Data Science Techniques and Applications
Coursework II

Baran Buluttekin
13153116

March 23, 2019

We can start of our analysis by getting the dataset. Corresponding Heart Diseases dataset[1, 3] can be downloaded from kaggle by clicking download link in the website. When the download finished we received compressed version of file named *heart-disese-uci.zip*. Fallowing the uncompression of the file we will obtain csv file *heart.csv* which is in final format for analysis.

Next step is to loading python libraries to start the analysis. In this coursework fallowing open source libraries used:

1. **Pandas:** Tabular data manipulation library that will be use to load the data and cleaning/reshaping.[5]
2. **Matplotlib:** Will be used as main plotting library that has many pre build high level plotting tools.[4]
3. **Seaborn:** Python library for special plotting styles such as pairplots.[7]
4. **Scikit-Learn:** Open source machine learning library that includes various machine learning algorithms.[6]

As a first step libraries mentioned above loaded in python in the fallowing format.

```
import pandas as pd
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from sklearn.decomposition import PCA
import seaborn as sns
```

With the help of pandas library we can load the dataset and view first 5 rows to check if the data frame is in expected format.

```
df = pd.read_csv("heart.csv")
df.head()
```

age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
63.0	1.0	3.0	145.0	233.0	1.0	0.0	150.0	0.0	2.3	0.0	0.0	1.0	1.0
37.0	1.0	2.0	130.0	250.0	0.0	1.0	187.0	0.0	3.5	0.0	0.0	2.0	1.0
41.0	0.0	1.0	130.0	204.0	0.0	0.0	172.0	0.0	1.4	2.0	0.0	2.0	1.0
56.0	1.0	1.0	120.0	236.0	0.0	1.0	178.0	0.0	0.8	2.0	0.0	2.0	1.0
57.0	0.0	0.0	120.0	354.0	0.0	1.0	163.0	1.0	0.6	2.0	0.0	2.0	1.0

Fallowing the guide of coursework, I choose "age", "chol" and "trestbps" as an important dimensions. Description of the columns from the coursework I is given below as a reminder.

1. **age**: Age in years (max:77, min:29)
2. **chol**: cholesterol level (mg/dl)
3. **trestbps**: Blood pressure recording for resting

In order to get a better grasp of the data distribution with these three variables, I will use pairplot. Pairplot will help us see different scatterplot variation of these three variable with each other and with relation to the target outcome of the dataset.

```
selected_dims = ["age", "chol", "trestbps", "target"]
sns.pairplot(df[selected_dims], hue="target")
```

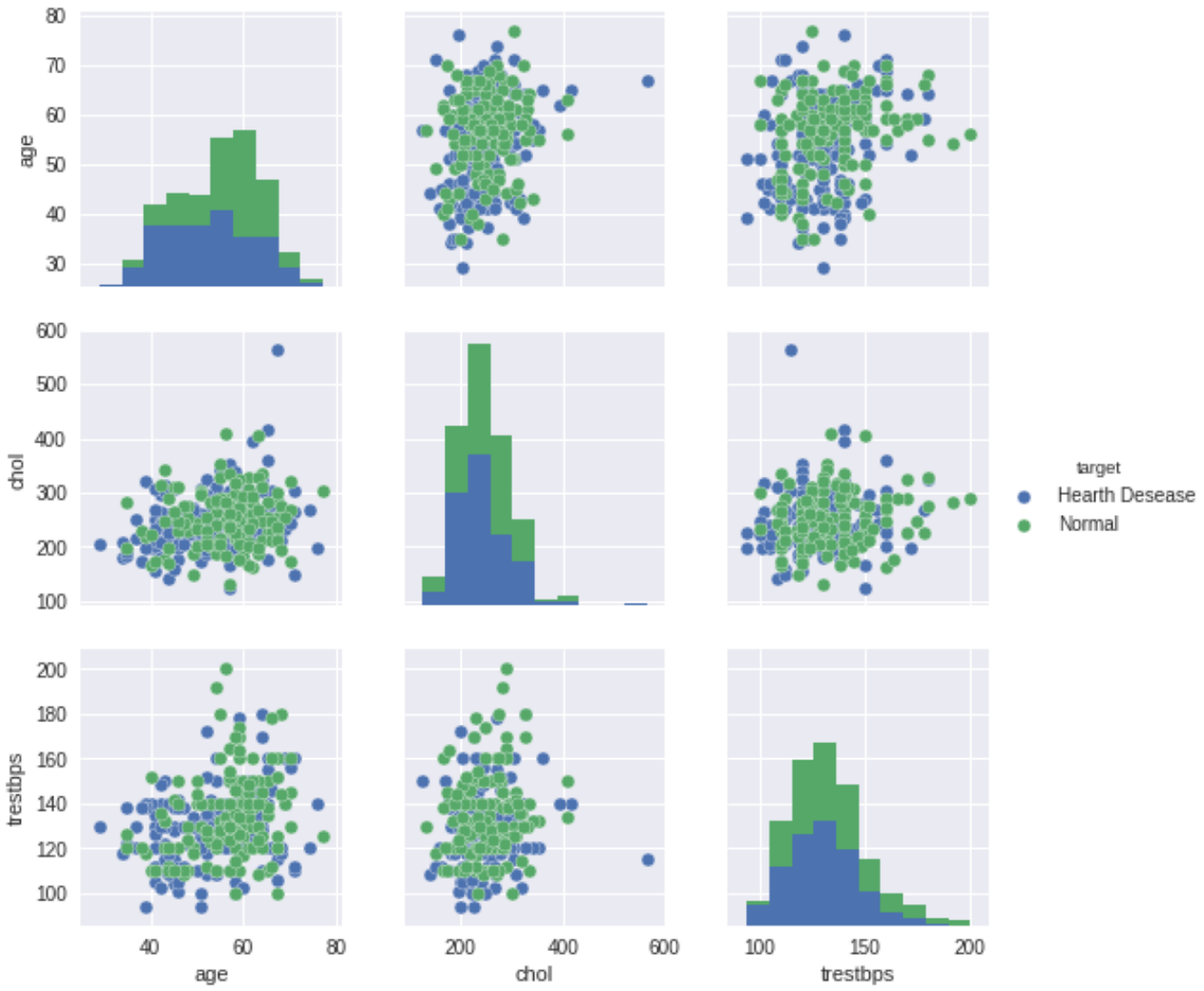

Figure 1: Pairplot with chosen dimensions.

We can also plot these 3 dimensions in 3D plot to see their relation to target variable. I created helper function quickly plot 3D plots given dimensions in dataset. Code for the function follows.

```
def plot_3d(dataframe, x_label, y_label, z_label, target_label, title):
    labelTp = [("Normal", 0), ("Hearth Desease", 1)]
    fig = plt.figure(1, figsize=(8, 6))
    ax = Axes3D(fig, elev=-140, azim=110)
    ax.scatter(dataframe[x_label], dataframe[y_label], dataframe[z_label], c
                                          =dataframe[target_label],
               cmap=plt.cm.Set1, edgecolor='k', s=40)
    ax.set_title(title)
    ax.set_xlabel(x_label)
```

```
    ax.w_xaxis.set_ticklabels([])
    ax.set_ylabel(y_label)
    ax.w_yaxis.set_ticklabels([])
    ax.set_zlabel(z_label)
    ax.w_zaxis.set_ticklabels([])

    sc = ax.scatter(dataframe[x_label], dataframe[y_label], dataframe[
                                            z_label], c=dataframe[
                                            target_label], cmap="Spectral",
                                            edgecolor='k')

    clrs = [sc.cmap(sc.norm(x)) for x in [1, 0]]
    custom_l = [plt.Line2D([],[], ls="", marker='.',
                    mec='k', mfc=i, mew=.1, ms=20) for i in clrs]
    lgd = ax.legend(custom_l, [l[0] for l in labelTp],
                loc='center left', bbox_to_anchor=(1.0, .5))
    plt.show()
```

We can observe their relationships by calling the function with the *"age"*, *"chol"* and *"trestbps"*.

```
plot_3d(df, "age", "chol", "trestbps", "target", "Heart Desease clustering
                                        by 3 dimensions")
```

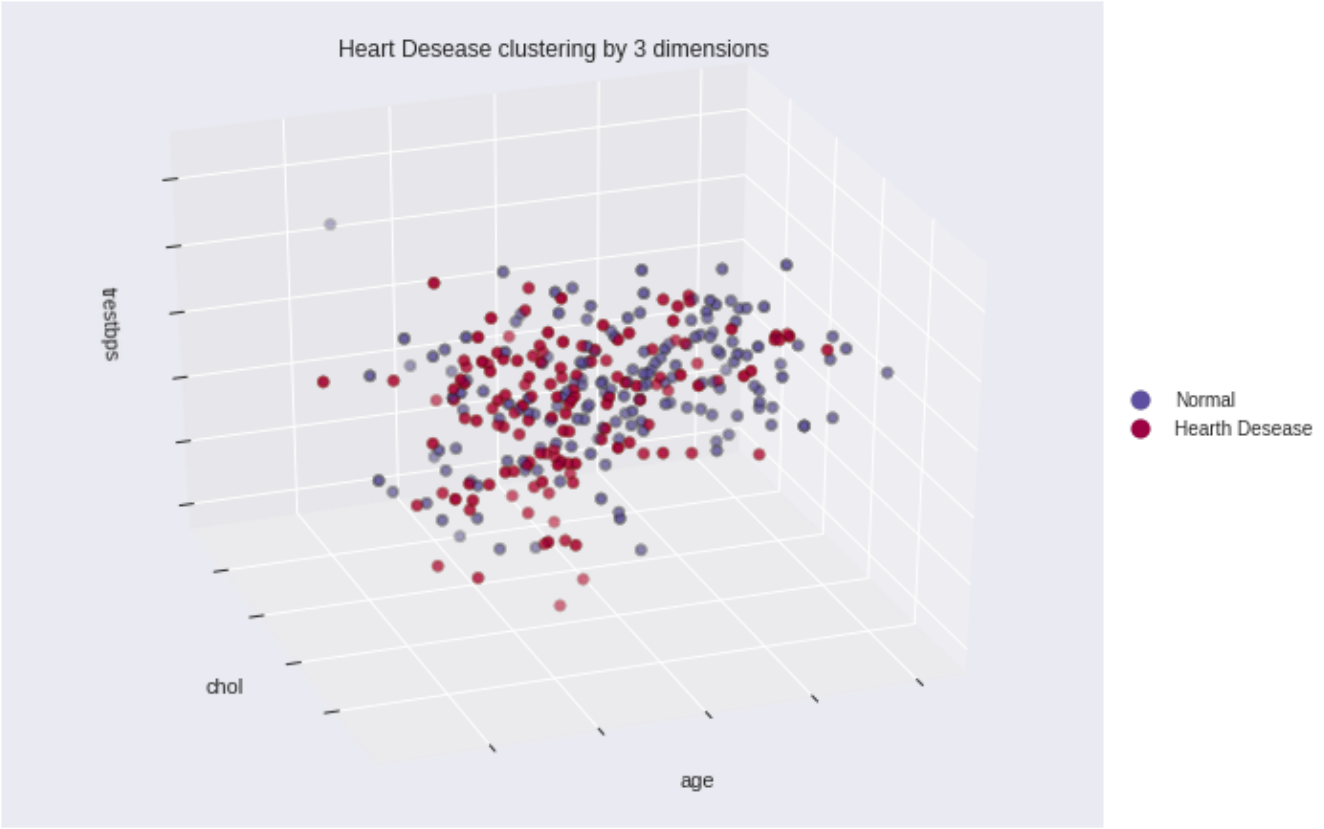

Figure 2: 3D graph of the selected dimensions.

Applying PCA On Dataset

Principle Component Analysis can be compute by computing standard matrix factorization technique called *Singular Value Decomposition* (SVD) that can essentially decompose training matrix X into dot product of three matrices $U \cdot \sum \cdot V^H$ where V^H is the eigenvectors.

I will be using Scikit Learn PCA class to obtain principle component analysis of the dataset. Using Scikit Learn is relatively straight forward since the library has unified API for all the algorithms. I will start with initiating class instance then *fit_transform* method can be called on that instance while passing dataset. User can specify the *n_components* they wish to receive in the class instance as an attribute and the result of PCA will be same number of dimensions for decomposed matrix. In the fallowing example I choose '3' components and received the three dimensional PCA matrix.

```
pca = PCA(n_components=3)
df_reduced = pca.fit_transform(df[df.columns[:-1]])
df_reduced[:5]
```

```
array([[-12.26734484,   2.87383781,  14.96987876],
       [  2.69013712, -39.87137362,   0.8778823 ],
       [-42.95021407, -23.63681988,   1.75944589],
       [-10.94475636, -28.43803577,  -7.04418048],
       [106.97905333, -15.87446784, -14.8615748 ]])
```

Yet choosing arbitrary number for components not a good practice and do not tell us anything about nature of the decomposed matrix. It is usually more sensible to choose number of components based on the percentage of variance (e.g., 95%) of the data. Scikit Learn has useful *explained_variance_ratio_* attribute available which is generated from fitting the data. This attribute give us the percentage of explained variance in each of the components of decomposition. For above example we can retrieve the explained variance with the code below.

```
pca.explained_variance_ratio_

>>> array([0.7475642 , 0.15037022, 0.08459685])
```

This result tell us that the first eigenvalues of the decompose matrix explains $\approx 75\%$, second eigenvalues $\approx 15\%$ and third eigenvalues $\approx 8\%$ of the variance.

In order to choose number of components that will give us the total explained variance of 95% we can fit a PCA instance without explicitly specifying the number of components. This instance will return us maximum number of eigenvalues available and their explained variance where we can calculate desired level of explained variance and associated number of components.

```
pca_search = PCA()
pca_search.fit_transform(df[df.columns[:-1]])
cum_sum = np.cumsum(pca_search.explained_variance_ratio_)
confidence_degree = np.argmax(cum_sum >= 0.95) + 1
confidence_degree

>>> 3
```

We can observe from the example above in fact three components are enough to reach over 95% variance in this dataset. Another way to obtain PCA values with same ratio of variance is to pass the ratio to PCA instance as a number of components. For example *PCA(n_components=0.95)* will also give us same result. In some cases it might be useful to visualize the number of components as a variable of explained variance and choose the desired number of components from the graph.

```
plt.plot(list(range(1, len(cum_sum) + 1)), cum_sum)
plt.xlabel("Number of components")
```

```
    plt.ylabel("Explained variance")
```

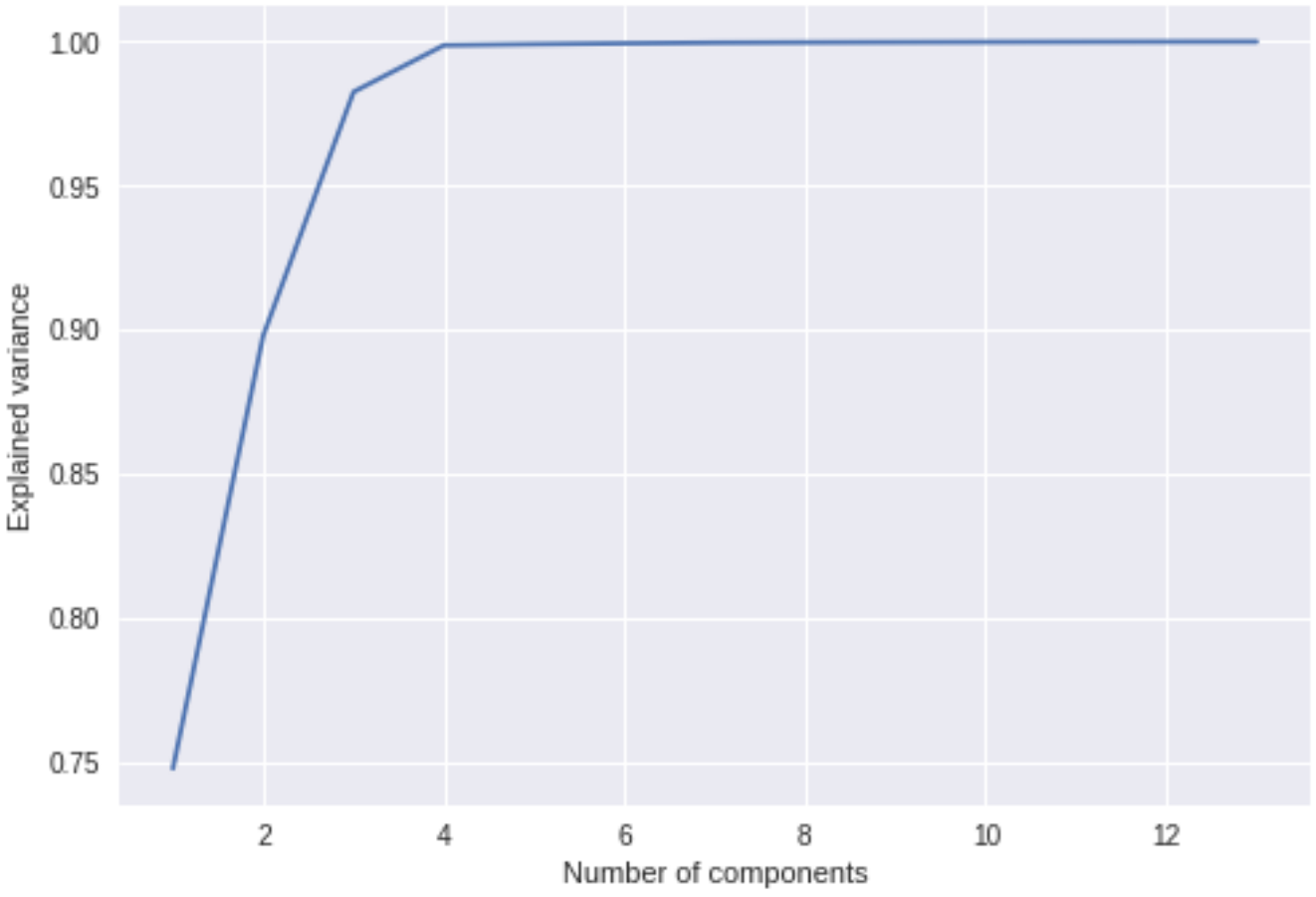

Figure 3: Ratio of explained variance in respect to number of components.

Given that we choose the number of components, we can plot the PCA values and observe how they represent the dataset.

```
    labelTp = [("Normal", 0), ("Hearth Desease", 1)]
    fig = plt.figure(1, figsize=(10, 6))
    ax = Axes3D(fig, elev=-140, azim=110)
    ax.scatter(df_reduced[:, 0], df_reduced[:, 1], df_reduced[:, 2], c=y,
               cmap=plt.cm.Set1, edgecolor='k', s=40)
    ax.set_title("PCA with 3 components")
    ax.set_xlabel("1st eigenvector")
    ax.w_xaxis.set_ticklabels([])
    ax.set_ylabel("2nd eigenvector")
    ax.w_yaxis.set_ticklabels([])
    ax.set_zlabel("3rd eigenvector")
```

```
    ax.w_zaxis.set_ticklabels([])

    sc = ax.scatter(df_reduced[:, 0], df_reduced[:, 1], df_reduced[:, 2], c=y,
                                            cmap="Spectral", edgecolor='k')

    clrs = [sc.cmap(sc.norm(x)) for x in [1, 0]]
    custom_l = [plt.Line2D([],[], ls="", marker='.',
                    mec='k', mfc=i, mew=.1, ms=20) for i in clrs]
    ax.legend(custom_l, [l[0] for l in labelTp],
            loc='center left', bbox_to_anchor=(1.0, .5))
```

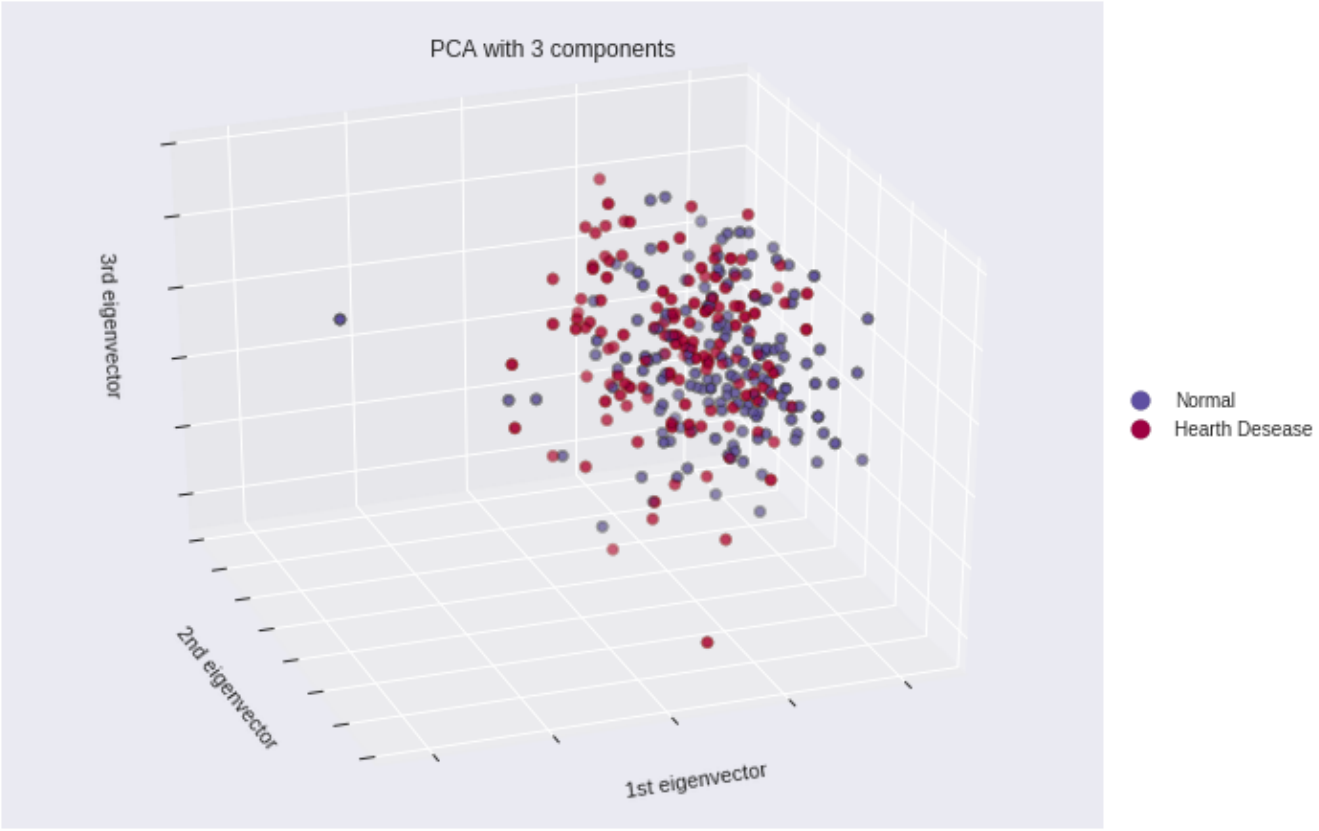

Figure 4: Heart Disease dataset with reduced dimensions.

Plot of PCA values clearly shows that even though complexity of the dataset reduced it did not helped data to be linearly separable. In some cases such as Swiss roll example[2] PCA can help increase linear or simpler decision boundaries to separate the data but it is not always the case and this generally depends on the data. In my dataset, inherent noise in the data we observed from the Figure 2 preserved in the Figure 4 and data remains linearly non-separable.

References

[1] Janosi et al. *UCI Machine Learning Repository*. 1988. URL: https://archive.ics.uci.edu/ml/datasets/Heart+Disease.

[2] Aurlien Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. 1st. O'Reilly Media, Inc., 2017, pp. 209–213. ISBN: 1491962291, 9781491962299.

[3] *Heart Disease UCI*. URL: https://www.kaggle.com/ronitf/heart-disease-uci.

[4] J. D. Hunter. "Matplotlib: A 2D Graphics Environment". In: *Computing in Science Engineering* 9.3 (May 2007), pp. 90–95. ISSN: 1521-9615. DOI: 10.1109/MCSE.2007.55.

[5] Wes McKinney. "Data Structures for Statistical Computing in Python". In: *Proceedings of the 9th Python in Science Conference*. Ed. by Stéfan van der Walt and Jarrod Millman. 2010, pp. 51–56.

[6] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[7] Michael Waskom et al. *mwaskom/seaborn: v0.8.1 (September 2017)*. Sept. 2017. DOI: 10.5281/zenodo.883859. URL: https://doi.org/10.5281/zenodo.883859.