22mcb1002

April 7, 2023

[65]:
```
import pandas as pd
import numpy as np
from sklearn.preprocessing import MaxAbsScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import RobustScaler
```

[2]:
```
df = pd.read_csv('/content/drive/MyDrive/Data Analytics/heart.csv')
```

[3]:
```
df.columns
```

[3]:
```
Index(['id', 'age', 'sex', 'cp', 'trestbps', 'chol', 'restecg', 'thalch',
       'exang', 'oldpeak', 'slope', 'ca'],
      dtype='object')
```

[4]:
```
df.head()
```

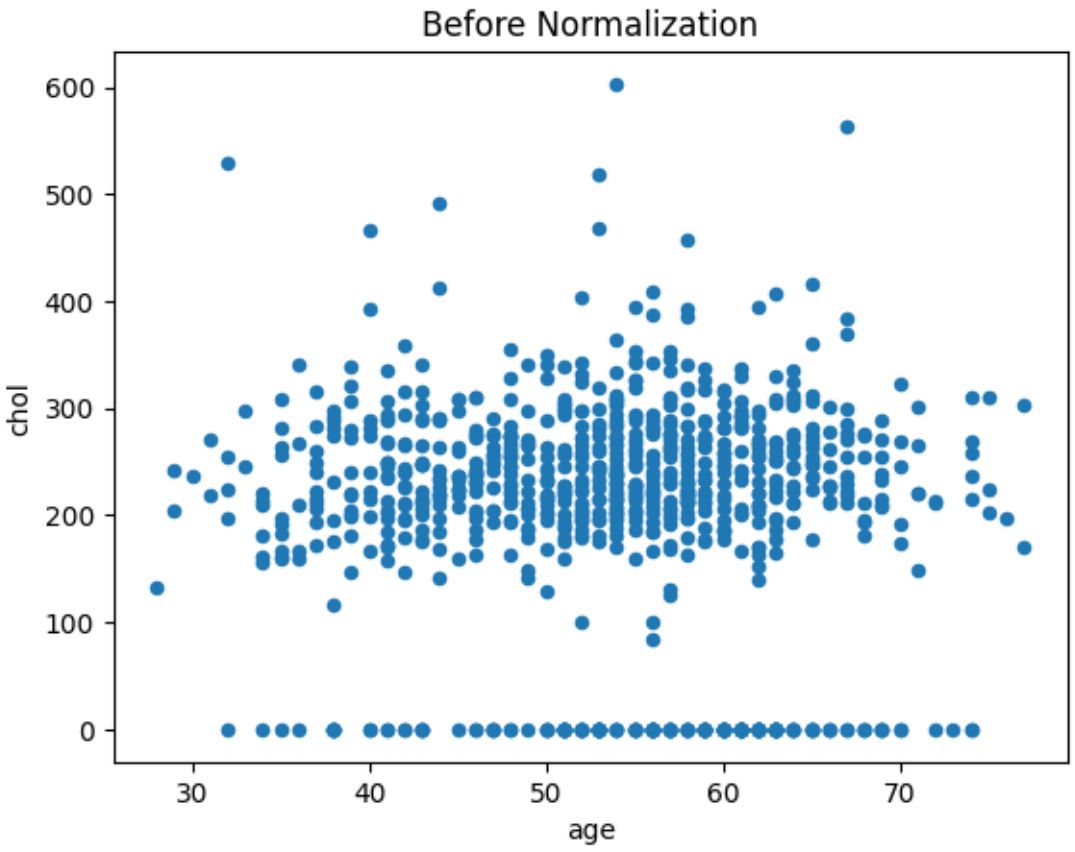
[4]:

	id	age	sex	cp	trestbps	chol	restecg	thalch	exang	oldpeak	slope	ca
0	1	63	Male	typical angina	145.0	233.0	lv hypertrophy	150.0	False	2.3	downsloping	0.0
1	2	67	Male	asymptomatic	160.0	286.0	lv hypertrophy	108.0	True	1.5	flat	3.0
2	3	67	Male	asymptomatic	120.0	229.0	lv hypertrophy	129.0	True	2.6	flat	2.0
3	4	37	Male	non-anginal	130.0	250.0	normal	187.0	False	3.5	downsloping	0.0
4	5	41	Female	atypical angina	130.0	204.0	lv hypertrophy	172.0	False	1.4	upsloping	0.0

[39]:
```
df.plot.scatter(x = 'age', y = 'chol', title = 'Before Normalization')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  scatter = ax.scatter(
```

[39]: <Axes: title={'center': 'Before Normalization'}, xlabel='age', ylabel='chol'>



[7]:
```
df.columns[:4]
```

[7]:
```
Index(['id', 'age', 'sex', 'cp'], dtype='object')
```

[9]:
```
data_numeric = df.select_dtypes(include = 'number')
data_numeric.columns
```

[9]:
```
Index(['id', 'age', 'trestbps', 'chol', 'thalch', 'oldpeak', 'ca'],
dtype='object')
```

1 Maximum absolute scaling

Without using predifined function

[49]:
```
df1 = df[['age', 'chol']]
def max_abs_scaling(df1):
  df_scaled = df1.copy()
```

```
  for col in df_scaled.columns[:2]:
    df_scaled[col] = df_scaled[col]/df_scaled[col].abs().max()
  return df_scaled
df_cars_scaled = max_abs_scaling(df1)
df_cars_scaled.head()
```

[49]:

	age	chol
0	0.818182	0.386401
1	0.870130	0.474295
2	0.870130	0.379768
3	0.480519	0.414594
4	0.532468	0.338308

[28]:
```
abs_scaler = MaxAbsScaler()
df1 = df[['age', 'chol']]
df1.columns
```

[28]:
```
Index(['age', 'chol'], dtype='object')
```

Calculating maximum absolute value

[30]:
```
abs_scaler.fit(df1)
MaxAbsScaler(copy = True)
```

[30]:
```
MaxAbsScaler()
```

[31]:
```
abs_scaler.max_abs_
```

[31]:
```
array([ 77., 603.])
```

Transforming data

[33]:
```
scaled_data = abs_scaler.transform(df1)
scaled_data.dtype
```

[33]:
```
dtype('float64')
```

Using predefined function

[42]:
```
df_scaled_data = pd.DataFrame(scaled_data, columns = df1.columns)
df_scaled_data.head()
```

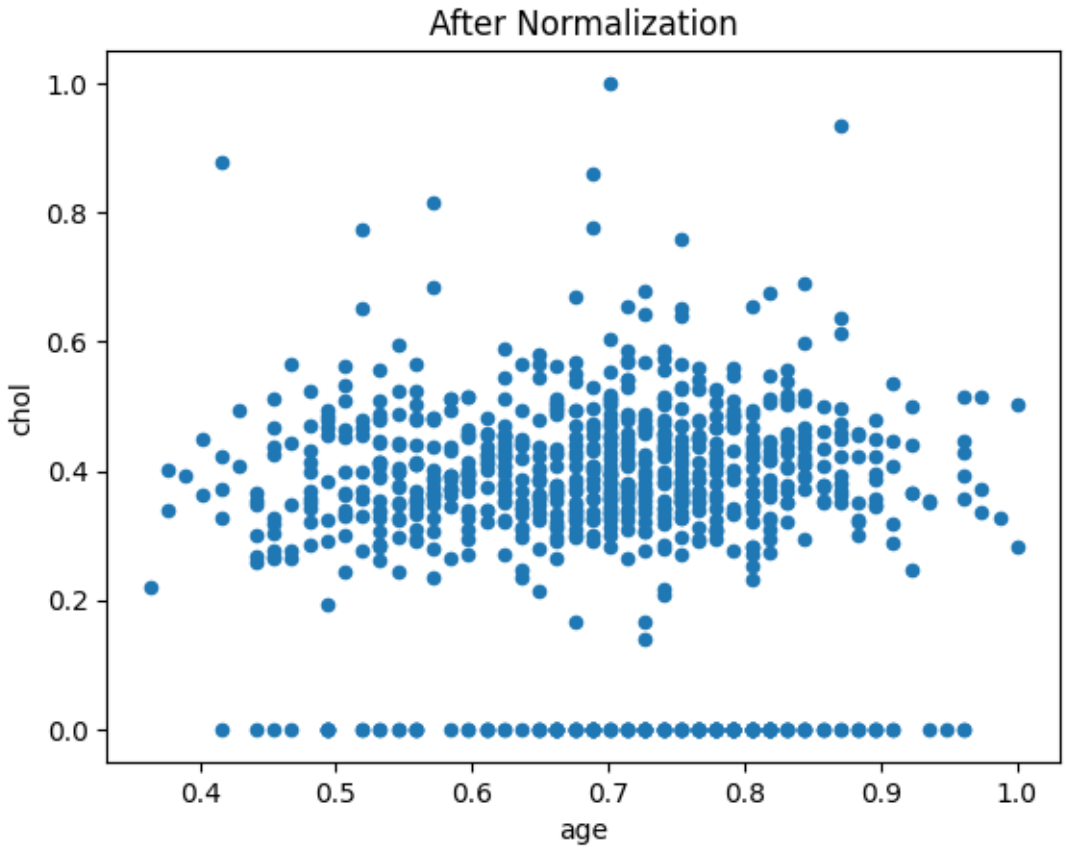
[42]:

	age	chol
0	0.818182	0.386401
1	0.870130	0.474295
2	0.870130	0.379768
3	0.480519	0.414594
4	0.532468	0.338308

[43]:
```
df_scaled_data.plot.scatter(x = 'age', y = 'chol', title = 'After␣
↪Normalization')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  scatter = ax.scatter(
```

[43]:
```
<Axes: title={'center': 'After Normalization'}, xlabel='age', ylabel='chol'>
```



2 Min-Max Normalisation

Without using predifined function

[45]:
```
def min_max_scaling(df):
  df_scaled = df.copy()
  for col in df_scaled.columns:
    df_scaled[col] = (df_scaled[col]-df_scaled[col].min())/(df_scaled[col].
↪max()-df_scaled[col].min())
```

```
  return df_scaled
df_cars_scaled = min_max_scaling(df1)
df_cars_scaled.head()
```

[45]:

	age	chol
0	0.714286	0.386401
1	0.795918	0.474295
2	0.795918	0.379768
3	0.183673	0.414594
4	0.265306	0.338308

Using predifined function

[48]:
```
MMscaler = MinMaxScaler()
df_norm = pd.DataFrame(MMscaler.fit_transform(df1), columns = df1.columns)
df_norm.head()
```

[48]:

	age	chol
0	0.714286	0.386401
1	0.795918	0.474295
2	0.795918	0.379768
3	0.183673	0.414594
4	0.265306	0.338308

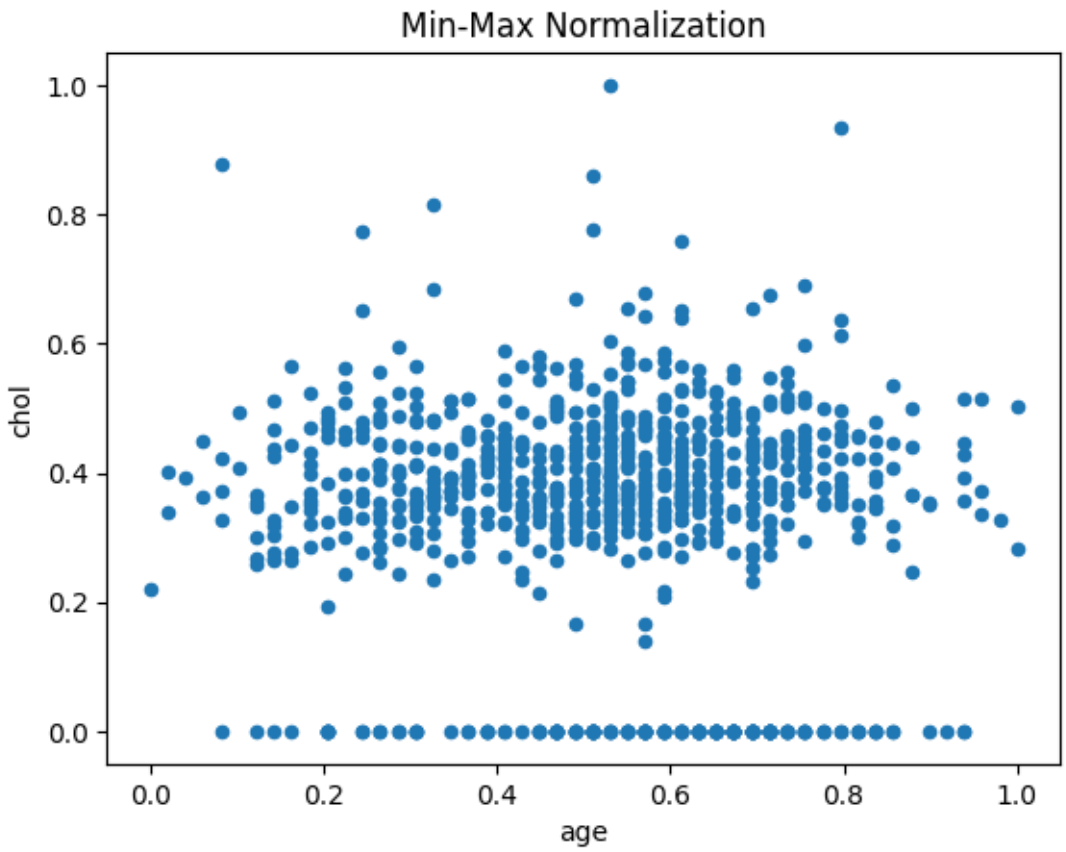
[51]:
```
MMscaler.data_min_
#MMscaler.data_max_
```

[51]:
```
array([28.,  0.])
```

[52]:
```
df_norm.plot.scatter(x = 'age', y = 'chol', title = 'Min-Max Normalization')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  scatter = ax.scatter(
```

[52]:
```
<Axes: title={'center': 'Min-Max Normalization'}, xlabel='age', ylabel='chol'>
```



3 Z Score method

Without using predefined function

[53]:
```
def z_norm(df):
  df_scaled=df.copy()
  for col in df_scaled.columns:
    df_scaled[col]=(df_scaled[col]-df_scaled[col].mean())/(df_scaled[col].std())
  return df_scaled
df_scaled_cars = z_norm(df1)
df_scaled_cars.head()
```

[53]:

	age	chol
0	1.006838	0.305736
1	1.431255	0.784158
2	1.431255	0.269628
3	-1.751875	0.459192
4	-1.327458	0.043958

Using predefined function

[55]:
```
stdscaler = StandardScaler()
stdscaler_data = stdscaler.fit_transform(df1)
stdscaler_df = pd.DataFrame(stdscaler_data, columns = df1.columns)
stdscaler_df.head()
```

[55]:

	age	chol
0	1.007386	0.305908
1	1.432034	0.784599
2	1.432034	0.269780
3	-1.752828	0.459450
4	-1.328180	0.043982

[56]:
```
stdscaler.scale_
```

[56]:
```
array([  9.41956171, 110.71855645])
```

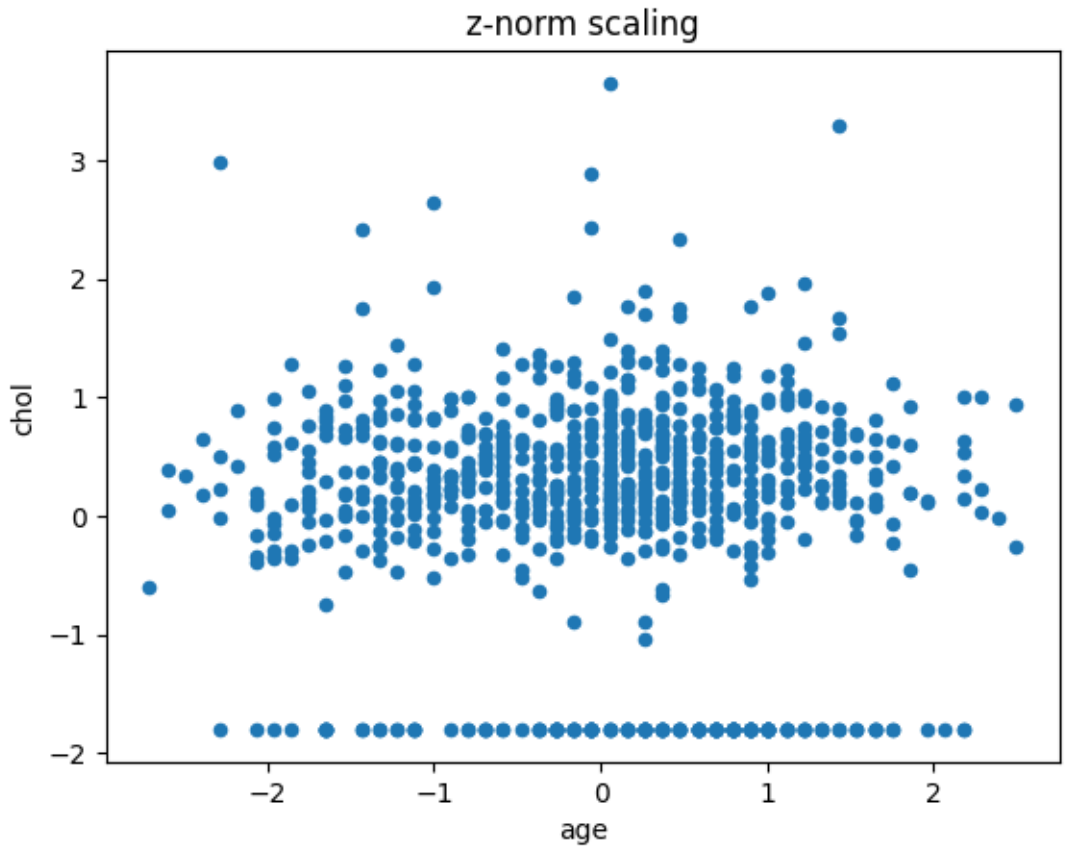
[57]:
```
stdscaler.mean_
```

[57]:
```
array([ 53.51086957, 199.13033708])
```

[58]:
```
stdscaler_df.plot.scatter(x = 'age', y = 'chol', title = "z-norm scaling")
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  scatter = ax.scatter(
```

[58]:
```
<Axes: title={'center': 'z-norm scaling'}, xlabel='age', ylabel='chol'>
```



4 Robust Scaling

Without using predefined function

[59]:
```
def robust_scaling(df):
  df_scaled = df.copy()
  for col in df_scaled.columns:
    df_scaled[col] = (df_scaled[col]-df_scaled[col].median())/(df_scaled[col].
 ↪quantile(0.75)-df_scaled[col].quantile(0.25))
  return df_scaled
df_scaled=robust_scaling(df1)
df_scaled.head()
```

[59]:

	age	chol
0	0.692308	0.107527
1	1.000000	0.677419
2	1.000000	0.064516
3	-1.307692	0.290323
4	-1.000000	-0.204301

Using predefined function

[62]:
```
rob_scale = RobustScaler()
rob_scaledata = rob_scale.fit_transform(df1)
rob_scaled_df = pd.DataFrame(rob_scaledata, columns = df1.columns)
rob_scaled_df.head()
```

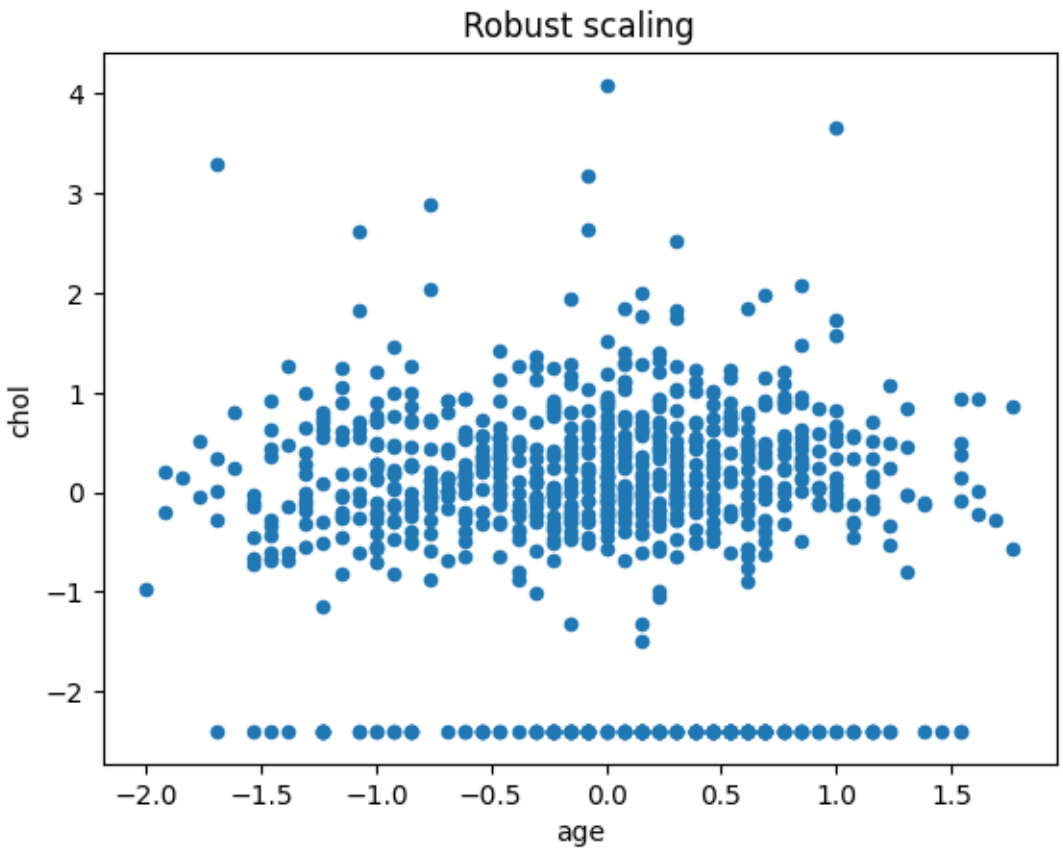
[62]:

	age	chol
0	0.692308	0.107527
1	1.000000	0.677419
2	1.000000	0.064516
3	-1.307692	0.290323
4	-1.000000	-0.204301

[63]:
```
rob_scaled_df.plot.scatter(x = 'age', y = 'chol',title = 'Robust scaling')
```

```
/usr/local/lib/python3.9/dist-packages/pandas/plotting/_matplotlib/core.py:1114:
UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will
be ignored
  scatter = ax.scatter(
```

[63]: <Axes: title={'center': 'Robust scaling'}, xlabel='age', ylabel='chol'>



5 Binning

Binning by distance - use cut()

[64]:
```
min_age = df1['age'].min()
max_age = df1['age'].max()
print(min_age)
print(max_age)
```

```
28
77
```

[66]:
```
# Returns equally spaced values
np.linspace(1,10, 4)
```

[66]:
```
array([ 1.,  4.,  7., 10.])
```

[67]:
```
bins = np.linspace(min_age, max_age, 4)
bins
```

[67]:
```
array([28.        , 44.33333333, 60.66666667, 77.        ])
```

[69]:
```
labels =['young','middle','senior']
df1.columns
```

[69]:
```
Index(['age', 'chol'], dtype='object')
```

[70]:
```
df1['Age_categ'] = pd.cut(df1['age'], bins = bins, labels = labels,␣
 ↪include_lowest = True)
df1.columns
```

```
<ipython-input-70-017515a25297>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df1['Age_categ'] = pd.cut(df1['age'], bins = bins, labels = labels,
include_lowest = True)
```

[70]:
```
Index(['age', 'chol', 'Age_categ'], dtype='object')
```

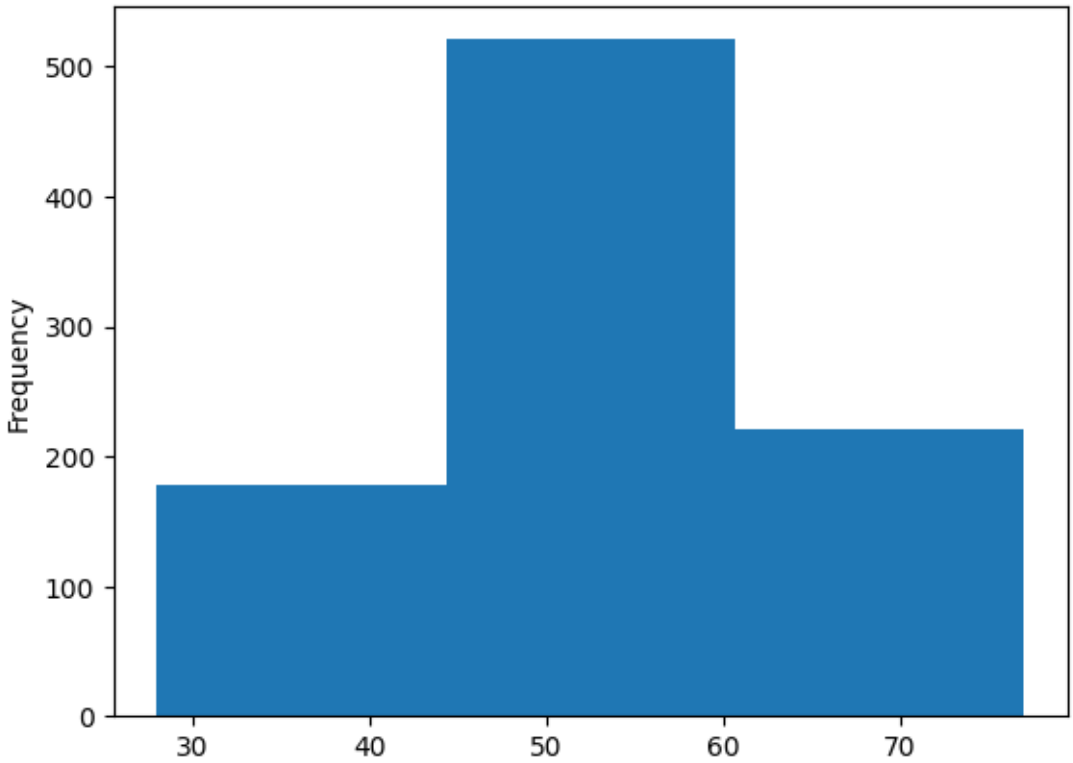
[71]:
```
df1.head()
```

[71]:

	age	chol	Age_categ
0	63	233.0	senior
1	67	286.0	senior
2	67	229.0	senior
3	37	250.0	young
4	41	204.0	young

[72]:
```
df1['age'].plot.hist(bins = 3)
```

[72]:
```
<Axes: ylabel='Frequency'>
```



Binning by frequency - use qcut()

[73]:
```
df1['Age_categ_freq'] = pd.qcut(df1['age'], q = 3, labels = labels, precision =␣
↪1)
df1.columns
```

```
<ipython-input-73-0bf73af84618>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-
docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
```

```
  df1['Age_categ_freq'] = pd.qcut(df1['age'], q = 3, labels = labels, precision
= 1)
```

[73]:
```
Index(['age', 'chol', 'Age_categ', 'Age_categ_freq'], dtype='object')
```

[74]:
```
df1.head()
```

[74]:

	age	chol	Age_categ	Age_categ_freq
0	63	233.0	senior	senior
1	67	286.0	senior	senior
2	67	229.0	senior	senior
3	37	250.0	young	young
4	41	204.0	young	young