Data Visualizations using Matplotlib

July 7, 2024

Abstract

Matplotlib is a popular plotting library for Python. It is used for creating static, animated, and interactive visualizations in Python.

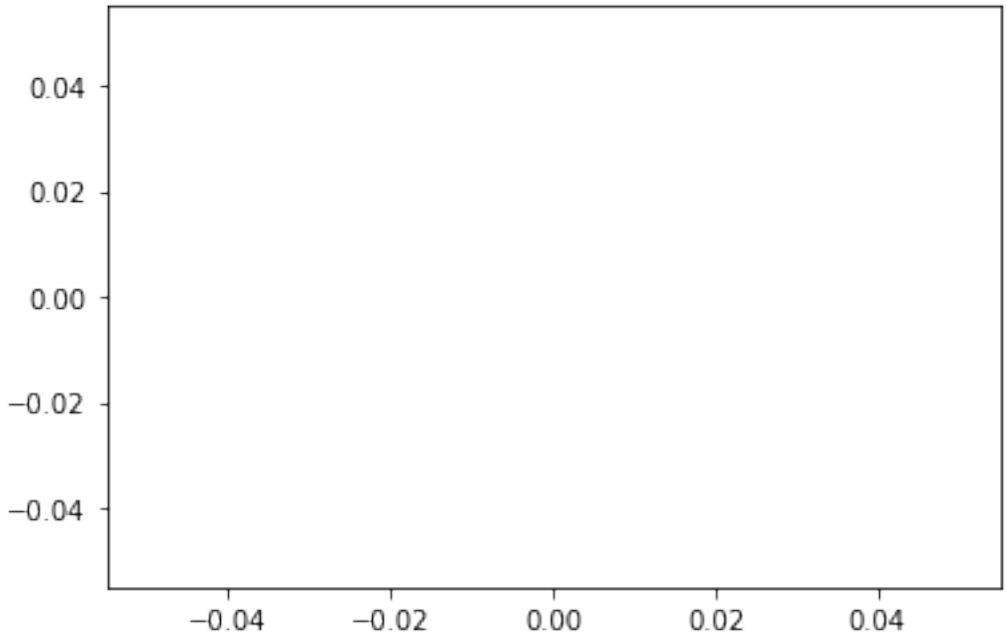
- Credits -
  - zero-to-mastery-ml - https://github.com/mrdbourke/zero-to-mastery-ml/tree/master/data
  - https://www.udemy.com/course/complete-machine-learning-and-data-science-zero-to-mastery

0.0.1 Introduction to Matplotlib

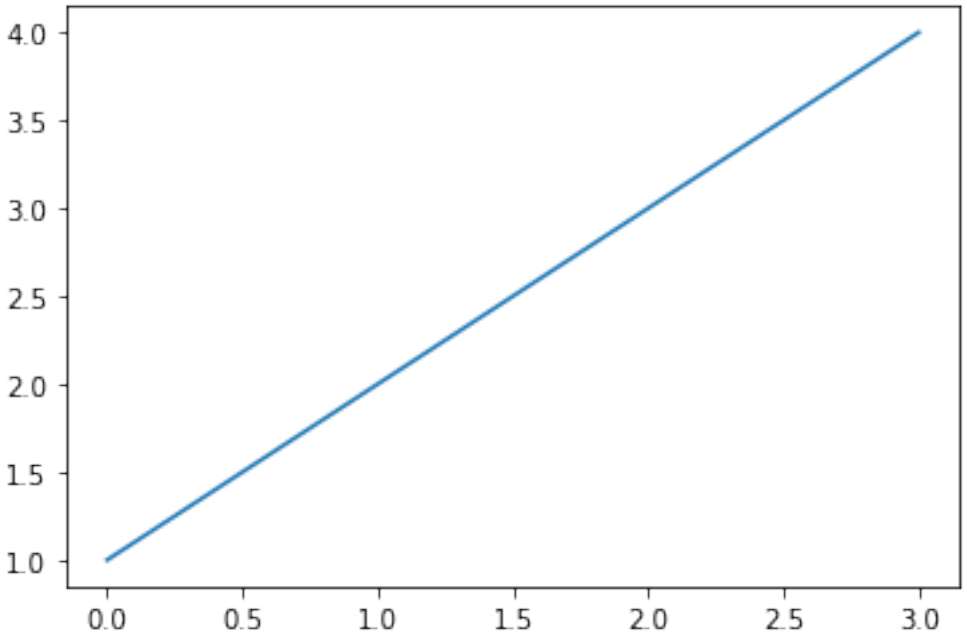
- https://matplotlib.org/stable/api/pyplot_summary.html

[1]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
```

[6]:
```
# Creating our first figure
plt.plot();
plt.show()
```

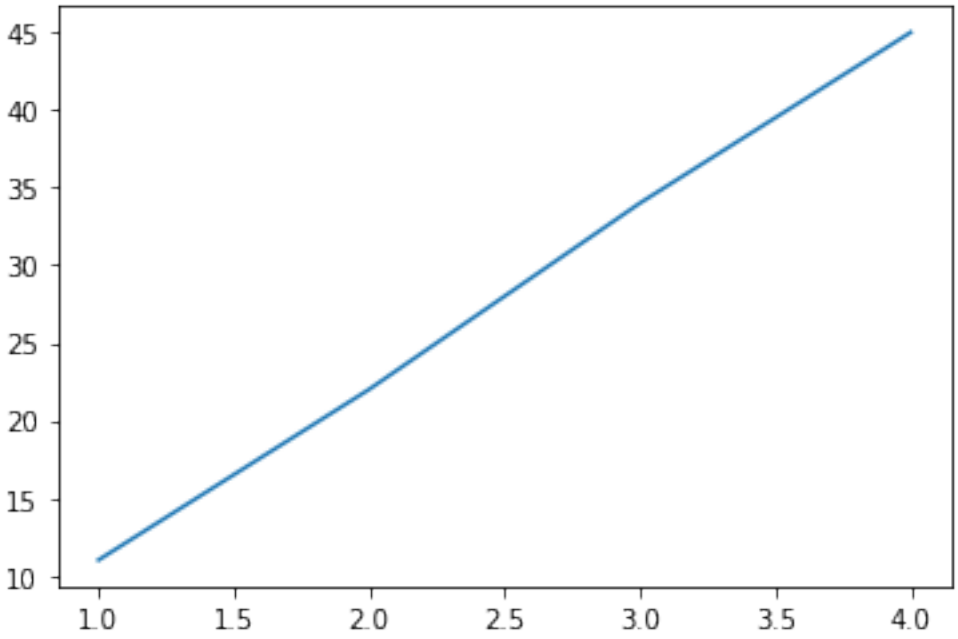


[8]:
```
plt.plot([1,2,3,4]);
```

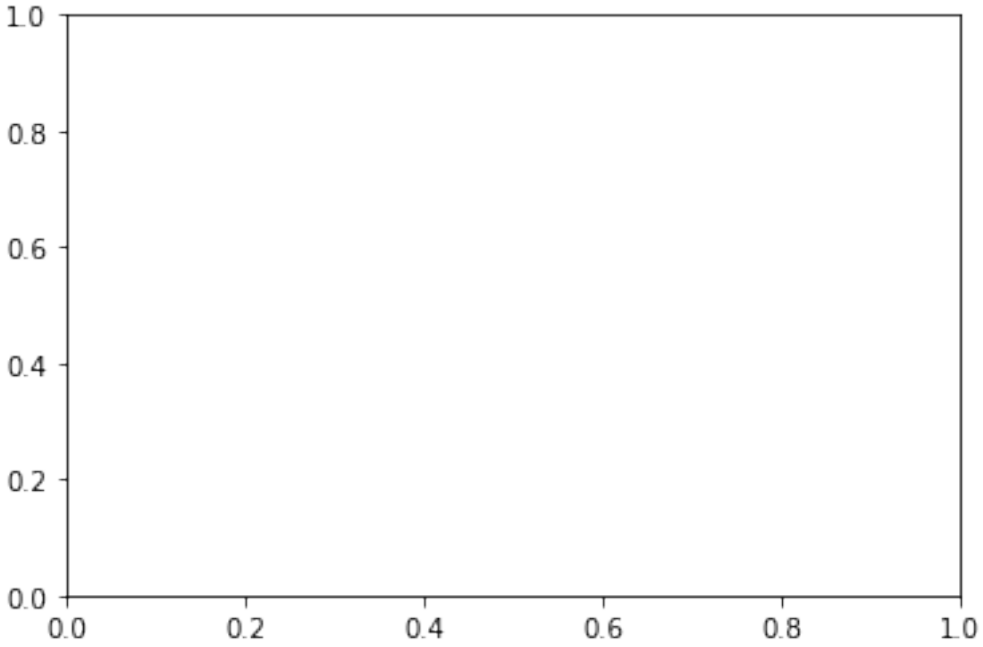


[9]:
```
x = [1,2,3,4]
y =[11,22,34,45]
```

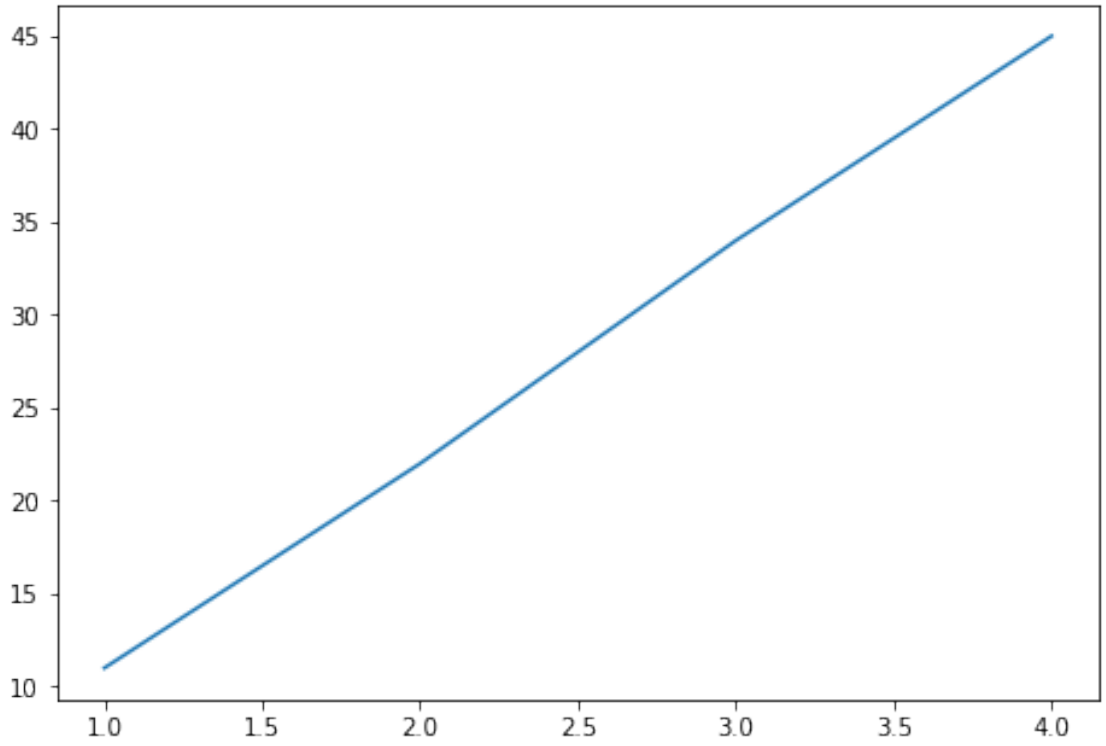
```
plt.plot(x,y);
```



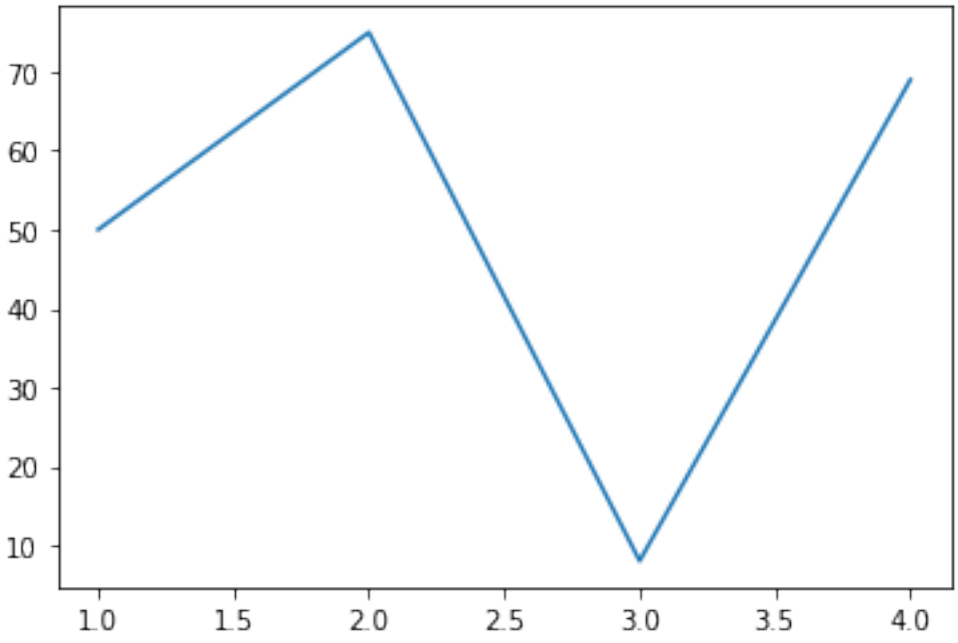
[10]:
```
# 1st method
fig  = plt.figure() #creates a figure
ax = fig.add_subplot() # add some axes
plt.show()
```



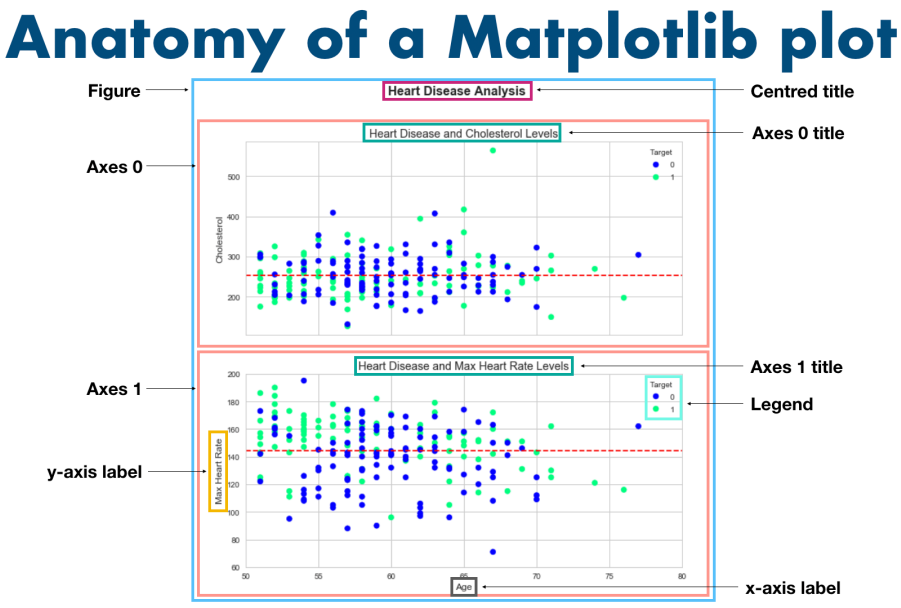
[14]:
```
#2nd method
fig = plt.figure()
ax = fig.add_axes([1,1,1,1])
ax.plot(x,y)
plt.show()
```



[18]:
```
#3rd method - Recommended https://matplotlib.org/stable/tutorials/lifecycle.
↪html#sphx-glr-tutorials-lifecycle-py

fig, ax = plt.subplots()
ax.plot(x,[50,75,8,69]);
```



0.0.2 Anatomy of a Matplotlib

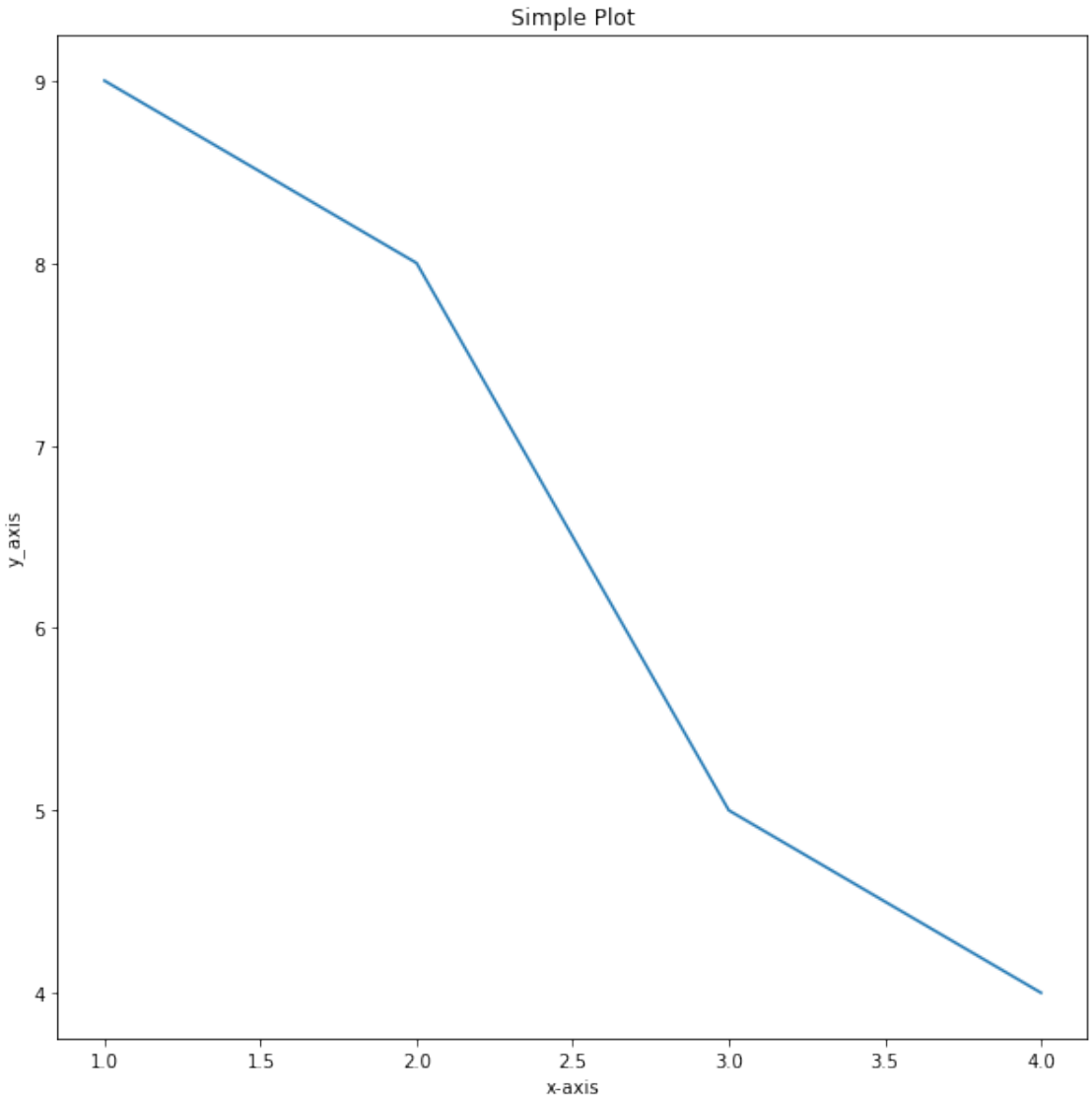


0.0.3 Matplotlib example workflow

Lets run this all in one cell

[24]:
```
# 0. import matplotlib and get it ready for plotting in jupyter
```

```
%matplotlib inline
import matplotlib.pyplot as plt

# 1. Prepare data

x = [1,2,3,4]
y=[9,8,5,4]

# 2. Setup plot
fig, ax = plt.subplots(figsize=(10,10)) #Width and # height

# 3 . Plot data
ax.plot(x,y)

# 4 . Customise the plot

ax.set(title="Simple Plot",
       xlabel="x-axis",
       ylabel="y_axis")
#5 .  Save the figure
fig.savefig("./Resources/Sample-plot.jpg")
```



0.0.4 Making figures with Numpy

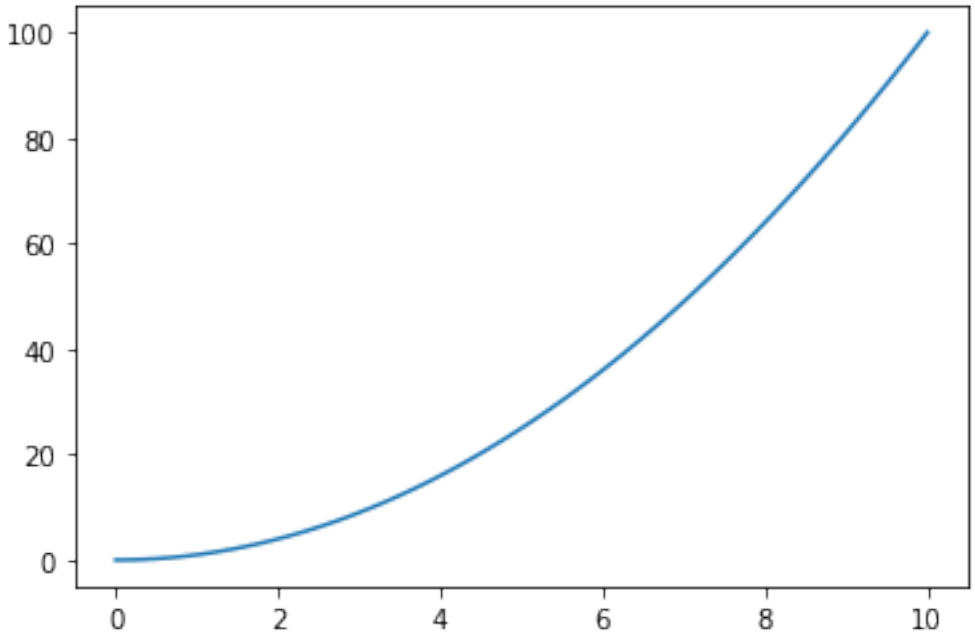
We will be creating: * Line plot * Scatter Plot * Bar plot * Histogram * Subplots

[26]:
```
import numpy as np
```

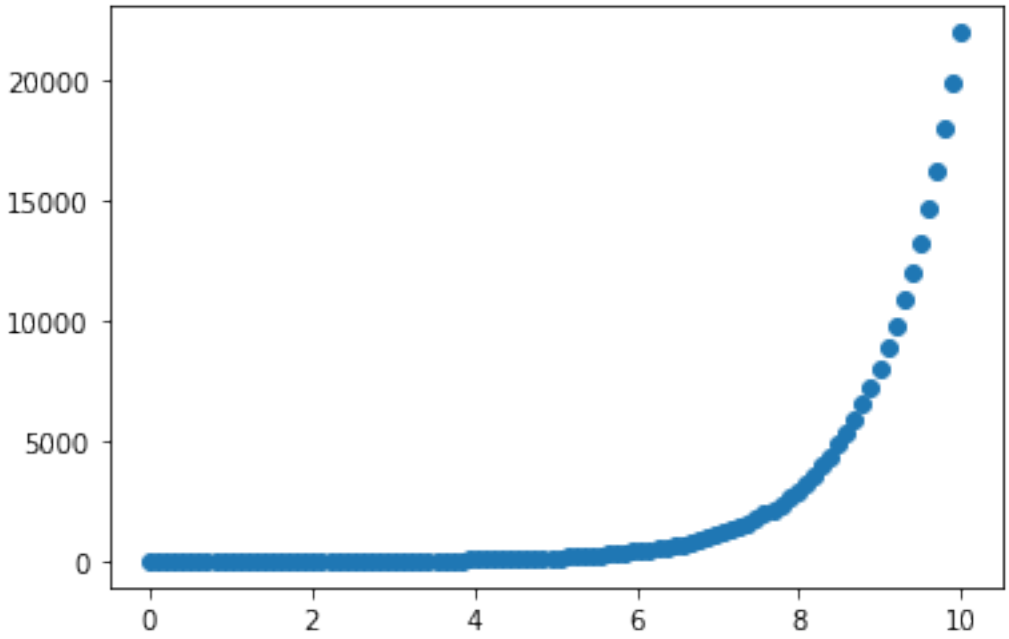
[31]:
```
# Create some data

x = np.linspace(0,10,100)
x[:10]
```

[31]:
```
array([0.        , 0.1010101 , 0.2020202 , 0.3030303 , 0.4040404 ,
       0.50505051, 0.60606061, 0.70707071, 0.80808081, 0.90909091])
```

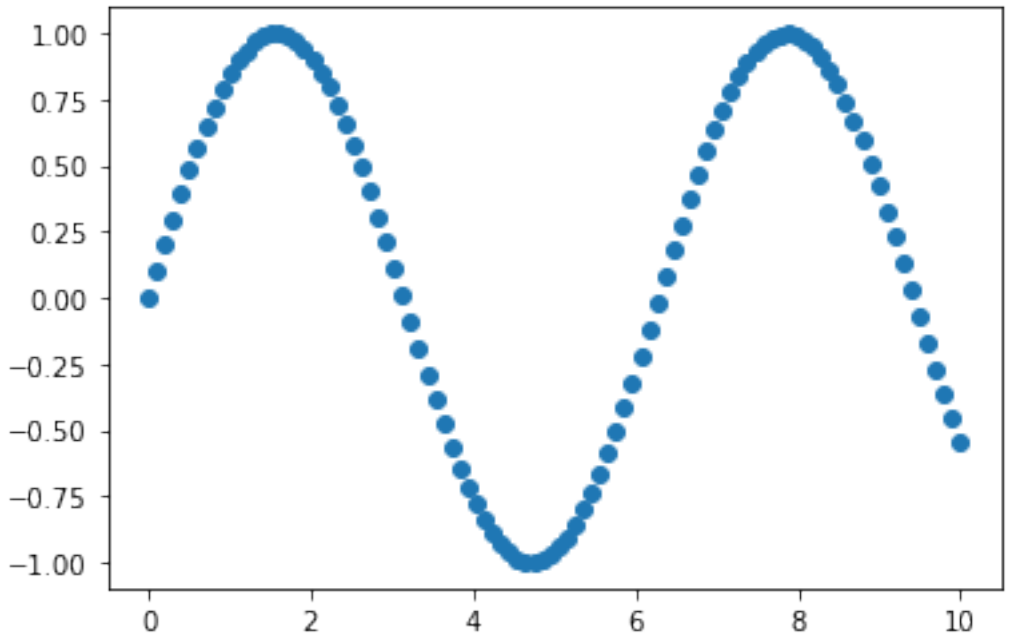
[35]:
```
# Plot the data and create a line plot

fig, ax = plt.subplots()
ax.plot(x,x**2);
```



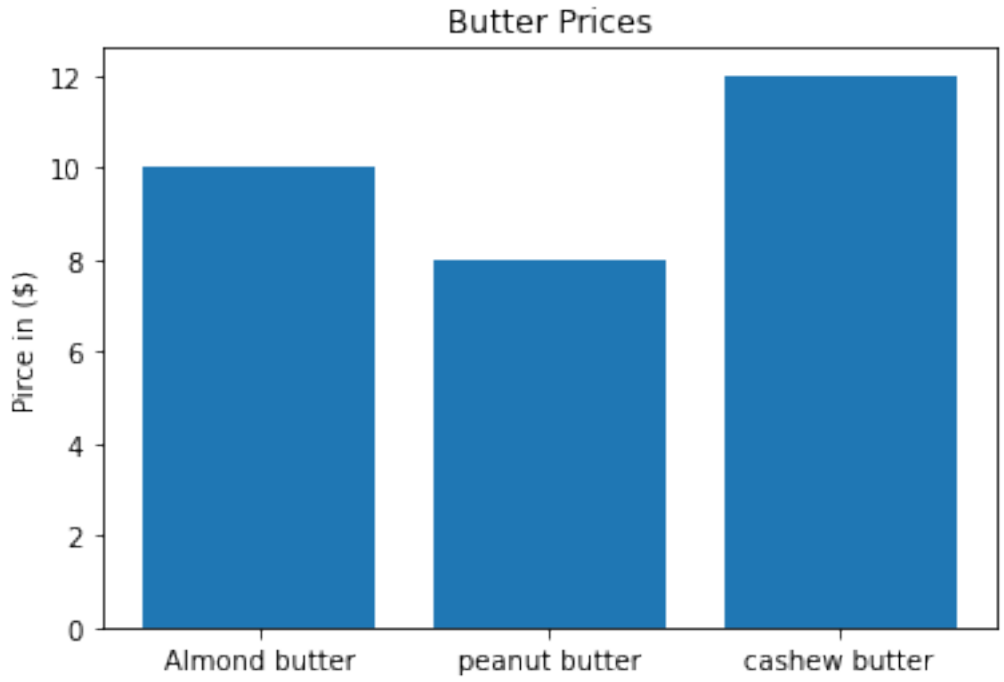
[38]:
```
# Use the same data and create a scatter plot

fig, ax = plt.subplots()
ax.scatter(x, np.exp(x));
```



[40]:
```
# Another scatter plot

fig, ax = plt.subplots()
ax.scatter(x, np.sin(x));
```

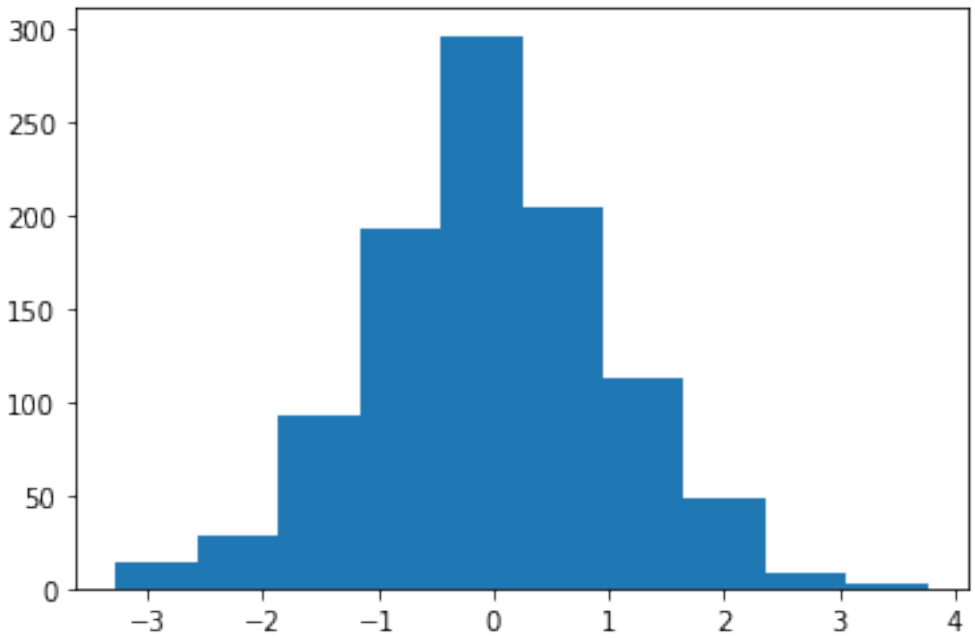


[49]:
```
# Make a plot from dictionary

nut_butter_prices = {"Almond butter":10,
                     "peanut butter":8,
                     "cashew butter":12}

fig, ax =plt.subplots()
ax.bar(nut_butter_prices.keys(),nut_butter_prices.values());
ax.set(title="Butter Prices",ylabel="Pirce in ($)");
```



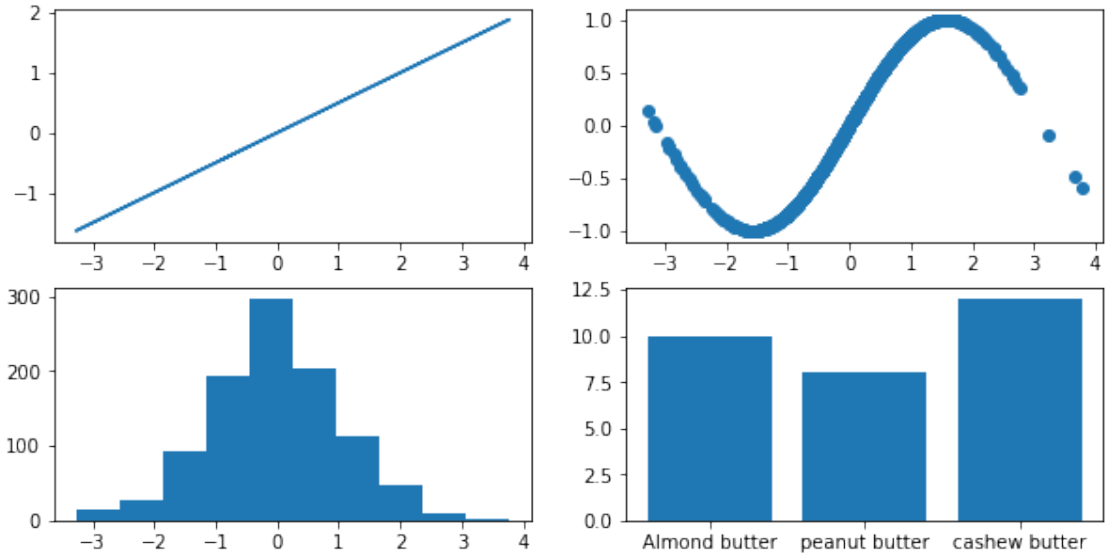
[59]:
```
fig, ax = plt.subplots()
ax.barh(list(nut_butter_prices.keys()),list(nut_butter_prices.values()));
ax.set(title="Butter Prices",xlabel="Pirce in ($)");
```

[65]:
```
# Make some data for histogram and plot it
x= np.random.randn(1000)
fig, ax = plt.subplots()
ax.hist(x);
```



0.0.5 Creating subplots

[78]:
```
# Option 1
fig, ((ax1,ax2),(ax3,ax4)) = plt.subplots(nrows=2, ncols=2, figsize=(10,5))

# Plot different axis

ax1.plot(x,x/2);
ax2.scatter(x,np.sin(x));
ax3.hist(x);
ax4.bar(nut_butter_prices.keys(),nut_butter_prices.values());
```



Plotting using Pandas Dataframe

[79]:
```
# Make a dataframe
car_sales=pd.read_csv("./Resources/car-sales.csv")
car_sales
```

[79]:

	Make	Colour	Odometer (KM)	Doors	Price
0	Toyota	White	150043	4	$4,000.00
1	Honda	Red	87899	4	$5,000.00
2	Toyota	Blue	32549	3	$7,000.00
3	BMW	Black	11179	5	$22,000.00
4	Nissan	White	213095	4	$3,500.00
5	Toyota	Green	99213	4	$4,500.00
6	Honda	Blue	45698	4	$7,500.00

```
7   Honda   Blue          54738      4   $7,000.00
8  Toyota  White          60000      4   $6,250.00
9  Nissan  White          31600      4   $9,700.00
```

[80]:
```
car_sales["Price"]=car_sales["Price"].str.replace('[\$\,\.]','').astype(int)
```

```
/local/pkg/python/root-python-3.7/lib/python3.7/site-
packages/ipykernel_launcher.py:1: FutureWarning: The default value of regex will
change from True to False in a future version.
  """Entry point for launching an IPython kernel.
```

[81]:
```
car_sales
```

[81]:
```
     Make Colour  Odometer (KM)  Doors    Price
0  Toyota  White         150043      4   400000
1   Honda    Red          87899      4   500000
2  Toyota   Blue          32549      3   700000
3     BMW  Black          11179      5  2200000
4  Nissan  White         213095      4   350000
5  Toyota  Green          99213      4   450000
6   Honda   Blue          45698      4   750000
7   Honda   Blue          54738      4   700000
8  Toyota  White          60000      4   625000
9  Nissan  White          31600      4   970000
```

[86]:
```
car_sales["Price"]=car_sales["Price"]/100
```

[87]:
```
car_sales
```

[87]:
```
     Make Colour  Odometer (KM)  Doors    Price
0  Toyota  White         150043      4   4000.0
1   Honda    Red          87899      4   5000.0
2  Toyota   Blue          32549      3   7000.0
3     BMW  Black          11179      5  22000.0
4  Nissan  White         213095      4   3500.0
5  Toyota  Green          99213      4   4500.0
6   Honda   Blue          45698      4   7500.0
7   Honda   Blue          54738      4   7000.0
8  Toyota  White          60000      4   6250.0
9  Nissan  White          31600      4   9700.0
```

[88]:
```
car_sales["Sales_date"]=pd.date_range("1/1/2020",periods=len(car_sales))
car_sales
```

[88]:
```
     Make Colour  Odometer (KM)  Doors    Price Sales_date
0  Toyota  White         150043      4   4000.0 2020-01-01
1   Honda    Red          87899      4   5000.0 2020-01-02
2  Toyota   Blue          32549      3   7000.0 2020-01-03
```

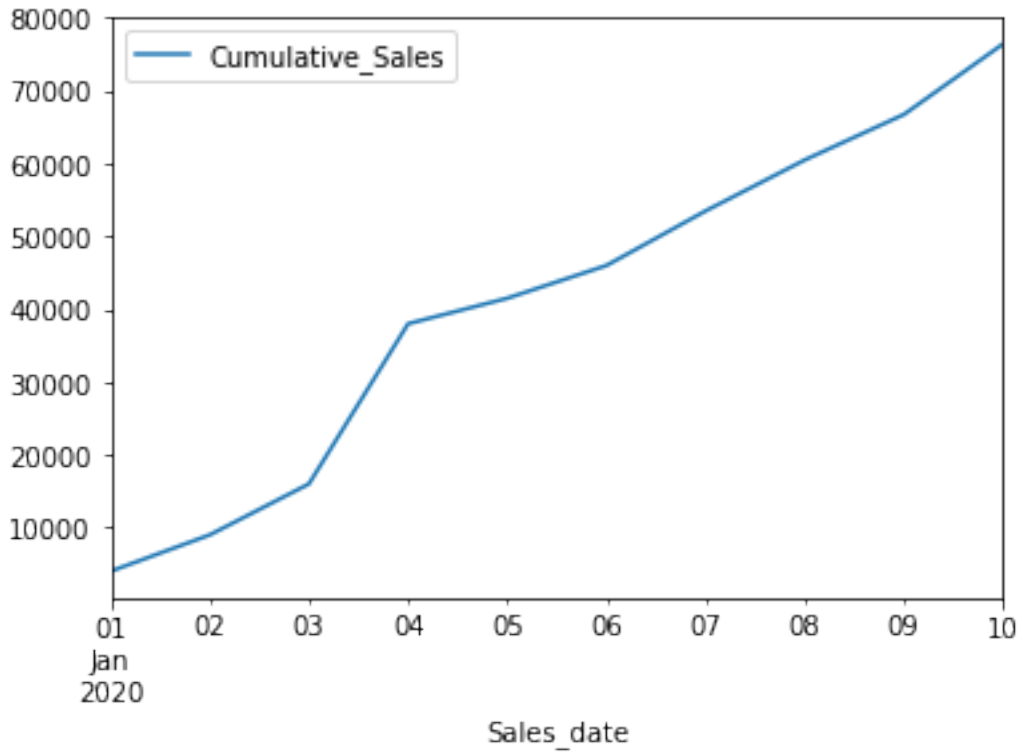
```
3     BMW  Black          11179      5  22000.0 2020-01-04
4  Nissan  White         213095      4   3500.0 2020-01-05
5  Toyota  Green          99213      4   4500.0 2020-01-06
6   Honda   Blue          45698      4   7500.0 2020-01-07
7   Honda   Blue          54738      4   7000.0 2020-01-08
8  Toyota  White          60000      4   6250.0 2020-01-09
9  Nissan  White          31600      4   9700.0 2020-01-10
```

[89]:
```
car_sales["Cumulative_Sales"]=car_sales["Price"].cumsum()
car_sales
```

[89]:

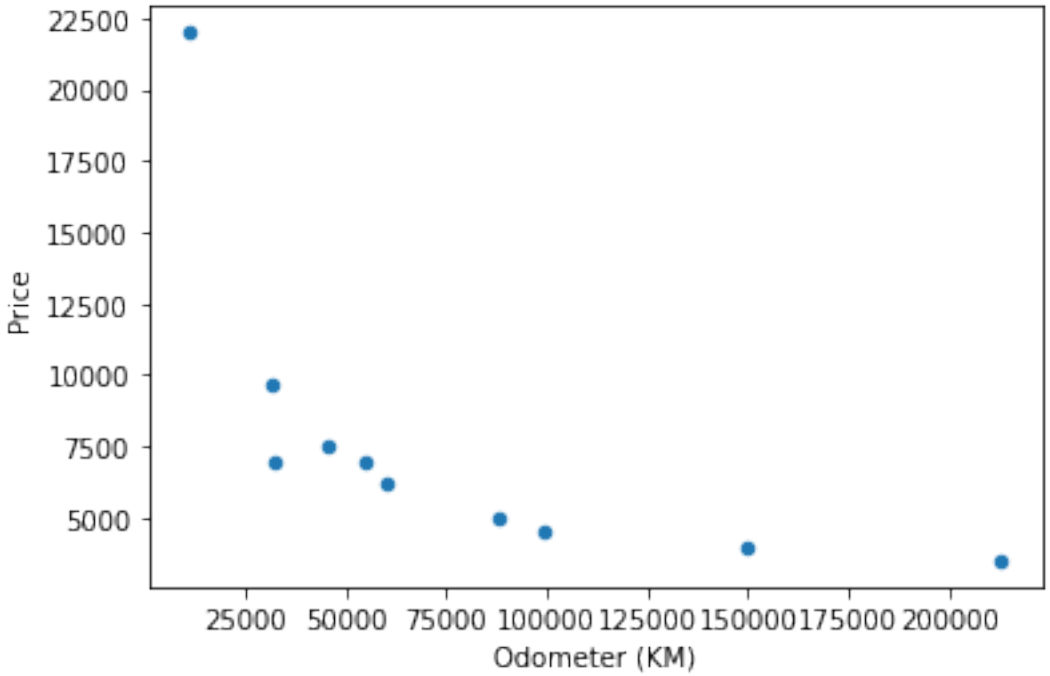
	Make	Colour	Odometer (KM)	Doors	Price	Sales_date	Cumulative_Sales
0	Toyota	White	150043	4	4000.0	2020-01-01	4000.0
1	Honda	Red	87899	4	5000.0	2020-01-02	9000.0
2	Toyota	Blue	32549	3	7000.0	2020-01-03	16000.0
3	BMW	Black	11179	5	22000.0	2020-01-04	38000.0
4	Nissan	White	213095	4	3500.0	2020-01-05	41500.0
5	Toyota	Green	99213	4	4500.0	2020-01-06	46000.0
6	Honda	Blue	45698	4	7500.0	2020-01-07	53500.0
7	Honda	Blue	54738	4	7000.0	2020-01-08	60500.0
8	Toyota	White	60000	4	6250.0	2020-01-09	66750.0
9	Nissan	White	31600	4	9700.0	2020-01-10	76450.0

[92]:
```
#Lets plot the total sales over time
car_sales.plot(x="Sales_date",y="Cumulative_Sales");
```

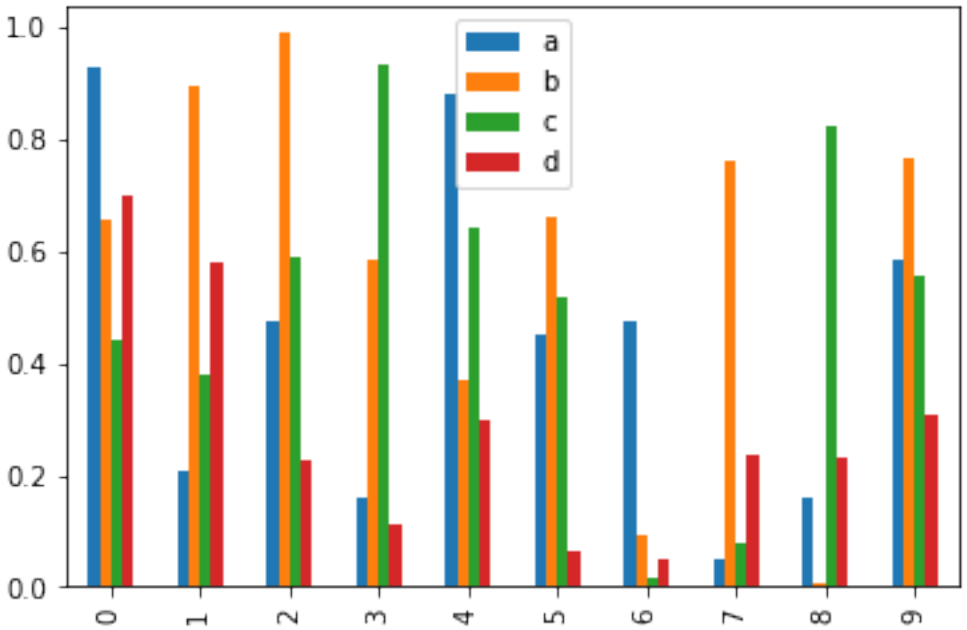


[96]:
```
car_sales.plot(x="Odometer (KM)",y="Price", kind="scatter")
```

[96]:
```
<AxesSubplot:xlabel='Odometer (KM)', ylabel='Price'>
```



[102]:
```
# How about a bar graph

x = np.random.rand(10,4)

df = pd.DataFrame(x,columns=["a","b","c","d"])
df.plot(kind="bar");
```

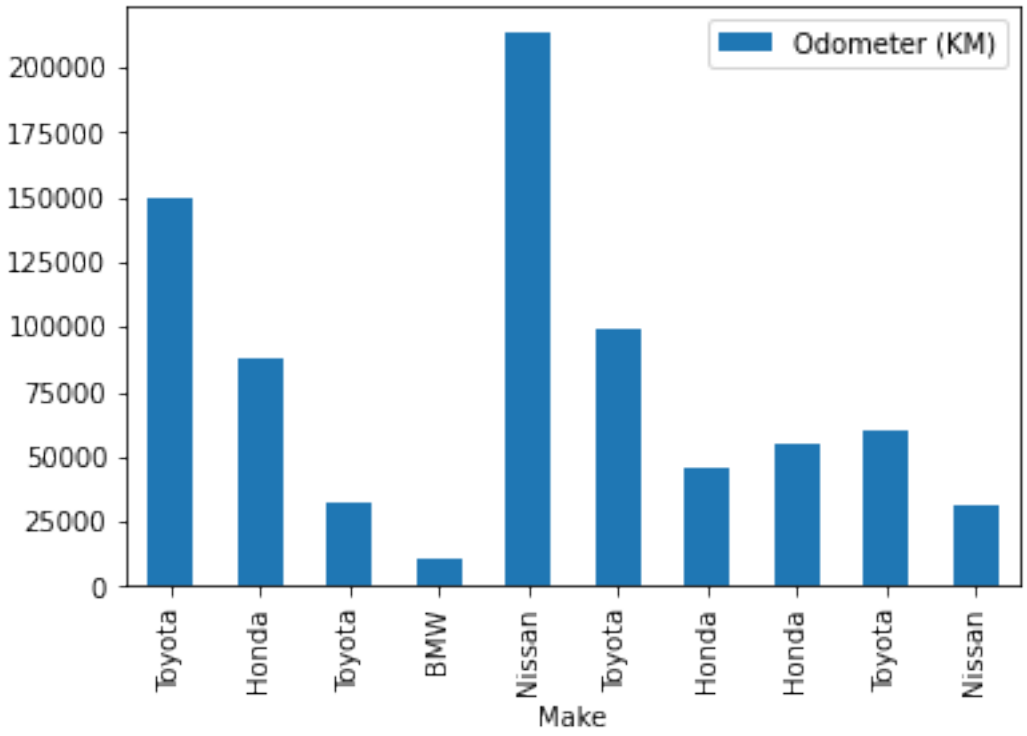


[103]:
```
car_sales
```

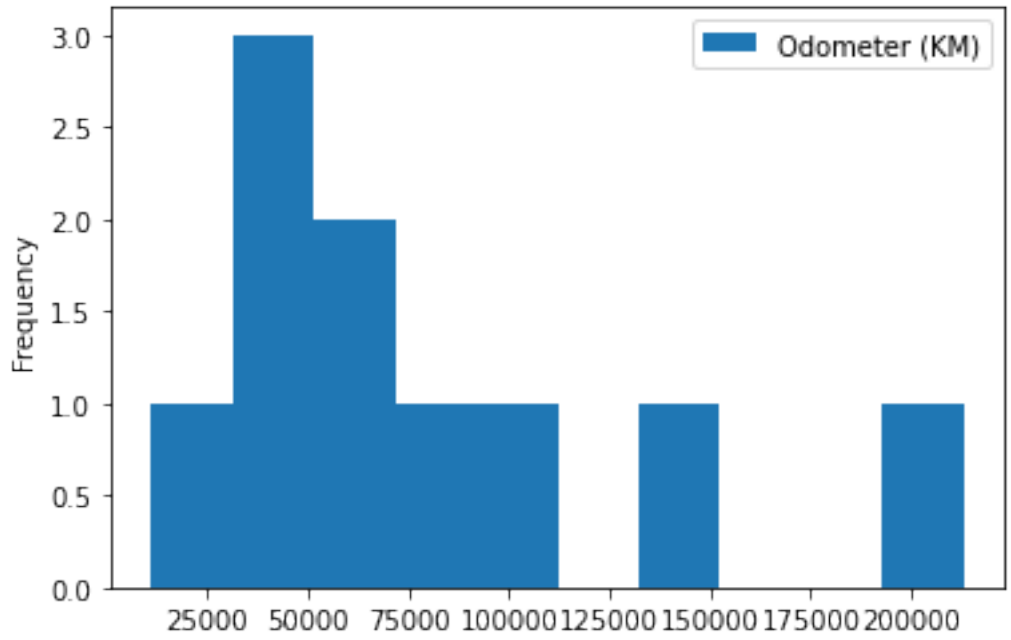
[103]:

	Make	Colour	Odometer (KM)	Doors	Price	Sales_date	Cumulative_Sales
0	Toyota	White	150043	4	4000.0	2020-01-01	4000.0
1	Honda	Red	87899	4	5000.0	2020-01-02	9000.0
2	Toyota	Blue	32549	3	7000.0	2020-01-03	16000.0
3	BMW	Black	11179	5	22000.0	2020-01-04	38000.0
4	Nissan	White	213095	4	3500.0	2020-01-05	41500.0
5	Toyota	Green	99213	4	4500.0	2020-01-06	46000.0
6	Honda	Blue	45698	4	7500.0	2020-01-07	53500.0
7	Honda	Blue	54738	4	7000.0	2020-01-08	60500.0
8	Toyota	White	60000	4	6250.0	2020-01-09	66750.0
9	Nissan	White	31600	4	9700.0	2020-01-10	76450.0

[107]:
```
car_sales.plot(kind="bar",x="Make",y="Odometer (KM)");
```



[112]:
```
# How about histograms

car_sales.plot(kind="hist",y="Odometer (KM)",bins=10);
```



[113]:
```
#Lets try on another dataset
heart_disease = pd.read_csv("./Resources/heart-disease.csv")
heart_disease
```

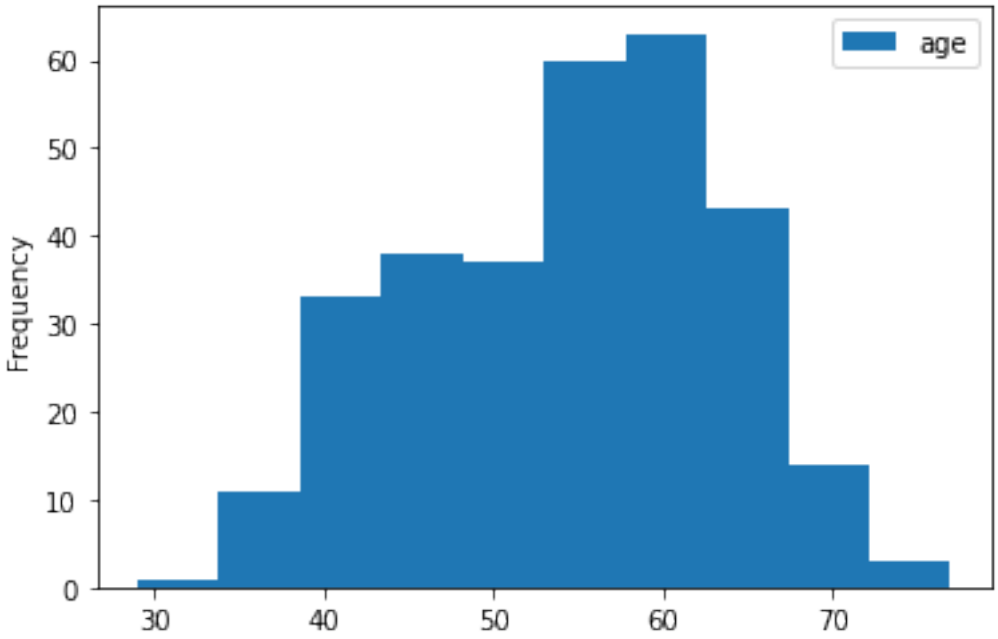
[113]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	\
0	63	1	3	145	233	1	0	150	0	2.3	
1	37	1	2	130	250	0	1	187	0	3.5	
2	41	0	1	130	204	0	0	172	0	1.4	
3	56	1	1	120	236	0	1	178	0	0.8	
4	57	0	0	120	354	0	1	163	1	0.6	
..	…	…	..	…	…	…	…	…	…	…	
298	57	0	0	140	241	0	1	123	1	0.2	
299	45	1	3	110	264	0	1	132	0	1.2	
300	68	1	0	144	193	1	1	141	0	3.4	
301	57	1	0	130	131	0	1	115	1	1.2	
302	57	0	1	130	236	0	0	174	0	0.0	

	slope	ca	thal	target
0	0	0	1	1
1	0	0	2	1
2	2	0	2	1
3	2	0	2	1
4	2	0	2	1
..	…	..	…	…
298	1	0	3	0
299	1	0	3	0
300	1	2	3	0
301	1	1	3	0
302	1	1	2	0

[303 rows x 14 columns]

[117]:
```
# Create a histogram
heart_disease.plot(kind="hist",y=["age"],bins=10);
```

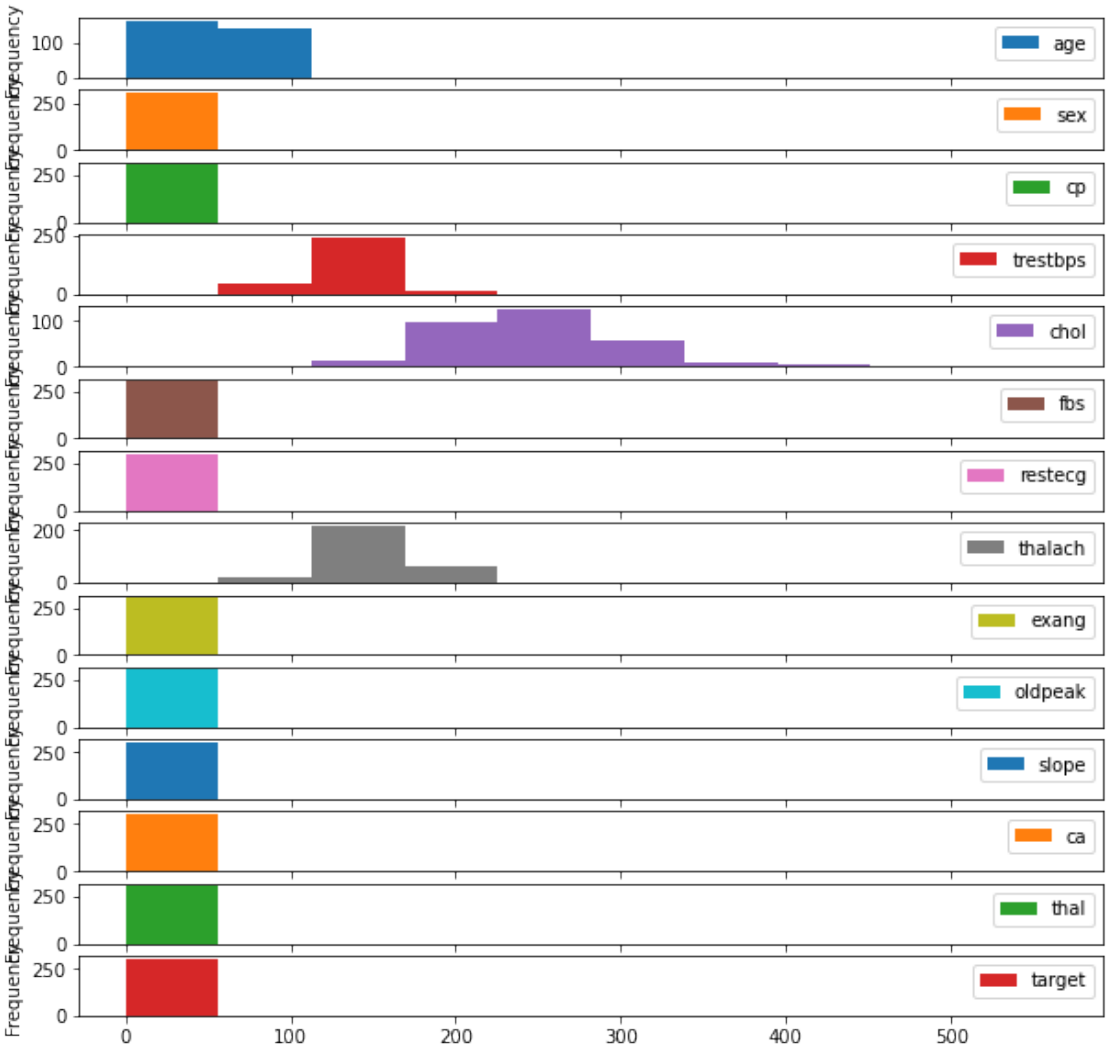


[118]:
```
heart_disease.head()
```

[118]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

[121]:
```
heart_disease.plot.hist(figsize=(10,10),subplots=True);
```



0.0.6 Which one should you use ? (pyplot Vs Matplotlib OO method?)

- When plotting something quickly, okay to use pyplot method.
- When plotting something more advanced, use OO method.

[122]:
```
heart_disease.head()
```

[122]:

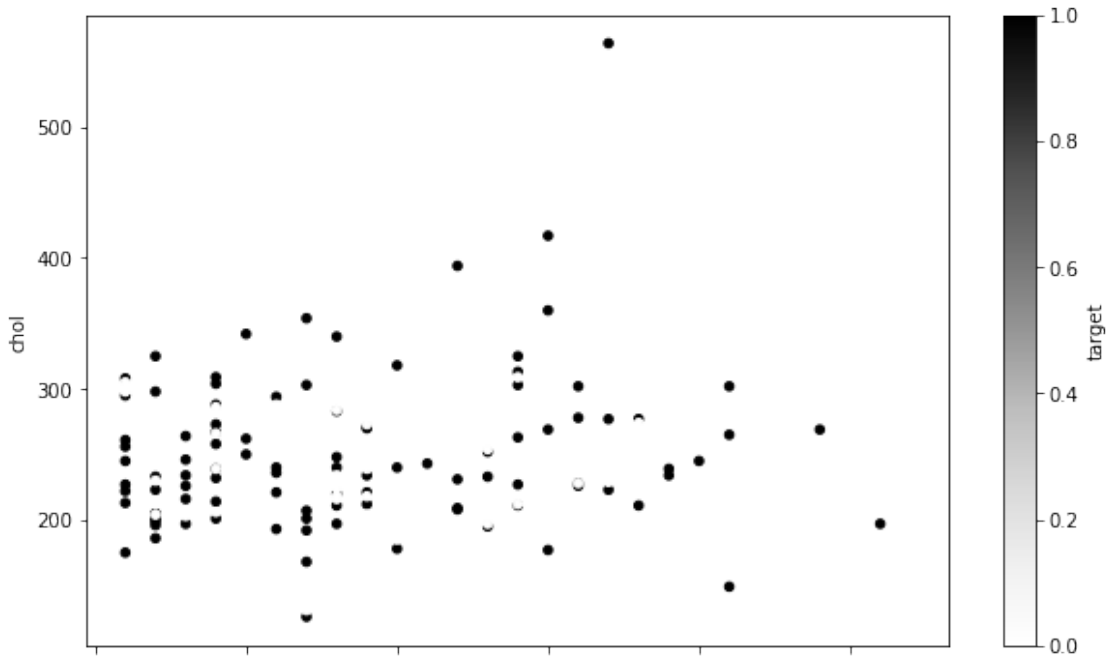
	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0			
1	37	1	2	130	250	0	1	187	0	3.5	0			
2	41	0	1	130	204	0	0	172	0	1.4	2			
3	56	1	1	120	236	0	1	178	0	0.8	2			
4	57	0	0	120	354	0	1	163	1	0.6	2			

```
0   0     1        1
1   0     2        1
2   0     2        1
3   0     2        1
4   0     2        1
```

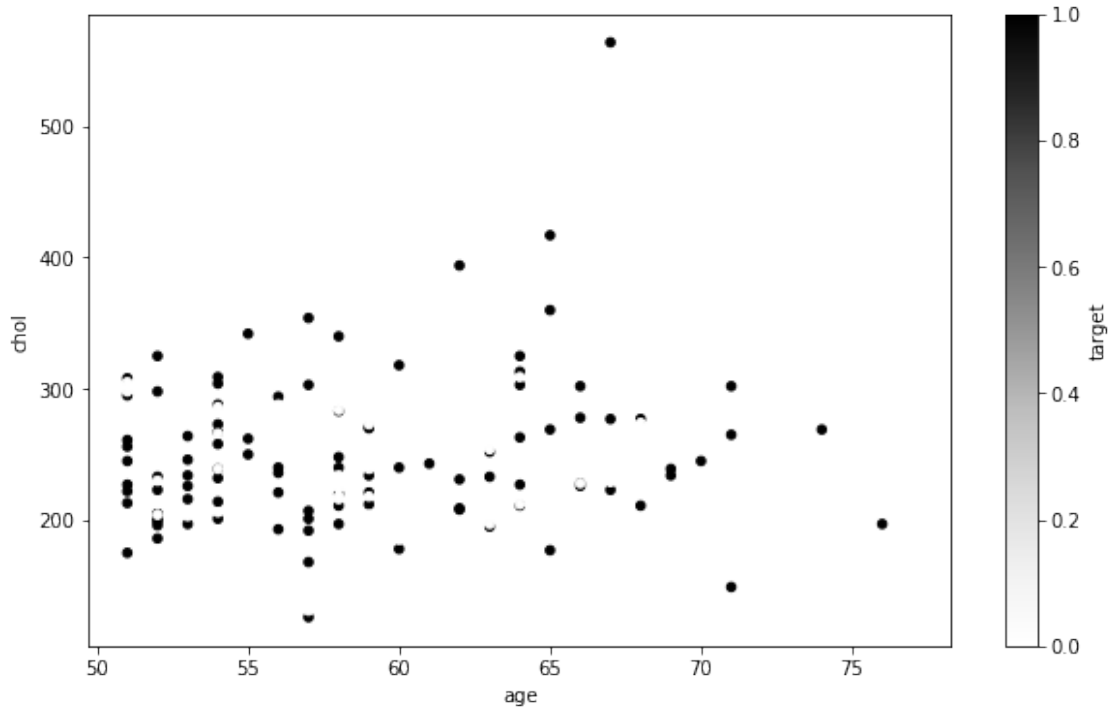
[125]:
```
#Lets create a subset of datasets
over_50=heart_disease[heart_disease["age"]> 50]
over_50
```

[125]:
```
      age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  \
0      63    1   3       145   233    1        0      150      0      2.3
3      56    1   1       120   236    0        1      178      0      0.8
4      57    0   0       120   354    0        1      163      1      0.6
5      57    1   0       140   192    0        1      148      0      0.4
6      56    0   1       140   294    0        0      153      0      1.3
..    ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
297    59    1   0       164   176    1        0       90      0      1.0
298    57    0   0       140   241    0        1      123      1      0.2
300    68    1   0       144   193    1        1      141      0      3.4
301    57    1   0       130   131    0        1      115      1      1.2
302    57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
3        2   0     2       1
4        2   0     2       1
5        1   0     1       1
6        1   0     2       1
..     ...  ..   ...     ...
297      1   2     1       0
298      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[208 rows x 14 columns]
```

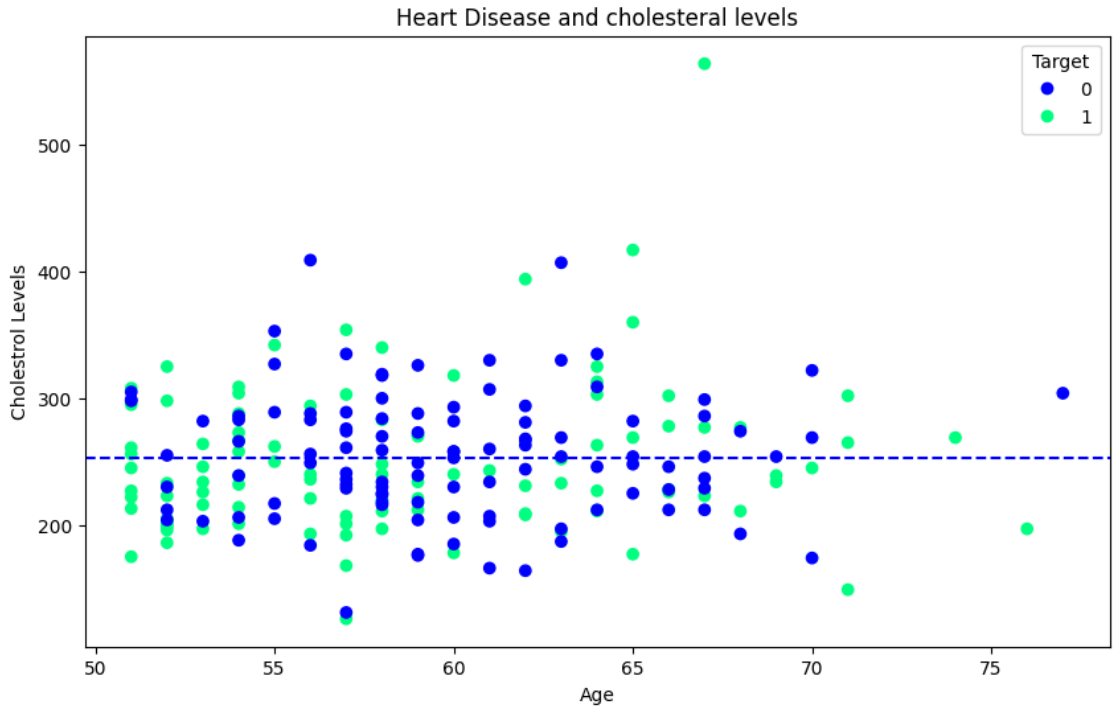
[131]:
```
#Pyplot Method
over_50.plot(kind="scatter",x="age",y="chol",c="target",figsize=(10,6));
```



[136]:
```
# OO (Object Oriented) method mixed with Plyplot method.

fig,ax=plt.subplots(figsize=(10,6))
over_50.plot(kind="scatter",x="age",y="chol",c="target",ax=ax);
#ax.set_xlim([45,80]);
```

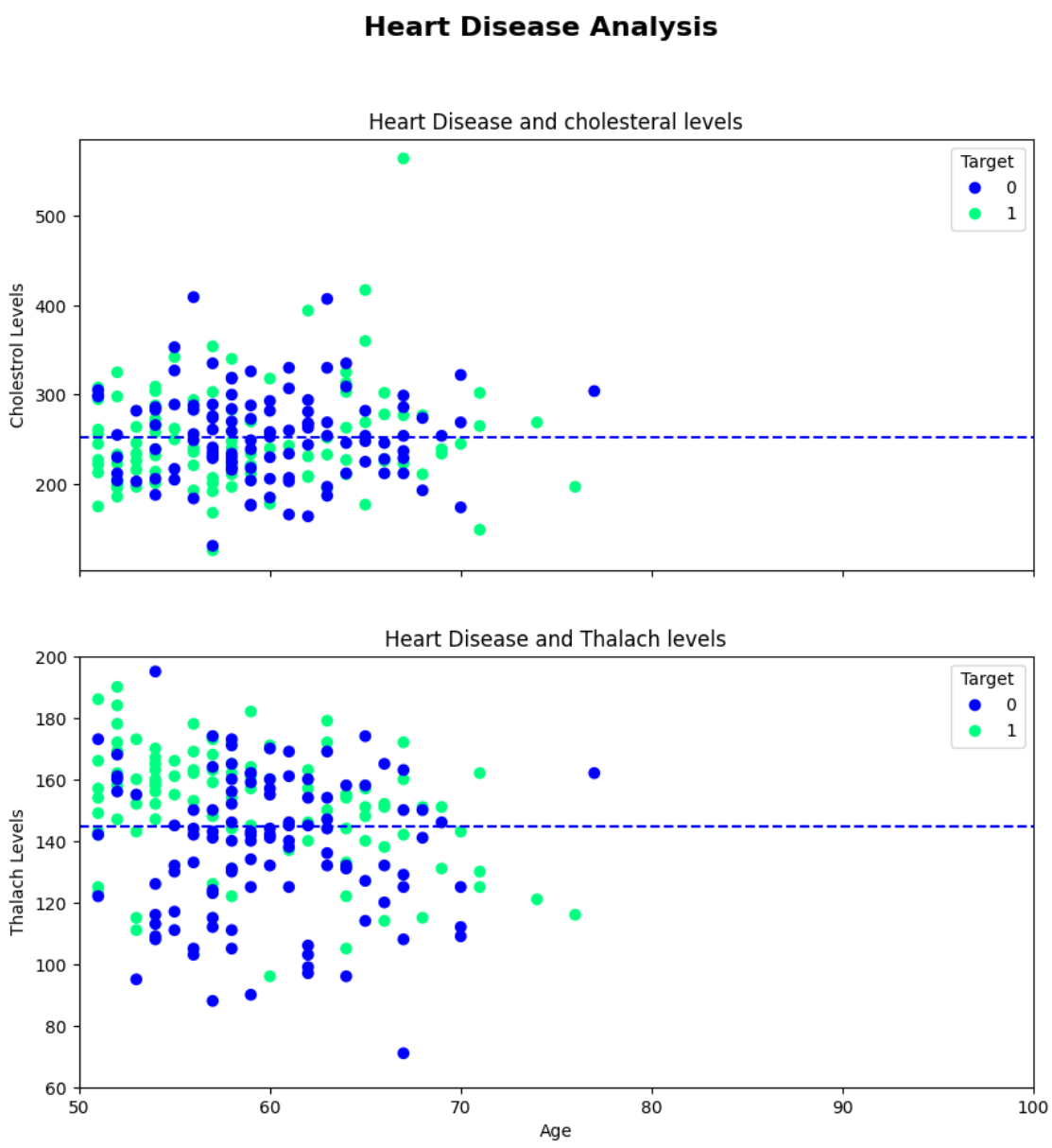


[176]:
```
## OO Method from scratch

fig,ax=plt.subplots(figsize=(10,6))
#Plot the data
scatter = ax.
 ↪scatter(x=over_50["age"],y=over_50["chol"],c=over_50["target"],cmap="winter");
 ↪

#Customise the plot

ax.set(title="Heart Disease and cholesteral␣
 ↪levels",xlabel="Age",ylabel="Cholestrol Levels");

#Add a legend
ax.legend(*scatter.legend_elements(), title="Target");

#Add a horizontal line
ax.axhline(over_50["chol"].mean(),linestyle="--",color="b");
```



[185]:
```
# Subplots of chol, age and thalach

fig,(ax0,ax1)=plt.subplots(nrows=2, ncols=1, figsize=(10,10),sharex=True)␣
 ↪#Single figure and two axes

#Plot the data
scatter = ax0.
 ↪scatter(x=over_50["age"],y=over_50["chol"],c=over_50["target"],cmap="winter");
 ↪ #Cmap allows to set the colors
scatter = ax1.
 ↪scatter(x=over_50["age"],y=over_50["thalach"],c=over_50["target"],cmap="winter");
 ↪ #Cmap allows to set the colors

#Customise the plot

ax0.set(title="Heart Disease and cholesteral levels",ylabel="Cholestrol␣
 ↪Levels");
ax1.set(title="Heart Disease and Thalach levels",xlabel="Age",ylabel="Thalach␣
 ↪Levels");

#Add a legend
ax0.legend(*scatter.legend_elements(), title="Target"); #uncover legend and set␣
 ↪the title as target for axis 0
```

```
ax1.legend(*scatter.legend_elements(), title="Target"); #uncover legend and set␣
 ↪the title as target for axis 1

#Add a horizontal line
ax0.axhline(over_50["chol"].mean(),linestyle="--",color="b");
ax1.axhline(over_50["thalach"].mean(),linestyle="--",color="b");

#Add a title to the figure
fig.suptitle("Heart Disease Analysis", fontsize=16, fontweight="bold")

#Set Axis limits
ax1.set_xlim([50,100]);
ax1.set_ylim([60,200]);
```



0.0.7 Customising Matplotlib

Choose colormap Cmap from the documentations - https://matplotlib.org/stable/users/explain/colors/colormaps.html

[158]:
```
# View the different style availables
plt.style.available
```

[158]:
```
['Solarize_Light2',
 '_classic_test_patch',
 '_mpl-gallery',
 '_mpl-gallery-nogrid',
 'bmh',
 'classic',
 'dark_background',
 'fast',
 'fivethirtyeight',
 'ggplot',
 'grayscale',
 'seaborn',
 'seaborn-bright',
 'seaborn-colorblind',
 'seaborn-dark',
 'seaborn-dark-palette',
 'seaborn-darkgrid',
 'seaborn-deep',
 'seaborn-muted',
 'seaborn-notebook',
 'seaborn-paper',
 'seaborn-pastel',
 'seaborn-poster',
 'seaborn-talk',
 'seaborn-ticks',
 'seaborn-white',
 'seaborn-whitegrid',
 'tableau-colorblind10']
```

[174]:
```
plt.style.use('default')
```

[ ]:
```

```