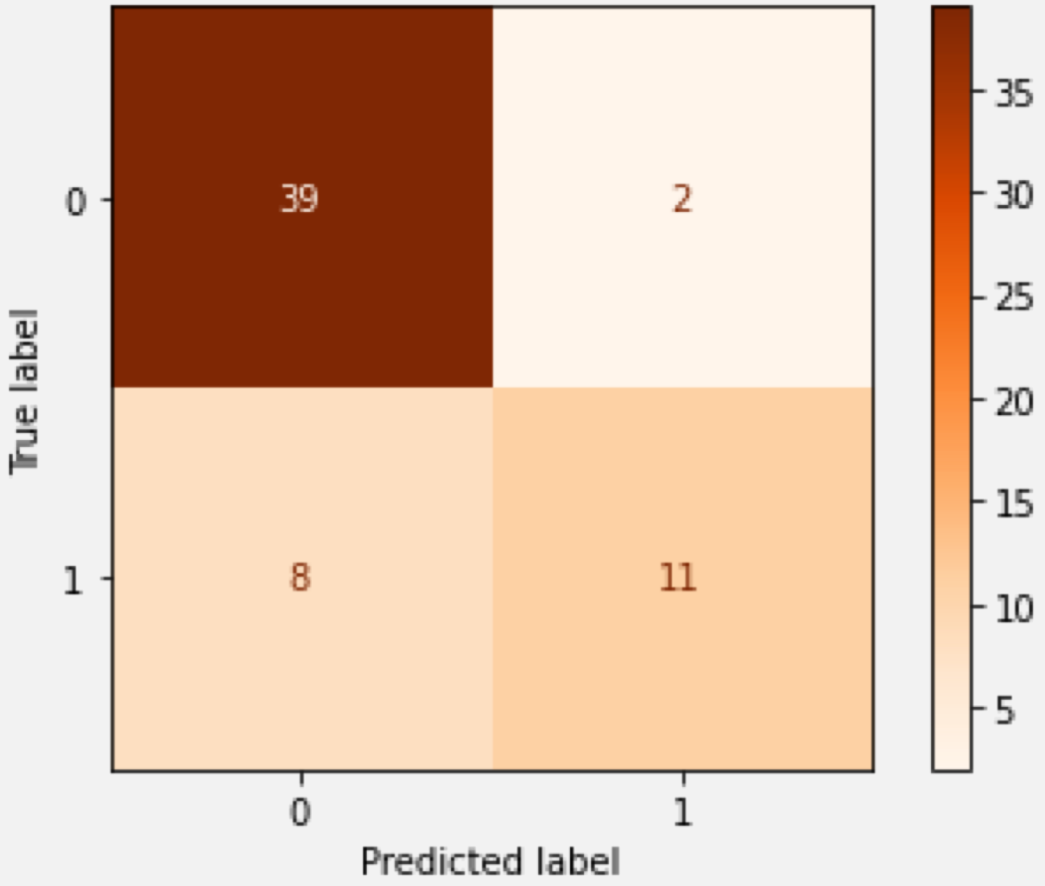
[61]
```
# create model objects with best_parameters
# use parameters from 'grid_search.best_params_' for each model
log_reg_best = LogisticRegression(random_state=  42,
                                  C= 1,
                                  intercept_scaling =5.0,
                                  max_iter = 100,
                                  multi_class = 'ovr',
                                  penalty = 'l1',
                                  solver = 'liblinear' ).fit(cardio_prepared, cardio_target)

random_best = RandomForestClassifier(random_state=42,
                                     max_depth = 2,
                                     max_features = 6,
                                     min_samples_leaf = 4,
                                     min_samples_split = 2,
                                     n_estimators = 10).fit(cardio_prepared, cardio_target)

dec_tree_best = DecisionTreeClassifier(random_state=42, max_depth =1, max_features = 8).fit(cardio_prepared, cardio_target)

SVC_best = SVC(C = 1, cache_size = 100, degree = 2, kernel = 'linear', max_iter = -1).fit(cardio_prepared, cardio_target)
```

Part 4: Create a ROC plot that shows all the trained ML classifier's performance. Select the best one for testing.





Part 4: Create a ROC plot that shows all the trained ML classifier's performance. Select the best one for testing.



Attribute	Model	CV	Accuracy
Creatinine Phosphokinase	Random Forest Classifier	10	0.833

	precision	recall	f1-score	support
0	0.83	0.95	0.89	41
1	0.85	0.58	0.69	19
accuracy			0.83	60
macro avg	0.84	0.77	0.79	60
weighted avg	0.83	0.83	0.82	60