FinalProject

August 28, 2022

1 Final Project

1.1 Due: April 25, 2022 @ 11:59pm

Name: Nikita Anistratov

Section: 001

Date: 04/12/2022

1.1.1 Import all proper packages:

- Here we import pandas, numpy, and pyplot functions
- plt.rcParams are just peramaters for our figuires

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (12, 7)
plt.rcParams['figure.dpi'] = 100
```

1.1.2 Import our data

- Now we import our csv file and do some housekeeping
- We change here all binary variables to type catagory using .astype("catagory")
- This allows to make nice clean descriptions of our binary varibles, and now we can use them as catagorical varibles
- We save this all as a dataframe

[2]:
```
heart_data = pd.read_csv('data/heart_failure_clinical_records_dataset.csv')

heart_data.sex = heart_data.sex.astype("category")
heart_data.sex.cat.categories = ["Female", "Male" ]

heart_data.anaemia = heart_data.anaemia.astype("category")
heart_data.anaemia.cat.categories = ["No Anaemia", "Anaemia"]

heart_data.diabetes = heart_data.diabetes.astype("category")
heart_data.diabetes.cat.categories = ["Non-diabetic", "Diabetic"]
```

```
heart_data.high_blood_pressure = heart_data.high_blood_pressure.
 ↪astype("category")
heart_data.high_blood_pressure.cat.categories = ["No high blood pressure",␣
 ↪"High blood pressure"]

heart_data.smoking = heart_data.smoking.astype("category")
heart_data.smoking.cat.categories = ["Non-smoker", "Smoker"]

heart_data.DEATH_EVENT = heart_data.DEATH_EVENT.astype("category")
heart_data.DEATH_EVENT.cat.categories = ["Not dead", "Dead"]

heart_data
```

[2]:

	age	anaemia	creatinine_phosphokinase	diabetes	\
0	75.0	No Anaemia	582	Non-diabetic	
1	55.0	No Anaemia	7861	Non-diabetic	
2	65.0	No Anaemia	146	Non-diabetic	
3	50.0	Anaemia	111	Non-diabetic	
4	65.0	Anaemia	160	Diabetic	
..	…	…	…	…	
294	62.0	No Anaemia	61	Diabetic	
295	55.0	No Anaemia	1820	Non-diabetic	
296	45.0	No Anaemia	2060	Diabetic	
297	45.0	No Anaemia	2413	Non-diabetic	
298	50.0	No Anaemia	196	Non-diabetic	

	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	\
0	20	High blood pressure	265000.00	1.9	
1	38	No high blood pressure	263358.03	1.1	
2	20	No high blood pressure	162000.00	1.3	
3	20	No high blood pressure	210000.00	1.9	
4	20	No high blood pressure	327000.00	2.7	
..	…	…	…	…	
294	38	High blood pressure	155000.00	1.1	
295	38	No high blood pressure	270000.00	1.2	
296	60	No high blood pressure	742000.00	0.8	
297	38	No high blood pressure	140000.00	1.4	
298	45	No high blood pressure	395000.00	1.6	

	serum_sodium	sex	smoking	time	DEATH_EVENT
0	130	Male	Non-smoker	4	Dead
1	136	Male	Non-smoker	6	Dead
2	129	Male	Smoker	7	Dead
3	137	Male	Non-smoker	7	Dead
4	116	Female	Non-smoker	8	Dead
..	…	…	…	…	…

```
294           143    Male      Smoker   270    Not dead
295           139  Female  Non-smoker   271    Not dead
296           138  Female  Non-smoker   278    Not dead
297           140    Male      Smoker   280    Not dead
298           136    Male      Smoker   285    Not dead

[299 rows x 13 columns]
```

1.1.3 A breif description of our data provided from the url below

The data we are exploring contains the medical records of 299 patients who had heart failure, collected during their follow-up period, where each patient profile has 13 clinical features.

Source:

Provide the names, email addresses, institutions, and other contact information of the donors and creators of the data set.The original dataset version was collected by Tanvir Ahmad, Assia Munir, Sajjad Haider Bhatti, Muhammad Aftab, and Muhammad Ali Raza (Government College University, Faisalabad, Pakistan) and made available by them on FigShare under the Attribution 4.0 International (CC BY 4.0: freedom to share and adapt the material) copyright in July 2017.

The current version of the dataset was elaborated by Davide Chicco (Krembil Research Institute, Toronto, Canada) and donated to the University of California Irvine Machine Learning Repository under the same Attribution 4.0 International (CC BY 4.0) copyright in January 2020. Davide Chicco can be reached at <davidechicco '@' davidechicco.it>

URL: https://archive.ics.uci.edu/ml/datasets/Heart+failure+clinical+records#

1.1.4 Now we give our own description of all the varibles using the information they have provided for ease of access and legability

1. Give description of your data set that includes a description of the variables (see Week 4 Review Notebook to see an example).

- The variables in this data frame are defined as:
- We have provided this in markdown text table

Variable	Description	Measurement	Range
Age	Age of the patient	Years	[40,..., 95]
Anaemia	Decrease of red blood cells or hemoglobin	Catgorical	0: No Anaemia, 1: Anaemia
Creatinine phosphokinase(CPK)	Level of the CPK enzyme in the blood	mcg/L	[23,..., 7861]
Diabetes	If the patient has diabetes	Catgorical	0: Non-diabetic, 1: Diabetic

Variable	Description	Measurement	Range
Ejection fraction	Percentage of blood leaving	Percentage	[14,…, 80]
High blood pressure	If a patient has hypertension	Catgorical	0: No hypertension, 1: Hypertension
Platelets	Platelets in the blood	kiloplatelets/mL	[25.01,…, 850.00]
Serum creatinine	Level of creatinine in the blood	mg/dL	[0.50,…, 9.40]
Serum sodium	Level of sodium in the blood	mEq/L	[114,…, 148]
Sex	Woman or Man	Catgorical	0: Woman, 1: Man
Smoking	If the patient smokes	Catgorical	0: Non-smoker, 1: Smoker
Time	Follow-up period	Days	[4,…,285]
Death event	If the patient died during the follow-up period	Catgorical	0: Not dead, 1: Dead

2. Load your data set and save it to a R tibble or pandas DataFrame object. Display the first 5 observations (rows).

1.1.5 We have already saved our imported data as a dataframe but now we use the head() function to dipsplay the first 5 rows

[3]:
```
heart_data.head()
```

[3]:
```
    age     anaemia  creatinine_phosphokinase      diabetes  \
0  75.0  No Anaemia                       582  Non-diabetic
1  55.0  No Anaemia                      7861  Non-diabetic
2  65.0  No Anaemia                       146  Non-diabetic
3  50.0     Anaemia                       111  Non-diabetic
4  65.0     Anaemia                       160      Diabetic

   ejection_fraction     high_blood_pressure  platelets  serum_creatinine  \
0                 20     High blood pressure  265000.00               1.9
1                 38  No high blood pressure  263358.03               1.1
2                 20  No high blood pressure  162000.00               1.3
3                 20  No high blood pressure  210000.00               1.9
4                 20  No high blood pressure  327000.00               2.7
```

	serum_sodium	sex	smoking	time	DEATH_EVENT
0	130	Male	Non-smoker	4	Dead
1	136	Male	Non-smoker	6	Dead
2	129	Male	Smoker	7	Dead
3	137	Male	Non-smoker	7	Dead
4	116	Female	Non-smoker	8	Dead

3. Identify the numerical and categorical features (columns) that you are going to investigate for this assignment. Write a brief paragraph describing what interests you about the features you chose.

1.1.6 Here we simply describe our chosen numarical and catagorical features that we will investigate

The numarical features we will be exploring are: age, serum_creatinine, serum_sodium. For catagorical features we will explore diabetes, high_blood_pressure, sex, smoking, DEATH_EVENT. These features were chosen sepciafically, because they are enough to create a good understanding of the health statistics, specifically regarding diabetics. serum_ceratinine and serum_serum relate to diabeties. High blood pressure, smoking, age and sex all give us another demographic of high blood pressure and will all relate to and effectthe vairiable high blood pressure. Age and sex will give us an undertanding of the variables diabeties and high blood pressure, giving us a few categorical.

4. Create some numerical summaries and display the results in a table.

- Here we are taking the mean, median and Standard deviation of varibles Age, Serum creatinine, and Serum sodium and displaying them in a table
- We use the numpy package as np, np.mean, np.median and np.std
- Lastly we display all this inofrmation in a dataframe and label the columns accordingly

[4]:
```
mean_age = np.mean(heart_data.age)

mean_serum_creatinine = np.mean(heart_data.serum_creatinine)

mean_serum_sodium = np.mean(heart_data.serum_sodium)

median_age = np.median(heart_data.age)

median_serum_creatinine = np.median(heart_data.serum_creatinine)

median_serum_sodium = np.median(heart_data.serum_sodium)

std_age = np.std(heart_data.age)

std_serum_creatinine = np.std(heart_data.serum_creatinine)

std_serum_sodium = np.std(heart_data.serum_sodium)
```

```
table = [ ['Age', mean_age, median_age, std_age],
         [ "Serum creatinine",mean_serum_creatinine, median_serum_creatinine,␣
 ↪std_serum_creatinine ], ["Serum sodium",mean_serum_sodium,␣
 ↪median_serum_sodium, std_serum_sodium]]
pd.DataFrame( table, columns=['Variable', 'Mean', 'Median', 'Standard␣
 ↪Deviation'])
```

[4]:

	Variable	Mean	Median	Standard Deviation
0	Age	60.833893	60.0	11.874901
1	Serum creatinine	1.393880	1.1	1.032779
2	Serum sodium	136.625418	137.0	4.405092

- Now we will use the pandas describe() function to get som more summeries on the three chosen varibles
- Age, Serum creatinine, and Serum sodium
- Also displayed as dataframe

[5]:
```
heart_data.describe()
```

[5]:

	age	creatinine_phosphokinase	ejection_fraction	platelets	\
count	299.000000	299.000000	299.000000	299.000000	
mean	60.833893	581.839465	38.083612	263358.029264	
std	11.894809	970.287881	11.834841	97804.236869	
min	40.000000	23.000000	14.000000	25100.000000	
25%	51.000000	116.500000	30.000000	212500.000000	
50%	60.000000	250.000000	38.000000	262000.000000	
75%	70.000000	582.000000	45.000000	303500.000000	
max	95.000000	7861.000000	80.000000	850000.000000	

	serum_creatinine	serum_sodium	time
count	299.00000	299.000000	299.000000
mean	1.39388	136.625418	130.260870
std	1.03451	4.412477	77.614208
min	0.50000	113.000000	4.000000
25%	0.90000	134.000000	73.000000
50%	1.10000	137.000000	115.000000
75%	1.40000	140.000000	203.000000
max	9.40000	148.000000	285.000000

5. Include measures of center (mean, median) and spread (standard deviation) across a single grouping variable.

- Here we are finding the mean over our chosen single grouping variable “Sex” using groupby()
- again using numpy .mean()
- And displaying the information in a dataframe

[6]:
```
heart_data.groupby("sex").mean()
```

[6]:
```
              age  creatinine_phosphokinase  ejection_fraction      platelets  \
sex
Female  59.777781                476.780952          40.466667  279964.021619
Male    61.405500                638.701031          36.793814  254370.249897

        serum_creatinine  serum_sodium        time
sex
Female          1.384095    136.790476  131.904762
Male            1.399175    136.536082  129.371134
```

- Here we are finding the Median over our chosen single grouping variable "Sex" using groupby()
- Again using numpy .median()
- And displaying the information in a dataframe

[7]:
```
heart_data.groupby("sex").median()
```

[7]:
```
         age  creatinine_phosphokinase  ejection_fraction  platelets  \
sex
Female  60.0                     250.0               38.0  263358.03
Male    60.0                     249.0               35.0  253000.00

        serum_creatinine  serum_sodium   time
sex
Female               1.0         137.0  109.0
Male                 1.1         137.0  117.5
```

- Here we are finding the standard deviation over our chosen single grouping variable "Sex"using groupby()
- Again using numpy .std()
- And displaying the information in a dataframe

[8]:
```
heart_data.groupby("sex").std()
```

[8]:
```
              age  creatinine_phosphokinase  ejection_fraction      platelets  \
sex
Female  11.240919                611.364190          12.728728  102108.749558
Male    12.224415               1114.894007          11.144308   94447.363939

        serum_creatinine  serum_sodium       time
sex
Female          1.118633      4.904267  77.625893
Male            0.988976      4.132675  77.794178
```

6. Create new a categorical column based on one of your numerical columns. For example, you could use categories like "High", "Medium", or "Low". Describe the meaning of the values in the new column.

We create a new catagorical column using pandas Catagorical() and numpy where()

- We have split our age into two catgories
- Less than 60 wil be middle aged
- Greater than 60 will be considered elderly
- We have added this catagory as Age_Range to the end of our original data frame

[9]:
```
mask = (heart_data['age'] < 60)
heart_data['Age_Range']  = pd.Categorical(np.where(mask, 'Middle Aged',␣
 ↪'Elderly'))
heart_data
```

[9]:

	age	anaemia	creatinine_phosphokinase	diabetes	\
0	75.0	No Anaemia	582	Non-diabetic	
1	55.0	No Anaemia	7861	Non-diabetic	
2	65.0	No Anaemia	146	Non-diabetic	
3	50.0	Anaemia	111	Non-diabetic	
4	65.0	Anaemia	160	Diabetic	
..	…	…	…	…	
294	62.0	No Anaemia	61	Diabetic	
295	55.0	No Anaemia	1820	Non-diabetic	
296	45.0	No Anaemia	2060	Diabetic	
297	45.0	No Anaemia	2413	Non-diabetic	
298	50.0	No Anaemia	196	Non-diabetic	

	ejection_fraction	high_blood_pressure	platelets	serum_creatinine	\
0	20	High blood pressure	265000.00	1.9	
1	38	No high blood pressure	263358.03	1.1	
2	20	No high blood pressure	162000.00	1.3	
3	20	No high blood pressure	210000.00	1.9	
4	20	No high blood pressure	327000.00	2.7	
..	…	…	…	…	
294	38	High blood pressure	155000.00	1.1	
295	38	No high blood pressure	270000.00	1.2	
296	60	No high blood pressure	742000.00	0.8	
297	38	No high blood pressure	140000.00	1.4	
298	45	No high blood pressure	395000.00	1.6	

	serum_sodium	sex	smoking	time	DEATH_EVENT	Age_Range
0	130	Male	Non-smoker	4	Dead	Elderly
1	136	Male	Non-smoker	6	Dead	Middle Aged
2	129	Male	Smoker	7	Dead	Elderly
3	137	Male	Non-smoker	7	Dead	Middle Aged
4	116	Female	Non-smoker	8	Dead	Elderly
..	…	…	…	…	…	…
294	143	Male	Smoker	270	Not dead	Elderly
295	139	Female	Non-smoker	271	Not dead	Middle Aged
296	138	Female	Non-smoker	278	Not dead	Middle Aged
297	140	Male	Smoker	280	Not dead	Middle Aged

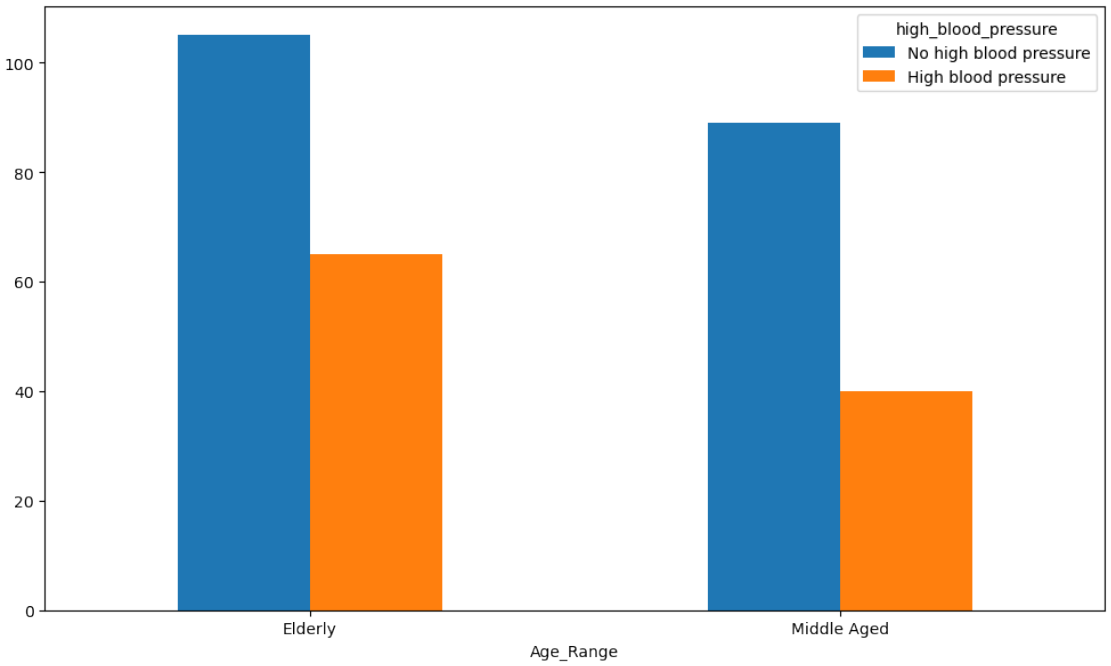
```
298          136    Male      Smoker   285    Not dead  Middle Aged

[299 rows x 14 columns]
```

7. Create a bar plot for two of the categorical variables. For one one bar plot use the column you created in part 6.

1.1.7 We Create a bar plot for Age Range the new catagorical column created in part 6 and high_blood_pressure:

- We will first make a two way table with high_blood_pressure and our new catagorical column Age_Range using pandas crosstab() to be able to plot these side by side
- Then we use plt.bar with the two way table to plot the bar plot

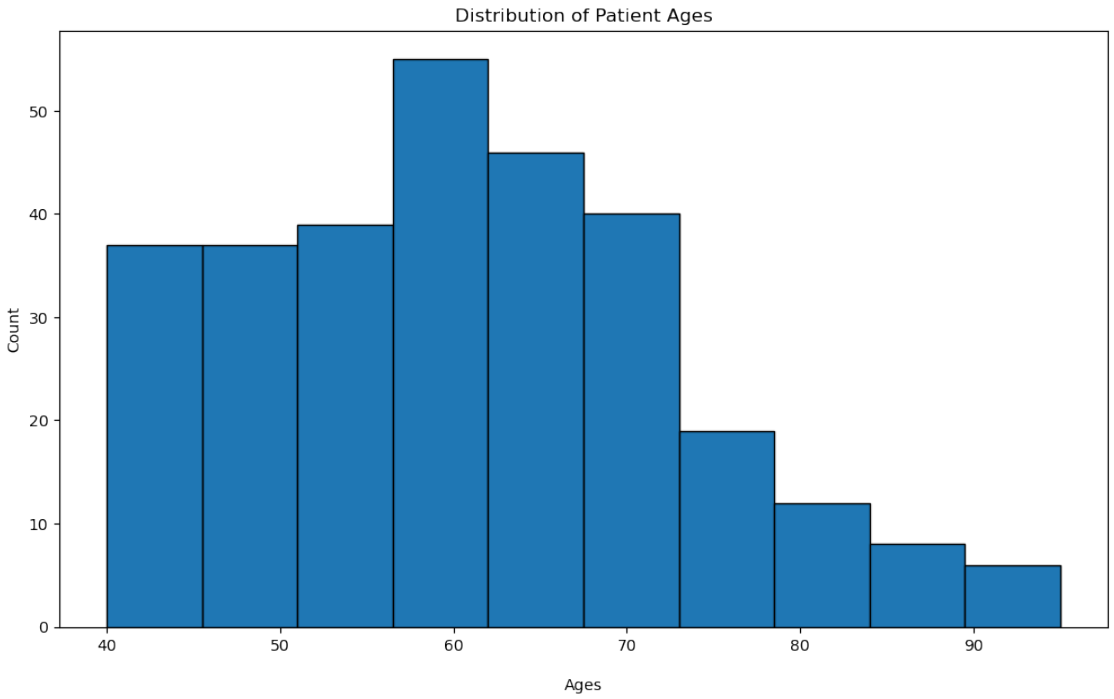
[10]:
```
two_way_tbl = pd.crosstab(heart_data.Age_Range, heart_data.high_blood_pressure)
two_way_tbl

two_way_tbl.plot.bar(rot = 0);
```



8. Create a histogram for one of the numerical variables. ### We create a histogram of the numarical varible age * Here we use the hist() function on age * And we label our axis using set_title, set_xlabel...

[11]:
```
ax = heart_data.age.hist(edgecolor = "black")
ax.grid(False)
ax.set_title("Distribution of Patient Ages")
```

```
ax.set_xlabel("\nAges")
ax.set_ylabel("Count");
```



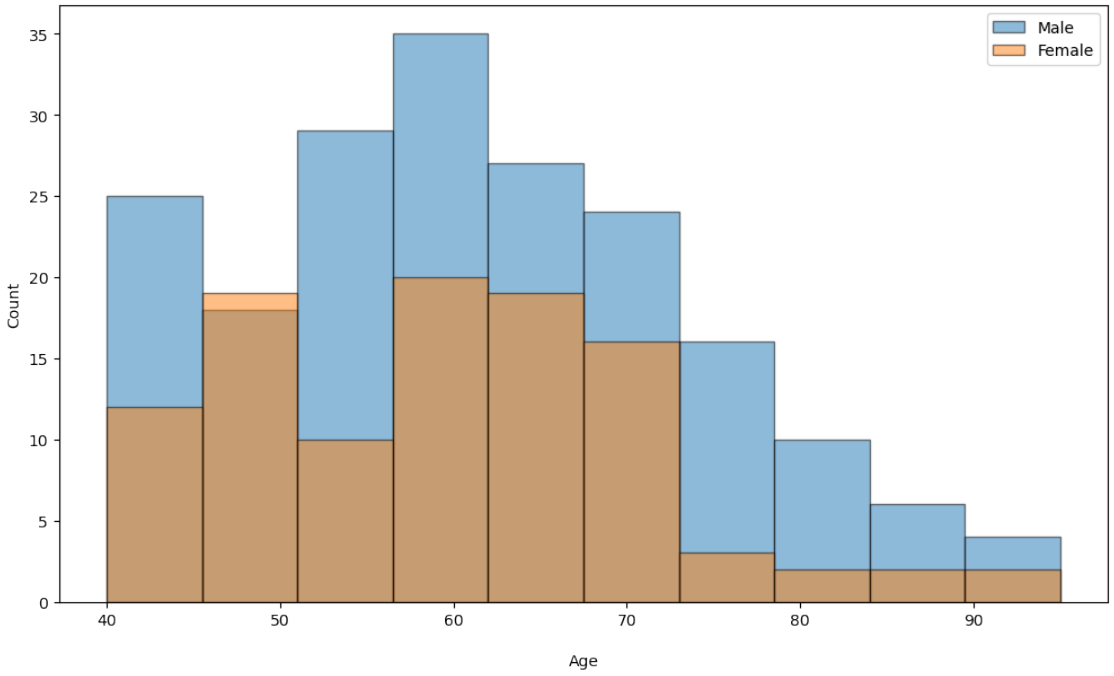
9. Create a histogram (ovelayed) for one of the numerical variables across one of the categorical variables (that is, create graphs that can compare the distributions across the groups). For at least one of the histogram plots across groups, make sure that the graphs are overlayed on the same plot. Add appropriate labels and titles.

1.1.8 We create an overlayed histogram of sex and age of patients

- we use the loc() fucntion to locate male and save to a varible and then we do the same with female
- We concatonate both of these with age so we can save them in individual varibles
- we then use these created varibles to plot male and female with age overlayed
- we use th hist() function
- as well as set_xlabel and set_ylabel for axis

[12]:
```
male = heart_data.loc[heart_data.sex == 'Male'].age
female = heart_data.loc[heart_data.sex == 'Female'].age
num_bins = 10
ax = male.hist(bins = num_bins, alpha = 0.5, label = 'Male', edgecolor =␣
 ↪'black')
ax = female.hist(bins = num_bins, alpha = 0.5, label = 'Female', edgecolor =␣
 ↪'black')
ax.grid(False)
```

```
ax.set_xlabel("\nAge")
ax.set_ylabel("Count")
ax.legend();
```



10. Create a scatter plot relating two numeric variables that includes the graph of the least squares regression line. Be sure to find the value of the correlation coefficient.

1.1.9 We create a scatter plot relating Age and sodium levels of patients

- We first create functions taken from our lessons to use to find standard units, and correlation coefficients as well as correlation coefficients using standard units

[13]:
```
def standard_units(col):
    """
    Purpose
    -------
    Standardize the values in column

    Parameters
    ----------
    col: A 1 dimensional numpy array or a column from a dataframe

    """

    svs = (col - np.mean(col))/np.std(col)
    return svs
```

```
def corr_coef(x, y):
    """
    Purpose
    -------
    Compute Pearson's Correlation Coefficient

    Parameters
    ----------
    x, y: A 1 dimensional numpy array or a column from a dataframe
    """
    xbar = np.mean(x)
    ybar = np.mean(y)
    numerator_sum = np.sum((x - xbar) * (y - ybar))
    denominator_sumx = np.sum((x - xbar) ** 2)
    denominator_sumy = np.sum((y - ybar) ** 2)
    return numerator_sum/np.sqrt(denominator_sumx*denominator_sumy)

def su_corr_coef(x, y):
    """
    Purpose
    -------
    Compute Pearson's Correlation Coefficient

    Parameters
    ----------
    x, y: A 1 dimensional numpy array or a column from a dataframe in standard␣
 ↪units
    """
    su_x = standard_units(x)
    su_y = standard_units(y)
    return np.mean(su_x * su_y)
```

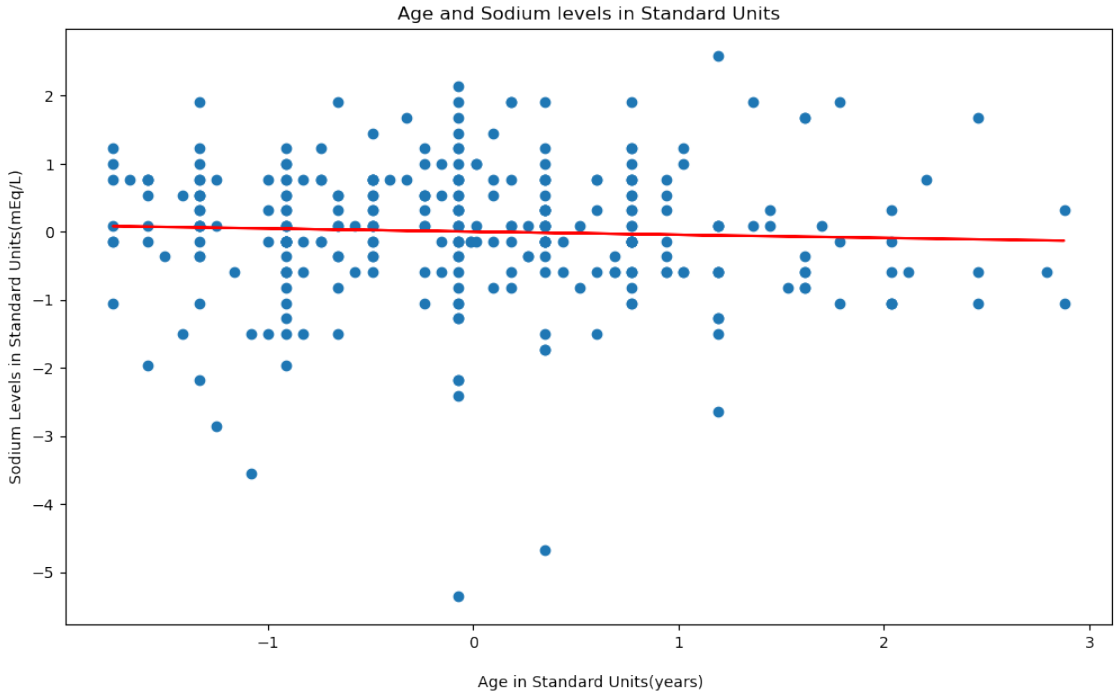
- Now we use this function corr_coef to find and display the correlation coefficient of age and serum_sodium (sodium levels)

[14]:
```
cor = corr_coef(heart_data.age,heart_data.serum_sodium)
print("The Correlation Coefficient between Age and Sodium: ", cor)
```

```
The Correlation Coefficient between Age and Sodium:  -0.045965840839560033
```

- Here we find standard units of our varibles
- Age is our x axis
- serum_sodium is our y axis
- We then use these standard units to find the standard unit correlation coefficient
- Lastly we plot using plt.scattar
- And title our axis as well as plot our regression line using plot(), using a forumla for regression line with standard units and standard unit correlation coefficient

[15]:
```
x_su = standard_units(heart_data.age)
y_su = standard_units(heart_data.serum_sodium)

r_su = su_corr_coef(heart_data.age, heart_data.serum_sodium)
plt.scatter(x = x_su, y = y_su)
plt.plot(x_su, r_su*x_su, c = 'r')
plt.title("Age and Sodium levels in Standard Units")
plt.xlabel("\n Age in Standard Units(years)")
plt.ylabel("Sodium Levels in Standard Units(mEq/L)");
```



- Now we use sklearn to find a similar scatter plot using different methods
- We import the linear regression model from sklearn.linear_model
- We thn use our model mp_model and the fit() fucntion to include our chosen varibles age and sodium

[16]:
```
from sklearn.linear_model import LinearRegression
mp_model = LinearRegression()
mp_model.fit(heart_data[['age']], heart_data['serum_sodium'])
```

[16]:
```
LinearRegression()
```

11. Use scikit learn to find the slope and y−intercept of the least squares regression line. Be sure to plot the line on the scatter plot.

1.1.10 Here we use Sklearn.linear_model to find the slope and y intercept of the regression line

- Just like the previous example but now using the sklearn linear regression model
- We call upon this mp_model varible we created to find the intercept and correlation coefficient
- we use .intercept and .coef to easily pull these values
- We print this below

[17]:
```
y_int = mp_model.intercept_
slope = mp_model.coef_[0]
print("The y-intercept is", y_int)
print("The slope is", slope)
```

```
The y-intercept is 137.66272151764863
The slope is -0.017051406817695344
```

- Now using this method we display the same scatter plot and regression line
- We see this method yeilds the same answer
- Before we used manual functions now we use built in sklearn functions and this proves to us we have done 10 correctly!

[18]:
```
y_int = mp_model.intercept_
slope = mp_model.coef_[0]
plt.scatter(x = heart_data.age, y = heart_data.serum_sodium)
plt.plot(heart_data.age, slope * heart_data.age + y_int, c = 'r');
```