1st Mandatory Assignment in STK-IN4300 by Jørn Eirik Betten

September 21, 2022

1 Summarization of the WBC data set.

I decided to take a closer look at the Wisconsin Breast Cancer data set, which can be downloaded from https://archive-beta.ics.uci.edu/ml/datasets/breast+cancer+wisconsin+diagnostic. The WBC data set contains data received from Dr. William H. Wolberg in the period between January 1989 to Nov 1991. Since jupyter notebook is something I normally don't write reports in, I will give a citation to Wolberg here:
O. L. Mangasarian and W. H. Wolberg: "Cancer diagnosis via linear programming", SIAM News, Volume 23, Number 5, September 1990, pp 1 and 18.

Table 1 displays a summary of the WBC data set.

[2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Adding path to location where I stored the data. I you wanna run this code,␣
 ↪you'll need to store
# the data in this location, then everything should run smoothly.
path ="/home/jeb/Documents/STKIN4300/MAs/MA1/data/WBCData/"

filename = "breast-cancer-wisconsin.data"
col_names = ["ID", "clump_thickness", "cell_size_uniformity",␣
 ↪"cell_shape_uniformity", "marginal_adhesion",
             "SE_cell_size", "bare_nuclei", "bland_chromatin",␣
 ↪"normal_nucleoli", "mitoses", "tumor_classification"]
data = pd.read_csv(path + filename)
data.columns = col_names
data
```

[2]:
```
          ID  clump_thickness  cell_size_uniformity  cell_shape_uniformity  \
0    1002945                5                     4                      4
1    1015425                3                     1                      1
2    1016277                6                     8                      8
3    1017023                4                     1                      1
4    1017122                8                    10                     10
..       …                …                     …                      …
```

```
693   776715              3                   1                      1
694   841769              2                   1                      1
695   888820              5                  10                     10
696   897471              4                   8                      6
697   897471              4                   8                      8

     marginal_adhesion  SE_cell_size bare_nuclei  bland_chromatin  \
0                    5             7          10                3
1                    1             2           2                3
2                    1             3           4                3
3                    3             2           1                3
4                    8             7          10                9
..                 …             …          …               …
693                  1             3           2                1
694                  1             2           1                1
695                  3             7           3                8
696                  4             3           4               10
697                  5             4           5               10

     normal_nucleoli  mitoses  tumor_classification
0                  2        1                     2
1                  1        1                     2
2                  7        1                     2
3                  1        1                     2
4                  7        1                     4
..               …      …                   …
693                1        1                     2
694                1        1                     2
695               10        2                     4
696                6        1                     4
697                4        1                     4

[698 rows x 11 columns]
```

There are two values in the tumor classification column: 2 and 4, these are respectively benign and malignent tumors.

[3]:
```python
# Need to replace three missing values in the bare nuclei column.
# These values are not used when calculating the estimates.
data = data.replace("?", np.nan)

# Running through the data, collecting wanted estimates.
for name in col_names:
    if name == "ID":
        print("ID is just the patient number. Moving on..")
    elif name == "target":
        num_mal_tumors = data[data["target"]==4].count()
        num_ben_tumors = data[data["target"]==2].count()

        print(f"The data set contains {num_mal_tumors[0]} people with malignent␣
↪tumors,")
        print(f"and {num_ben_tumors[0]} people with benign tumors.")

    else:
        col = data[name].astype(float)
        mean = col.mean()
        std = col.std()
        median = col.median()
        q1 = col.quantile(0.25)
        q3 = col.quantile(0.75)
        minimum = col.min()
        maximum = col.max()
        print(f"{name}:\n mean={mean:.1f} +-{std:.1f}, \n median={median:.
↪2f}({q1:.2f}-{q3:.2f}),\n min-max =({minimum:.0f}-{maximum:.0f})")
```

```
ID is just the patient number. Moving on..
clump_thickness:
 mean=4.4 +-2.8,
 median=4.00(2.00-6.00),
 min-max =(1-10)
cell_size_uniformity:
 mean=3.1 +-3.1,
 median=1.00(1.00-5.00),
 min-max =(1-10)
cell_shape_uniformity:
 mean=3.2 +-3.0,
 median=1.00(1.00-5.00),
 min-max =(1-10)
marginal_adhesion:
 mean=2.8 +-2.9,
 median=1.00(1.00-4.00),
 min-max =(1-10)
SE_cell_size:
 mean=3.2 +-2.2,
 median=2.00(2.00-4.00),
 min-max =(1-10)
bare_nuclei:
 mean=3.5 +-3.6,
 median=1.00(1.00-6.00),
 min-max =(1-10)
bland_chromatin:
 mean=3.4 +-2.4,
```

Ordinal continuous (N=698)	Mean (SD)	Median (IQR)	Min to Max
Clump Thickness	4.4 (2.8)	4 (2-6)	1-10
Uniformity of Cell Size	3.1 (3.1)	1 (1-5)	1-10
Uniformity of Cell Shape	3.2 (3.0)	1 (1-5)	1-10
Marginal Adhesion	2.8 (2.9)	1 (1-4)	1-10
Single Epithelial Cell Size	3.2 (2.2)	2 (2-4)	1-10
Bare Nuclei	3.6 (3.7)	1 (1-6)	1-10
Bland Chromatin	3.4 (2.4)	3 (2-5)	1-10
Normal Nucleoli	2.9 (3.1)	1 (1-4)	1-10
Mitoses	1.6 (1.7)	1 (1-1)	1-10
Categorical		Instances (%)	
Tumor Classification			
Benign		457 (65.5)	
Malignent		241 (34.5)	

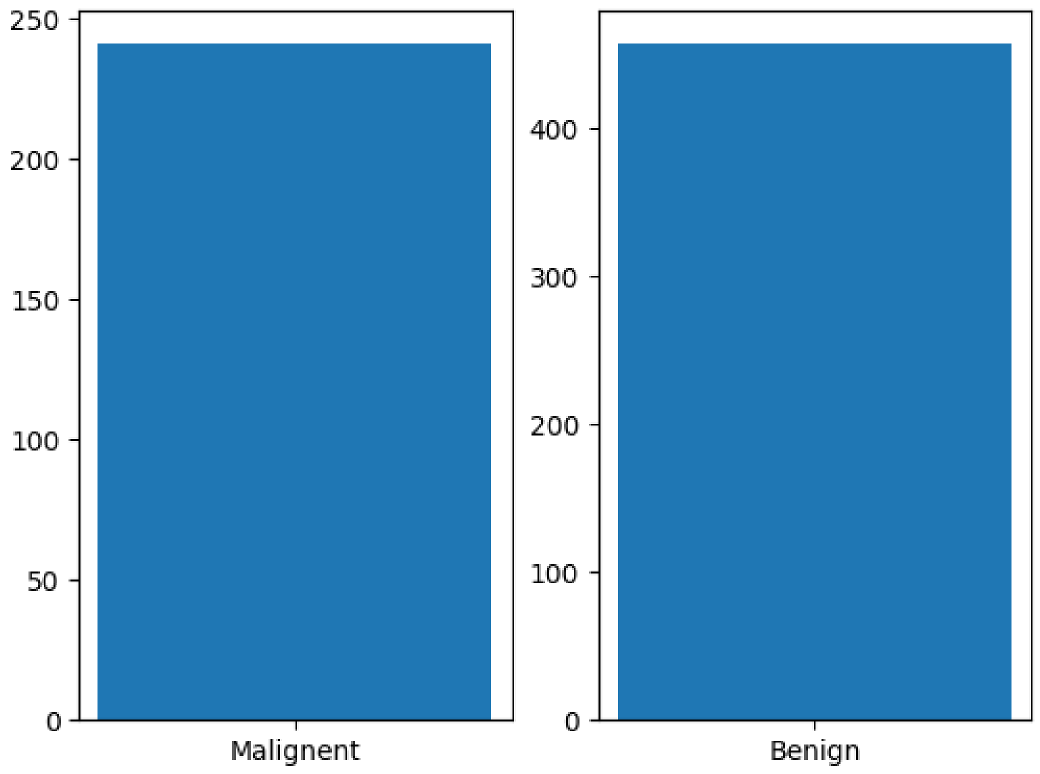
```
 median=3.00(2.00-5.00),
 min-max =(1-10)
normal_nucleoli:
 mean=2.9 +-3.1,
 median=1.00(1.00-4.00),
 min-max =(1-10)
mitoses:
 mean=1.6 +-1.7,
 median=1.00(1.00-1.00),
 min-max =(1-10)
tumor_classification:
 mean=2.7 +-1.0,
 median=2.00(2.00-4.00),
 min-max =(2-4)
```

Table 1: Summary of the WBC data set. SD: standard deviation, IQR: interquartile range, Min: minimum value, Max: maximum value.

2 Bad figures

A bad figure is a figure that does not give a good representation of the data, and/or is not aesthetically pleasing to the eye. My first figure is a barplot of the distribution of the tumor classifications within the WBC dataset. I have created an extreme example of bad figuring. I have created a figure with two subplots that each has its own axis scaling. They are of similar sizes because of the difference in the scaling of the axes. This figure is therefore bad in a scientific sense, as it is misleading the audience. It is also bad in an aesthetic sense, due to all the clutter. I have added another variable, the clump thickness, which is ordinal in nature, and divided into ten categories (which I have treated as a continuous specter). The figure I made below shows the distribution of the clump thickness among the gathered data. The barplot shows that most of the data collected is has clump thicknesses in the lower range (2-6). The problem I have this barplot is the clutter, and that it feels stretched in the x-direction. Another problem is that the information displayed may not be very interesting. In the next section I will display some improvements on these designs.

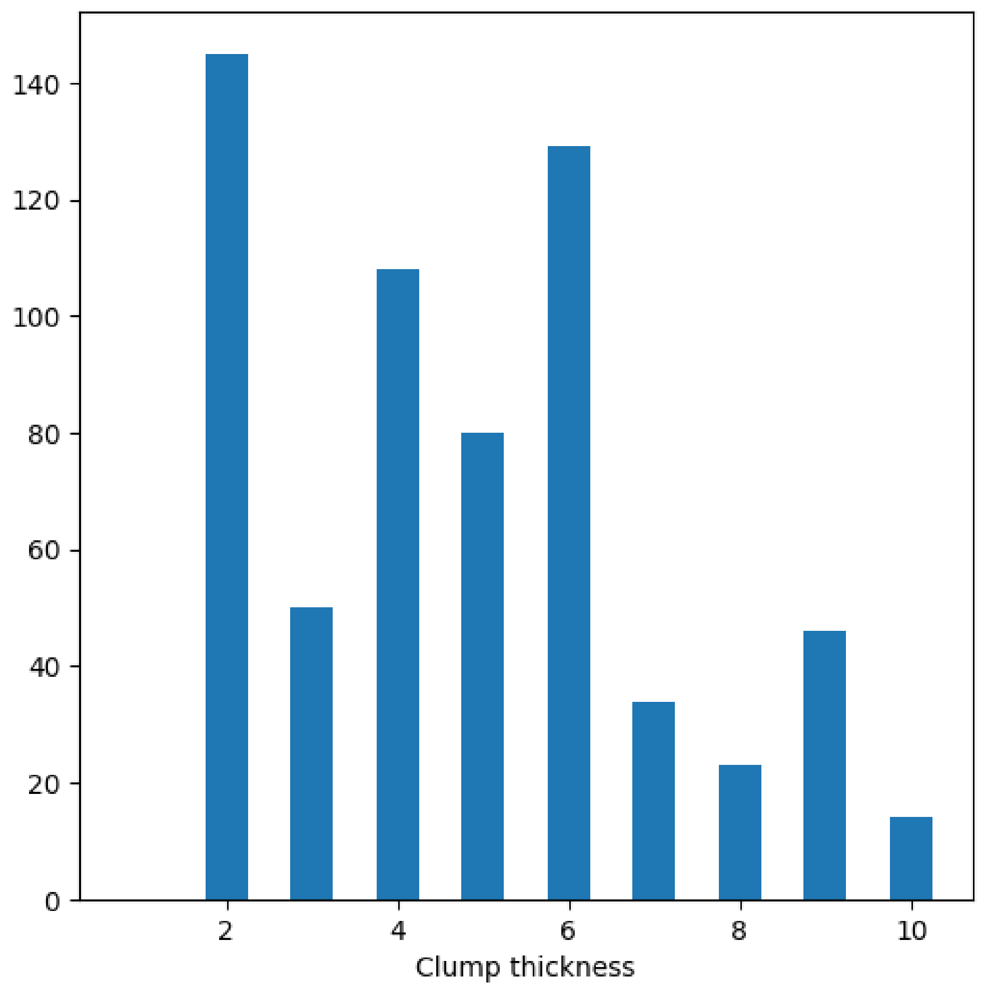
[4]:
```
# Plotting bad barplot of the tumor classifications
fig, ax = plt.subplots(nrows=1, ncols=2)
ax[1].bar(1, 457)
ax[0].bar(2, 241)
ax[1].set_xticks([1], labels=["Benign"])
ax[0].set_xticks([2], labels=["Malignent"])
#plt.xlabel("Tumor classification")
plt.show()
```



[30]:
```
values = np.linspace(1, 10, 10)
bars = np.zeros(10)
for i in range(1, 10, 1):
    bars[i] = data[data["clump_thickness"]==i].count()[0]



fig = plt.figure(figsize=(6,6))
plt.bar(values, bars, width=0.5)
plt.xlabel("Clump thickness")
plt.show()
```

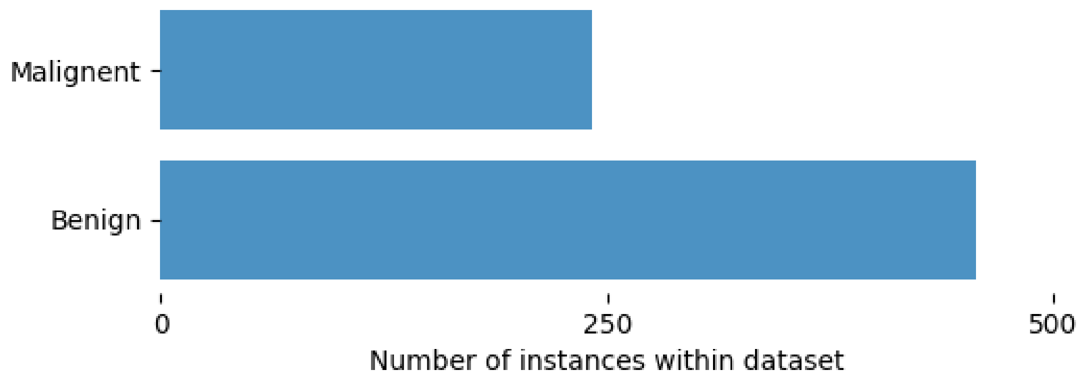


3 Good (at least better) figures

For displaying the distribution of benign and malignent tumors I have chosen to stay with the barplot. Since I only had two categories within the tumor classification, the size of the barplot is a bit awkward (not square). I have made two possible solutions where I have removed the clutter from the figure and plotted both bars on the same axis, which yields, in my opinion, a more aesthetically pleasing and more informative plot. The tab:blue color in matplotlib is the standard blue, which I have toned a bit down. The first one, which in my opinion is the best, is a horizontal barplot. I feel like the information displayed is compact and evident from the plot alone. The vertical barplot is similar, and displayed below. You can decide for yourself which one pleases you more.
In the case of the vizualization of the clump thickness, I wanted to make the plot more informative, and therefore plotted both the clump thickness and its tumor classification as a double barplot. I felt like a square design, as in its bad figure brother, but I removed the clutter, and toned down the color. Then I widened every bar such that the figure would appear more compact. The result is displayed in the last plot of this notebook, and is, in my opinion very informative and useful. It clearly displays correlation between the clump thickness and the tumor classification.

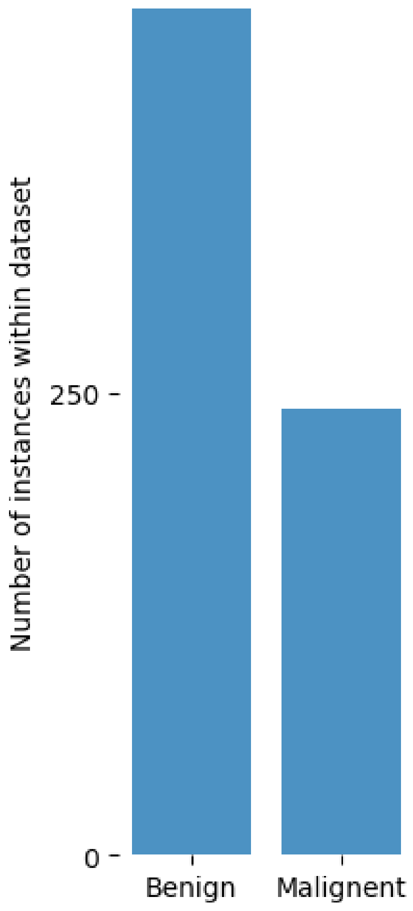
[27]:
```python
fig, ax = plt.subplots(figsize=(6, 2))
plt.barh(1, 457, alpha=0.8, color="tab:blue")
plt.barh(2, 241,  alpha=0.8, color="tab:blue")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.xlabel("Number of instances within dataset")
plt.yticks(ticks=[1, 2], labels=["Benign", "Malignent"])
plt.xticks(ticks=[0, 250, 500])

plt.show()
```



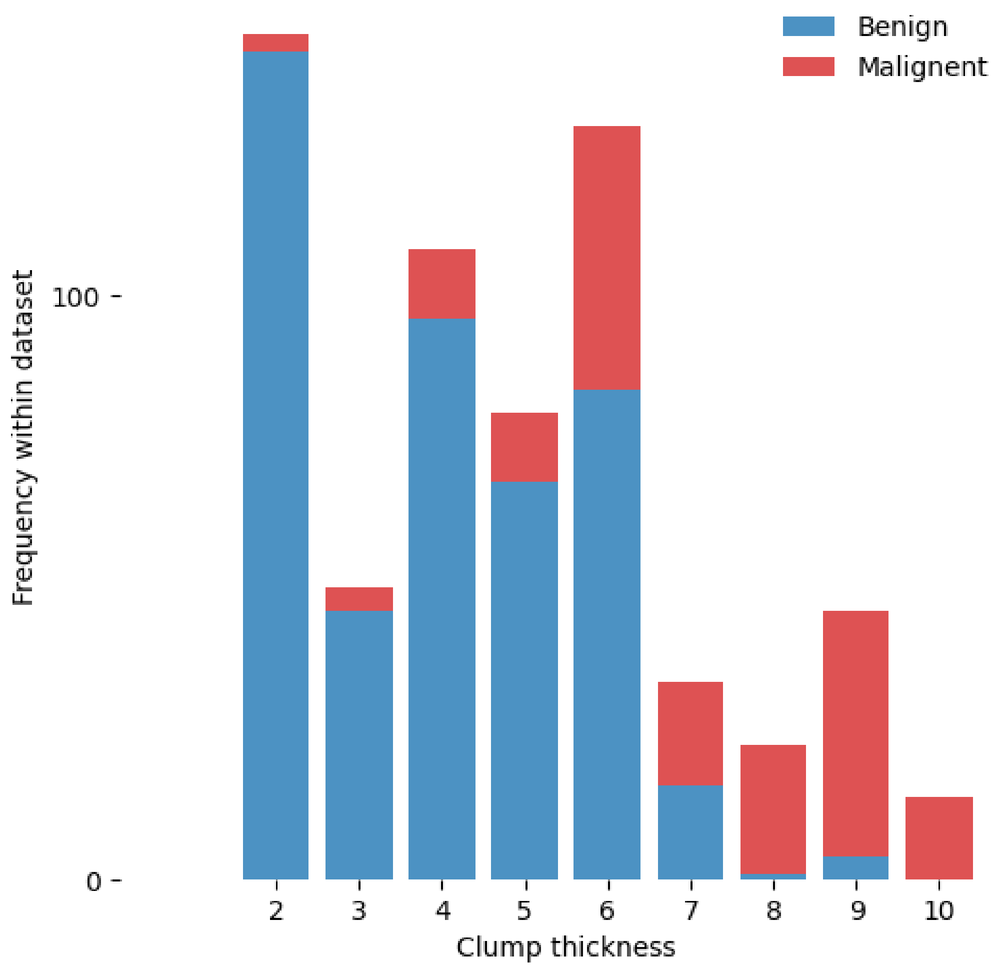
[28]:
```python
fig, ax = plt.subplots(figsize=(2, 6))
plt.bar(1, 457, alpha=0.8, color="tab:blue")
plt.bar(2, 241,  alpha=0.8, color="tab:blue")
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
plt.ylabel("Number of instances within dataset")
plt.xticks(ticks=[1, 2], labels=["Benign", "Malignent"])
plt.yticks(ticks=[0, 250])
#plt.grid(axis='x')

plt.show()
```



[29]:
```
values = np.linspace(1, 10, 10)
ben_bars = np.zeros(10)
mal_bars = np.zeros(10)
for i in range(1, 10, 1):
    ben_bars[i] = data[(data["clump_thickness"]==i) &␣
 ↪(data["tumor_classification"]==2)].count()[0]
    mal_bars[i] = data[(data["clump_thickness"]==i) &␣
 ↪(data["tumor_classification"]==4)].count()[0]


fig, ax = plt.subplots(figsize=(6,6))
plt.bar(values, ben_bars, color="tab:blue", label="Benign", alpha=0.8)
plt.bar(values, mal_bars, bottom=ben_bars, color="tab:red", label="Malignent",␣
 ↪alpha=0.8)
plt.xlabel("Clump thickness")
plt.ylabel("Frequency within dataset")
plt.xticks([1, 2, 3, 4, 5, 6, 7, 8, 9, 10])
plt.yticks([0, 50, 100, 150])
#plt.grid(axis='y')
ax.spines['top'].set_visible(False)
ax.spines['right'].set_visible(False)
ax.spines['bottom'].set_visible(False)
ax.spines['left'].set_visible(False)
ax.set_xticks([2, 3, 4, 5, 6, 7, 8, 9, 10])
ax.set_yticks([0, 100])
plt.legend(frameon=False)
plt.show()
```


Benign
Malignent
Frequency within dataset
Clump thickness