# Exploratory Data Analysis, Visualization, and Prediction Model in Python

## Using Pandas, Matplotlib, Seaborn, and Scikit\_learn Libraries in Python

By Rashida Nasrin Sucky

Apr 13 2021

Fonte: <https://towardsdatascience.com/exploratory-data-analysis-visualization-and-prediction-model-in-python-241b954e1731>

Homem ao lado de água e montanha ao fundo

Descrição gerada automaticamente

Photo by [Kalen Emsley](https://unsplash.com/@kalenemsley?utm_source=medium&utm_medium=referral) on [Unsplash](https://unsplash.com?utm_source=medium&utm_medium=referral)

This article focuses on a data storytelling project. In other words Exploratory data analysis. After looking at a big dataset or even a small dataset, it is hard to make sense of it right away. It needs effort, more work, and analysis to extract some meaningful information from that dataset.

In this article, we will take a dataset and use some popular python libraries like Numpy, Pandas, Matplotlib, Seaborn to find some meaningful information from it. And at the end, we will run a prediction model from the scikit-learn library.

As a data scientist or a data analyst, you may have to deal with data where subject matter is not so well known to you. This dataset might be one of those kind. Lots of the columns are medical terms. But that should not be a big problem. It is still possible to explore the dataset with the help of great tools and techniques.

## This article will cover:

* How to approach a dataset to extract meaningful information from it or understand the data
* Use of a machine learning model for prediction

## Extracting Information From the Dataset

The dataset that will be used here is called the “heart failure clinical records” dataset. Please feel free to download the dataset from [Kaggle](https://www.kaggle.com/andrewmvd/heart-failure-clinical-data) and follow along.

Let’s import the necessary packages and the dataset in the jupyter notebook environment:

import pandas as pd  
import numpy as np  
import matplotlib.pyplot as pltdf = pd.read\_csv("heart\_failure\_clinical\_records\_dataset.csv")

Dataset is a bit bigger for showing a screenshot here. The dataset has 299 rows of data and here are the columns of the dataset:

df.columns

Output:

Index(['age', 'anaemia', 'creatinine\_phosphokinase', 'diabetes',  
 'ejection\_fraction', 'high\_blood\_pressure', 'platelets',  
 'serum\_creatinine', 'serum\_sodium', 'sex', 'smoking', 'time',  
 'DEATH\_EVENT', 'sex1', 'death'],  
 dtype='object')

Here, ‘age’, ‘creatinine\_phosphokinase’, ‘ejection\_fraction’, ‘platelets’, ‘serum\_creatinine’, ‘serum\_sodium’, ‘time’ are the continuous variables.

And ‘anaemia’, ‘diabetes’, ‘high\_blood\_pressure’, ‘sex’, ‘smoking’, ‘DEATH\_EVENT’ are the categorical variables.

All the categorical variables have only 0 and 1 values. So, ‘sex’ only says if it is a male or female, ‘high\_blood\_pressure’ says if the person has high blood pressure or not, ‘anaemia’ says if the person is suffering from anaemia or not.

I like to start most of the EDA project by observing the distribution of the continuous variables.

df[['age', 'creatinine\_phosphokinase',

'ejection\_fraction', 'platelets',

'serum\_creatinine', 'serum\_sodium']].hist(bins=20,

figsize=(15, 15))

plt.show()

Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente com confiança média

It shows at a glance where the majority population lies and the nature of the distribution.

To get an understanding of data, it is helpful to find some commonly used descriptive variable data such as mean, median, max, min, std, and quartiles.

continous\_var = ['age', 'creatinine\_phosphokinase',

'ejection\_fraction', 'platelets',

'serum\_creatinine', 'serum\_sodium']

df[continous\_var].describe()

Tabela

Descrição gerada automaticamente

Now as we have seen the distribution separately and statistical parameters, tt will be nice to see how each of these variables relates to the ‘DEATH\_EVENT’. This column has 0 and 1 values. I will change it to ‘yes’ and ‘no’. I also want to change the ‘sex’ column and replace 0 and 1 with ‘male’ and ‘female’.

df['sex1'] = df['sex'].replace({1: "Male", 0: "Female"})

df['death'] = df['DEATH\_EVENT'].replace({1: "yes", 0: "no"})

If you check the columns now, you will see two additional columns: ‘sex1’ and ‘death’ in the dataset. In the next plot, we will see a pairplot that will show the relationship between each of the continuous variables with the rest of them. We will also use a different color for death events.

sns.pairplot(df[["creatinine\_phosphokinase", "ejection\_fraction",

"platelets", "serum\_creatinine",

"serum\_sodium", "time", "death"]], hue = "death",

diag\_kind='kde', kind='scatter', palette='husl') plt.show()

Gráfico de dispersão

Descrição gerada automaticamente

Here, the red color shows the death event and the green color represents no death. This plot shows how each of these variables is segregated between death events. Besides, the scatter plot, the density plot in between shows a clear distinction of data for death events and no death events. Boxplots will give a little more clarity on that:

continous\_var = ['age', 'creatinine\_phosphokinase',

'ejection\_fraction', 'platelets', 'serum\_creatinine', 'serum\_sodium']

plt.figure(figsize=(16, 25))

for i, col in enumerate(continous\_var):

plt.subplot(6, 4, i\*2+1)  
 plt.subplots\_adjust(hspace =.25, wspace=.3)

plt.grid(True)

plt.title(col)

sns.kdeplot(df.loc[df["death"]=='no', col], label="alive", color = "green", shade=True, kernel='gau', cut=0)

sns.kdeplot(df.loc[df["death"]=='yes', col], label="dead", color = "red", shade=True, kernel='gau', cut=0)

plt.subplot(6, 4, i\*2+2)   
 sns.boxplot(y = col, data = df, x="death", palette = ["green", "red"])

Forma

Descrição gerada automaticamente

Some numeric data of the mean of each continuous variable for death events and no death events is necessary for a good report. Some solid numeric data help with a discussion. Here we will find the mean and median of each continuous variable for death events and no death events. As we saw in the distribution and density plots before, not all the variables are normally distributed. Some are skewed. So, mean alone will not be representative for each one.

y = df.groupby("death")["creatinine\_phosphokinase", "ejection\_fraction", "platelets", "serum\_creatinine",

"serum\_sodium", "time"].agg([np.mean, np.median])

y

Tabela

Descrição gerada automaticamente

Looks like the ‘time’ variable is very different between the death events.

We may explore some more on the ‘time’ variable later.

Let’s see if high blood pressure plays any role in deaths by gender.

df.groupby(['sex1', 'high\_blood\_pressure', 'death']).size().unstack().fillna(0).apply(lambda x: x/x.sum(), axis=1)

Output:

Tabela

Descrição gerada automaticamente

Now, you see some differences in proportions. But the difference is more in high blood pressure conditions than in gender. But even in high blood pressure conditions also the difference is not too drastic. In the female population, 28% of deaths happened in females with no high blood pressure and 39% of deaths happened in females with high blood pressure. It looks like a significant difference apparently. But not too drastic either. It requires some statistical inference to make a better conclusion which is not the scope of this article.

Other than the ‘death’ variable, we have five other categorical variables in this dataset. It is worth examining their relationship with the ‘death’ variable. I will use barplot or in the seaborn library, it is called the ‘countplot’ to do that.

binary\_var = ['anaemia', 'diabetes', 'high\_blood\_pressure',

'sex1', 'smoking']

plt.figure(figsize=(13, 9))

for i, var in enumerate(binary\_var):

plt.subplot(2, 3, i+1)

plt.title(var, fontsize=14)

plt.xlabel(var, fontsize=12)

plt.ylabel("Count", fontsize=12)

plt.subplots\_adjust(hspace = 0.4, wspace = 0.3)

sns.countplot(data= df, x = var, hue="death", palette = ['gray', "coral"])

Output:

Gráfico, Gráfico de barras

Descrição gerada automaticamente

In the plot above, it shows clearly that there is a difference in the number of death events between different sex(only male and female in this dataset), high\_blood pressure, smoking, and diabetes status. But at the same time, it also shows that the dataset is not balanced in terms of the number of people who smoke and do not smoke or the number of people having diabetes or not, or the number of males and the number of females. So, looking at the proportion will give us a clear idea.

At the moment, I can think of taking crosstab between each of these variables and the death variable. Let's start with the sex1 variable:

x = pd.crosstab(df["sex1"], df['death'])

x

Output:

Tabela

Descrição gerada automaticamente

You can see the numbers. How many deaths happen in the male population and how many deaths happened in the female population. But proportions will be more informative in this case. Because clearly number of male and the number of females are not the same.

x.apply(lambda z: z/z.sum(), axis=1)

Output:

Texto

Descrição gerada automaticamente com confiança média

Look! The deaths in male and female population both are approximately 32%. So they are the same. I did the same for the other four categorical variables as well.

Anaemia vs Death

Tabela

Descrição gerada automaticamente

The proportion of death for people with anemia is a bit higher.

Diabetes vs Death

Interface gráfica do usuário, Texto, chat ou mensagem de texto

Descrição gerada automaticamente com confiança média

Here, the proportion of death for people with diabetes and with no diabetes is exactly the same.

Smoking vs Death

Texto

Descrição gerada automaticamente com confiança média

The proportion of death for people with smoking and with no smoking habit are almost the same.

High Blood Pressure vs Death

Tabela

Descrição gerada automaticamente com confiança média

Here, the proportion of death is higher in people with high blood pressure.

In all the above analyses, we only tried to determine all the other variable’s relationship with the ‘death’ variable. We can do much more than that. Let’s try to see if we can extract some interesting information.

The next, element is a violin plot that shows the distribution of ‘time’ across smoking and no smoking males and females.

plt.figure(figsize=(8, 6))

a = sns.violinplot(df.smoking, df.time, hue=df.sex1, split=True) plt.title("Smoking vs Time Segregated by Gender", fontsize=14) plt.xlabel("Smoking", fontsize=12)

plt.ylabel("Time", fontsize=12)

plt.show()

Uma imagem contendo Gráfico

Descrição gerada automaticamente

In the case of no smoking people, males and females have the same distribution. On the other hand, in the case of smoking people, the distribution for males and females is very different. Most females lie in a narrow range from about 0 to 140. Whereas the male population has a range from -50 to 350.

Now I want to see the relation between ‘ejection\_fraction’ and ‘time’ segregated by ‘death’.

sns.lmplot(x="ejection\_fraction", y="time",

hue="death", data=df, scatter\_kws=dict(s=40,

linewidths=0.7, edgecolors='black'))

plt.xlabel("Ejection Fraction", fontsize=12)

plt.ylabel("Time", fontsize=12)

plt.title("Ejection fraction vs time segregated by death",fontsize=14)

plt.show()

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

This plot does not provide too much information for now. But you can see the regression line and the confidence band. As expected the confidence band is narrower in the middle where the density of data is higher and wider in the sides where the density of data is lower.

In the next plot, let’s see another comparison between the male and female population. How ‘time’ changes with ‘age’:

fig = plt.figure(figsize=(20, 8), dpi=80)

g = sns.lmplot(x='age', y='time',

data = df,

robust = True,

palette="Set1", col="sex1",

scatter\_kws=dict(s=60, linewidths=0.7, edgecolors="black"))

for ax in g.axes.flat:

ax.set\_title(ax.get\_title(), fontsize='x-large')

ax.set\_ylabel(ax.get\_ylabel(), fontsize='x-large')

ax.set\_xlabel(ax.get\_xlabel(), fontsize='x-large')

Gráfico, Gráfico de dispersão

Descrição gerada automaticamente

Notice the regression line here, For the male population, the regression line is much steeper. With ‘age’, ‘time’ goes down.

In most plots above, we tried to find out the relationships between variables. A heat map that shows the correlation amongst all the variables is very helpful. Heat maps are used in feature selection for machine learning and also in data analytics to understand the correlation between the variables.

plt.figure(figsize=(10, 10))

sns.heatmap(df.corr(), annot=True, linewidths=0.5, cmap="crest")

plt.show()

Gráfico

Descrição gerada automaticamente

You will see the use of those correlations in the next section.

## Prediction of Death

Using all the variables in the dataset, we can train a machine learning model to predict death. For this project, I will simply import a machine learning model from the scikit\_learn library and use it.

## Data preparation

In this dataset, all the columns were originally numeric which required for a machine learning model. We created two columns with strings in the beginning. Those columns need to be deleted.

df = df.drop(columns=['sex1', 'death'])

Also, I will use dropna() function on the dataset. The dataset is pretty clean. But by any chance, if there is any null value, it will delete those rows.

df = df.dropna()

As I mentioned before, the correlation variable above can be used for feature selection for machine learning. Notice the correlations in the heatmap between the DEATH\_EVENT and the other variables in the last column of the heat map. The correlation is very low between the DEATH\_EVENT and ‘anaemia’, ‘diabetes’, ‘sex’, ‘smoking’, ‘creatinine\_phosphokinase’.

I will simply drop those columns from the dataset.

df = df.drop(columns=['anaemia', 'diabetes', 'sex', 'smoking',

'creatinine\_phosphokinase'])

This is a good idea to bring all the variables in the dataset on the same scale. Look, different variables have different ranges. Like ‘platelets’ have a very high range. On the other hand, serum\_creatinine has a very low range. To bring all the continuous variable ranges similar, I will divide each of them by their maximum value.

Before that, I will copy the dataset df into the variable df2 to keep the original values intact.

df2 = df

continuous\_var = ['age', 'ejection\_fraction', 'platelets', 'serum\_creatinine', 'serum\_sodium']

for i in continous\_var:

df2[i] = df2[i]/max(df2[i])

Categorical variables do not need to be changed. They are already 0 or 1 values.

This machine learning model will be used to predict the DEATH\_EVENT. So, the DEATH\_EVENT is the output variable.

y = df2['DEATH\_EVENT']

I will use the rest of the variables as input variables. That the DEATH\_EVENT is dropped from the df2 that will be the input variables list.

X = df2.drop(columns=['DEATH\_EVENT'])

## Separation of Training and Test Data

One last step before training the model. Usually, a part of the data is kept separate from the training data in machine learning. So that after training the model, you can check the model with some data that is unseen by the model. But the labels or the outputs are known to you.

The scikit\_learn library has a train\_test\_split function for that.

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.25, random\_state=22)

## Using the Decision Tree Classifier

I used a decision tree classifier for this example. Here is the classifier and the results:

clf\_tree = DecisionTreeClassifier(random\_state=21, max\_depth = 7, max\_leaf\_nodes=6).fit(X\_train,y\_train)

y\_pred = clf\_tree.predict(X\_test)

print("Accuracy:",metrics.accuracy\_score(y\_test,y\_pred)) print("Precision:",metrics.precision\_score(y\_test,y\_pred,pos\_label=0)) print("Recall:",metrics.recall\_score(y\_test,y\_pred,pos\_label=0)) print("F Score:",metrics.f1\_score(y\_test,y\_pred,pos\_label=0)) print("Confusion Matrix:\n",metrics.confusion\_matrix(y\_test,y\_pred))

Output:

Accuracy: 0.84  
Precision: 0.9215686274509803  
Recall: 0.8545454545454545  
F Score: 0.8867924528301887  
Confusion Matrix:  
 [[47 8]  
 [ 4 16]]

Alternative code:

from sklearn.tree import DecisionTreeClassifier

from sklearn.metrics import accuracy\_score, precision\_score, recall\_score, f1\_score, confusion\_matrix

clf\_tree = DecisionTreeClassifier(random\_state=21, max\_depth = 7, max\_leaf\_nodes=6).fit(X\_train,y\_train)

y\_pred = clf\_tree.predict(X\_test)

print("Accuracy:",accuracy\_score(y\_test,y\_pred))

print("Precision:",precision\_score(y\_test,y\_pred,pos\_label=0))

print("Recall:",recall\_score(y\_test,y\_pred,pos\_label=0))

print("F Score:",f1\_score(y\_test,y\_pred,pos\_label=0))

print("Confusion Matrix:\n",confusion\_matrix(y\_test,y\_pred))

Output:

Accuracy: 0.8133333333333334

Precision: 0.9361702127659575

Recall: 0.8

F Score: 0.8627450980392157

Confusion Matrix:

[[44 11]

[ 3 17]]

The accuracy rate is 0.8133or 81%.

Also, the F Score is 0.86274 or 0.86. The closer the F Score to 1, the better the model is.

Now if you know the ‘age’, ‘ejection\_fraction’, ‘high\_blood\_pressure’, ‘platelets’, ‘serum\_creatinine’, ‘serum\_sodium’, ‘time’, about a person, you can predict if that person is dead or alive.

And the prediction is correct almost 81% of the time.

## Conclusion

In this demonstration, I tried to show you some techniques to understand this dataset and run a prediction model as well. There are many different ways to approach an exploratory data analysis task. This was my choice for this demonstration.

Feel free to follow!