Chapter 1: Review of the Core Modules of NumPy and Pandas

In [1]:
```
import pandas as pd
import numpy as np
```

In [2]:
```
adult_df = pd.read_csv('adult.csv')
adult_df.set_index(np.arange(10000,42561),inplace=True)
```

In [3]:
```
adult_df.iloc[5:7,0:2]
```

Out[3]:

	age	workclass
10005	37	Private
10006	49	Private

In [4]:
```
adult_df.loc['10005':'10007','age':'fnlwgt']
```

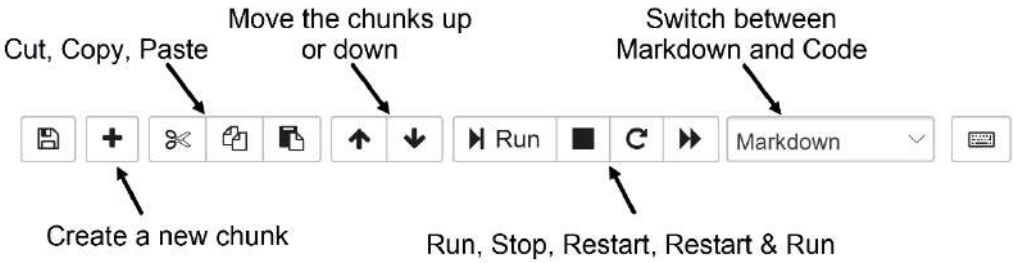
Out[4]:

	age	workclass	fnlwgt
10005	37	Private	284582
10006	49	Private	160187
10007	52	Self-emp-not-inc	209642

```
Hello World! This is a Markdown chunk!
```

In [1]:
```
print('Hello World! This is Code Chunk!')
```

```
Hello World! This is Code Chunk!
```



In [2]:
```
import numpy as np
```

In [3]:
```
lst_nums = [2,5,7,11,13,17,23,31,37,41,43,47]
np.mean(lst_nums)
```

Out[3]: 23.083333333333332

In [4]:
```
lst_nums = [2,5,7,11,13,17,23,31,37,41,43,47]
ary_nums = np.array(lst_nums)
ary_nums.mean()
```

Out[4]: 23.083333333333332

In [5]:
```
type(lst_nums)
```

Out[5]: list

In [6]:
```
type(ary_nums)
```

Out[6]: numpy.ndarray

In [7]:
```
np.arange(15)
```

Out[7]:
```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [8]:
```
np.arange(5,15)
```

Out[8]:
```
array([ 5,  6,  7,  8,  9, 10, 11, 12, 13, 14])
```

In [9]:
```
np.arange(-7.1,7)
```

Out[9]:
```
array([-7.1, -6.1, -5.1, -4.1, -3.1, -2.1, -1.1, -0.1,  0.9,  1.9,  2.9,
        3.9,  4.9,  5.9,  6.9])
```

In [10]:
```
np.zeros([4,5])
```

Out[10]:
```
array([[0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.],
       [0., 0., 0., 0., 0.]])
```

In [11]:
```
np.ones(7)
```

Out[11]:
```
array([1., 1., 1., 1., 1., 1., 1.])
```

In [12]:
```
Names = ['Jevon', 'Dawn', 'Kayleigh', 'Jadene', 'Kennedy', 'Kaydee',
         'Ansh', 'Flynn', 'Kier', 'Clarence']
Math_grades = [80, 50, 60, 70, 60, 100, 70, 70, 60, 70]
Science_grades = [90, 80, 50, 50, 60, 50, 90, 70, 80, 80]
History_grades = [60, 90, 50, 90, 100, 100, 100, 100, 90, 70]
```

In [13]:
```
Average_grades = np.zeros(10)
print(Average_grades)

for i, name in enumerate(Names):
    Average_grades[i] = np.mean([Math_grades[i],Science_grades[i],
                                 History_grades[i]])

print(Average_grades)
```

```
[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
[76.66666667 73.33333333 53.33333333 70.         73.33333333 83.33333333
 86.66666667 80.         76.66666667 73.33333333]
```

In [14]:
```
# better-looking report

for i, name in enumerate(Names):
    print("Average for {} : {}".format(name,Average_grades[i]))
```

```
Average for Jevon : 76.66666666666667
Average for Dawn : 73.33333333333333
Average for Kayleigh : 53.333333333333336
Average for Jadene : 70.0
Average for Kennedy : 73.33333333333333
Average for Kaydee : 83.33333333333333
Average for Ansh : 86.66666666666667
Average for Flynn : 80.0
Average for Kier : 76.66666666666667
Average for Clarence : 73.33333333333333
```

In [15]:
```
np.linspace(0,1,21)
```

Out[15]:
```
array([0.  , 0.05, 0.1 , 0.15, 0.2 , 0.25, 0.3 , 0.35, 0.4 , 0.45, 0.5 ,
       0.55, 0.6 , 0.65, 0.7 , 0.75, 0.8 , 0.85, 0.9 , 0.95, 1.  ])
```

In [16]:
```
np.linspace(10,1000,100)
```

Out[16]:
```
array([  10.,   20.,   30.,   40.,   50.,   60.,   70.,   80.,   90.,
        100.,  110.,  120.,  130.,  140.,  150.,  160.,  170.,  180.,
        190.,  200.,  210.,  220.,  230.,  240.,  250.,  260.,  270.,
        280.,  290.,  300.,  310.,  320.,  330.,  340.,  350.,  360.,
        370.,  380.,  390.,  400.,  410.,  420.,  430.,  440.,  450.,
        460.,  470.,  480.,  490.,  500.,  510.,  520.,  530.,  540.,
        550.,  560.,  570.,  580.,  590.,  600.,  610.,  620.,  630.,
        640.,  650.,  660.,  670.,  680.,  690.,  700.,  710.,  720.,
        730.,  740.,  750.,  760.,  770.,  780.,  790.,  800.,  810.,
        820.,  830.,  840.,  850.,  860.,  870.,  880.,  890.,  900.,
        910.,  920.,  930.,  940.,  950.,  960.,  970.,  980.,  990.,
       1000.])
```

In [16]:
```
Candidates = np.linspace(-1000,1000,2001)
#print(Candidates)

for candidate in Candidates:
    if(candidate**2 - 5*candidate +6 ==0):
        print("Just found a possible answer: {}".format(candidate))
```

```
Just found a possible answer: 2.0
Just found a possible answer: 3.0
```

In [17]:
```
import pandas as pd

adult_df = pd.read_csv('adult.csv')
adult_df.head()
```

Out[17]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	race	
0	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	
1	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	
2	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	
3	53	Private	234721	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	
4	28	Private	338409	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	F

In [18]:
```
type(adult_df.age)
```

Out[18]: pandas.core.series.Series

In [19]:
```
type(adult_df)
```

Out[19]: pandas.core.frame.DataFrame

In [20]:
```
adult_df.loc[0].index
```

Out[20]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capitalGain', 'capitalLoss', 'hoursPerWeek', 'nativeCountry',
       'income'],
      dtype='object')
```

In [21]:
```
adult_df.age.index
```

Out[21]:
```
RangeIndex(start=0, stop=32561, step=1)
```

In [22]:
```
adult_df.set_index(np.arange(10000,42561),inplace=True)
```

In [23]:
```
adult_df.set_index(np.arange(10000,42561))
```

Out[23]:

	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationship	ra
10000	39	State-gov	77516	Bachelors	13	Never-married	Adm-clerical	Not-in-family	Whi
10001	50	Self-emp-not-inc	83311	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	Whi
10002	38	Private	215646	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	Whi

In [24]:
```
adult_df.education-num
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
<ipython-input-24-a83283968914> in <module>
----> 1 adult_df.education-num

NameError: name 'num' is not defined
```

In [25]:
```
adult_df.iloc[2].loc['education']
```

Out[25]: 'HS-grad'

In [26]:
```
adult_df.education.loc[10002]
```

Out[26]: 'HS-grad'

In [27]:
```
adult_df['education'].iloc[2]
```

Out[27]: 'HS-grad'

In [28]:
```
adult_df.at[10002,'education']
```

Out[28]: 'HS-grad'

In [29]:
```
row_series = adult_df.loc[10002]
print(row_series.loc['education'])
print(row_series.iloc[3])
print(row_series['education'])
print(row_series.education)
```

```
HS-grad
HS-grad
HS-grad
HS-grad
```

In [30]:
```
columns_series = adult_df.education
print(columns_series.loc[10002])
print(columns_series.iloc[2])
print(columns_series[10002])
# print(row_series.10002)  This will give syntax error!
```

```
HS-grad
HS-grad
HS-grad
```

In [31]:
```
my_array = np.array([[2,3,5,7],[11,13,17,19],
                     [23,29,31,37,], [41,43,47,49]])
my_array
```

Out[31]:
```
array([[ 2,  3,  5,  7],
       [11, 13, 17, 19],
       [23, 29, 31, 37],
       [41, 43, 47, 49]])
```

In [32]:
```
my_array[1,1]
```

Out[32]:
```
13
```

In [33]:
```
my_array[1,:]
```

Out[33]:
```
array([11, 13, 17, 19])
```

In [34]:
```
my_array[:,1]
```

Out[34]:
```
array([ 3, 13, 29, 43])
```

In [35]:
```
my_array
```

Out[35]:
```
array([[ 2,  3,  5,  7],
       [11, 13, 17, 19],
       [23, 29, 31, 37],
       [41, 43, 47, 49]])
```

In [36]:
```
my_array[1:3,:]
```

Out[36]:
```
array([[11, 13, 17, 19],
       [23, 29, 31, 37]])
```

In [37]:
```
my_array[1:3,0:2]
```

Out[37]:
```
array([[11, 13],
       [23, 29]])
```

In [38]:
```
my_array[1:3,[0,2]]
```

Out[38]:
```
array([[11, 17],
       [23, 31]])
```

In [39]:
```
adult_df.loc[:,'education':'occupation']
```

Out[39]:

	education	education-num	marital-status	occupation
10000	Bachelors	13	Never-married	Adm-clerical
10001	Bachelors	13	Married-civ-spouse	Exec-managerial
10002	HS-grad	9	Divorced	Handlers-cleaners

In [40]:
```
adult_df.sort_values('education-num').reset_index().iloc[1:32561:3617]
```

Out[40]:

	index	age	workclass	fnlwgt	education	education-num	marital-status	occupation	relationsh
1	23248	68	Private	168794	Preschool	1	Never-married	Machine-op-inspct	Not-in-fam
3618	19607	25	Private	251854	11th	7	Never-married	Adm-clerical	Own-chi
7235	38845	31	Private	272856	HS-grad	9	Never-married	Craft-repair	Own-chi
10852	32759	56	Private	182273	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husbar
14469	10419	34	State-gov	240283	HS-grad	9	Divorced	Transport-moving	Unmarri
18086	31532	25	Self-emp-inc	98756	Some-college	10	Divorced	Adm-clerical	Own-chi
21703	17245	37	Federal-gov	40955	Some-college	10	Never-married	Other-service	Own-chi
25320	40595	43	Private	342567	Bachelors	13	Married-spouse-absent	Adm-clerical	Unmarri
28937	15200	43	Federal-gov	144778	Bachelors	13	Never-married	Exec-managerial	Not-in-fami
32554	27308	55	Self-emp-not-inc	53566	Doctorate	16	Divorced	Exec-managerial	Not-in-fami

In [41]:
```
twopowers_sr = pd.Series([1,2,4,8,16,32,64,128,256,512,1024])
BM = [False,False,False,True,False,False,False,True,True,True,True]
twopowers_sr[BM]
```

Out[41]:
```
3        8
7      128
8      256
9      512
10    1024
```

1	2	4	8	16	32	64	128	256	512	1024
False	False	False	True	False	False	False	True	True	True	True
●	●	●	○	●	●	●	○	○	○	○
			8				128	256	512	1024

In [42]:
```
twopowers_sr >=500
```

Out[42]:
```
0     False
1     False
2     False
3     False
4     False
5     False
6     False
7     False
8     False
9      True
10     True
dtype: bool
```

In [43]:
```
BM = twopowers_sr >=500
twopowers_sr[BM]
```

Out[43]:
```
9      512
10    1024
dtype: int64
```

In [44]:
```
twopowers_sr[twopowers_sr >=500]
```

Out[44]:
```
9      512
10    1024
dtype: int64
```

In [45]:
```
BM = adult_df.education == 'Preschool'
print('Mean: {}'.format(np.mean(adult_df[BM].age)))
print('Median: {}'.format(np.median(adult_df[BM].age)))
```

```
Mean: 42.76470588235294
Median: 41.0
```

In [46]:
```
BM1 = adult_df['education-num'] > 10
BM2 = adult_df['education-num'] < 10

print('More than 10 years of education - Capital Gain: {}'
      .format(np.mean(adult_df[BM1].capitalGain)))
print('Less than 10 years of education - Capital Gain: {}'
      .format(np.mean(adult_df[BM2].capitalGain)))
```

```
More than 10 years of education - Capital Gain: 2230.9397109166985
Less than 10 years of education - Capital Gain: 492.25532059102613
```

In [47]:
```
adult_df.shape
```

Out[47]: (32561, 15)

In [48]:
```
adult_df.columns
```

Out[48]:
```
Index(['age', 'workclass', 'fnlwgt', 'education', 'education-num',
       'marital-status', 'occupation', 'relationship', 'race', 'sex',
       'capitalGain', 'capitalLoss', 'hoursPerWeek', 'nativeCountry',
       'income'],
      dtype='object')
```

In [49]:
```
adult_df.columns = ['age', 'workclass', 'fnlwgt', 'education',
                    'education_num', 'marital_status', 'occupation',
                    'relationship', 'race', 'sex', 'capitalGain',
                    'capitalLoss', 'hoursPerWeek', 'nativeCountry',
                    'income']
```

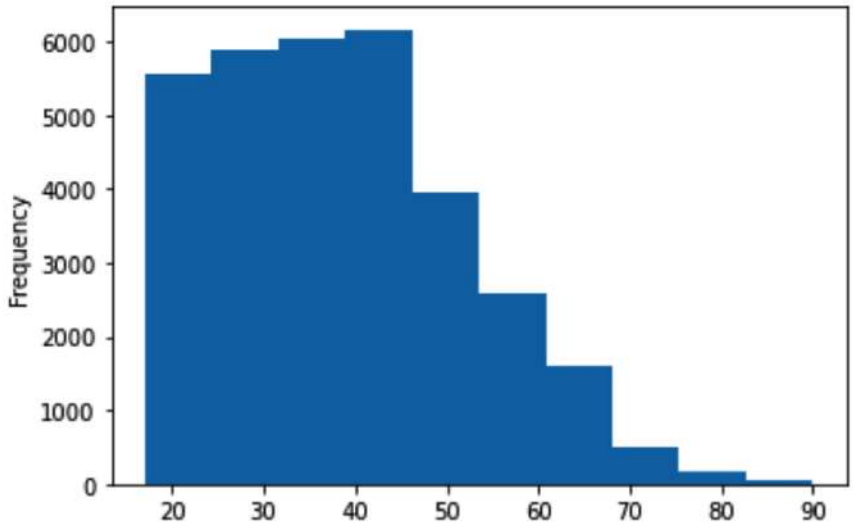
In [50]:
```
adult_df.describe()
```

Out[50]:

	age	fnlwgt	education_num	capitalGain	capitalLoss	hoursPerWee
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.00000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.43745
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.34742
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.00000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.00000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.00000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.00000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.00000

In [51]:
```
adult_df.age.plot.hist()
```

Out[51]: <matplotlib.axes._subplots.AxesSubplot at 0x29b11f8d8b0>



In [52]:
```
adult_df.relationship.unique()
```

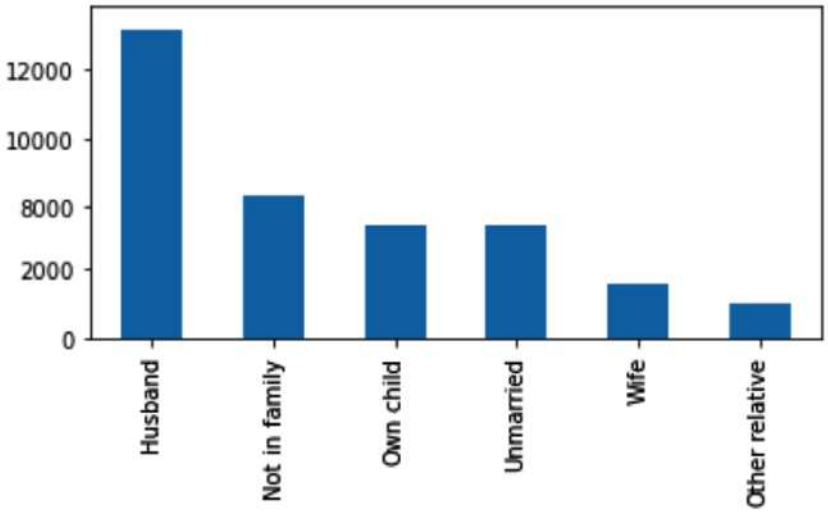
Out[52]:
```
array(['Not-in-family', 'Husband', 'Wife', 'Own-child', 'Unmarried',
       'Other-relative'], dtype=object)
```

In [53]:
```
adult_df.relationship.value_counts()
```

Out[53]:
```
Husband           13193
Not-in-family      8305
Own-child          5068
Unmarried          3446
Wife               1568
Other-relative      981
Name: relationship, dtype: int64
```

In [54]:
```
adult_df.relationship.value_counts().plot.bar()
```

Out[54]: <matplotlib.axes._subplots.AxesSubplot at 0x2341fcf2f40>



In [55]:
```
def MultiplyBy2(n):
    return n*2

adult_df.age.apply(MultiplyBy2)
```

Out[55]:
```
10000     78
10001    100
10002     76
10003    106
10004     56
         ...
42556     54
42557     80
42558    116
42559     44
42560    104
```

In [60]:

```
adult_df['lifeNoEd'] = adult_df.apply(
    lambda r: r.age-r.education_num,axis=1)

adult_df['capitalNet'] = adult_df.apply(
    lambda r: r.capitalGain - r.capitalLoss,axis=1)

adult_df[['education_num','lifeNoEd','capitalNet']].corr()
```

Out[60]:

	education_num	lifeNoEd	capitalNet
education_num	1.000000	-0.150452	0.117891
lifeNoEd	-0.150452	1.000000	0.051490
capitalNet	0.117891	0.051490	1.000000

Function	Description	Function	Description
`.count()`	Number of non-null observations	`.median()`	Arithmetic median of values
`.sum()`	Sum of values	`.min()`	Minimum of values
`.mean()`	Mean of values	`.max()`	Maximum of values
`.mad()`	Mean absolute deviation	`.mode()`	Mode of values
`.std()`	Unbiased standard deviation	`.Var()`	Unbiased variance
`.sem()`	Unbiased standard error of the mean	`.Describe()`	Count(), mean(), std(), and so on
`.skew()`	Unbiased skewness	`.kurt()`	Unbiased kurtosis

In [62]:

```
adult_df.groupby(['race','sex']).capitalNet.mean()
```

Out[62]:

```
race                 sex
Amer-Indian-Eskimo   Female     530.142857
                     Male       628.864583
Asian-Pac-Islander   Female     727.583815
                     Male      1707.440115
Black                Female     471.142765
                     Male       627.268324
Other                Female     218.385321
                     Male      1314.438272
White                Female     508.219857
                     Male      1266.413112
```

In [63]:
```
grb_result =adult_df.groupby(['race','sex']).capitalNet.mean()

print(grb_result.index)
```

```
MultiIndex([('Amer-Indian-Eskimo', 'Female'),
            ('Amer-Indian-Eskimo',   'Male'),
            ('Asian-Pac-Islander', 'Female'),
            ('Asian-Pac-Islander',   'Male'),
            (             'Black', 'Female'),
            (             'Black',   'Male'),
            (             'Other', 'Female'),
            (             'Other',   'Male'),
            (             'White', 'Female'),
            (             'White',   'Male')],
           names=['race', 'sex'])
```

In [64]:
```
grb_result =adult_df.groupby(['race','sex']).capitalNet.mean()
grb_result
```

Out[64]:
```
race                sex
Amer-Indian-Eskimo  Female     530.142857
                    Male       628.864583
Asian-Pac-Islander  Female     727.583815
                    Male      1707.440115
Black               Female     471.142765
                    Male       627.268324
Other               Female     218.385321
                    Male      1314.438272
White               Female     508.219857
                    Male      1266.413112
Name: capitalNet, dtype: float64
```

In [65]:
```
grb_result.unstack()
```

Out[65]:

sex race	Female	Male
Amer-Indian-Eskimo	530.142857	628.864583
Asian-Pac-Islander	727.583815	1707.440115
Black	471.142765	627.268324
Other	218.385321	1314.438272
White	508.219857	1266.413112

In [66]:
```
mlt_seris =adult_df.groupby(['race','sex','income']).fnlwgt.mean()
mlt_seris
```

Out[66]:
```
race                sex     income
Amer-Indian-Eskimo  Female  <=50K     0.001764
                            >50K      0.002395
                    Male    <=50K     0.002046
                            >50K      0.001954
Asian-Pac-Islander  Female  <=50K     0.002398
                            >50K      0.002305
                    Male    <=50K     0.002652
                            >50K      0.002762
Black               Female  <=50K     0.003454
                            >50K      0.003331
                    Male    <=50K     0.003922
                            >50K      0.003971
Other               Female  <=50K     0.002803
                            >50K      0.002593
                    Male    <=50K     0.003478
                            >50K      0.003310
White               Female  <=50K     0.002969
                            >50K      0.002978
                    Male    <=50K     0.003074
                            >50K      0.003025
Name: fnlwgt, dtype: float64
```

In [67]:
```
mlt_seris.unstack()
```

Out[67]:

	income	<=50K	>50K
race	sex		
Amer-Indian-Eskimo	Female	0.001764	0.002395
	Male	0.002046	0.001954
Asian-Pac-Islander	Female	0.002398	0.002305
	Male	0.002652	0.002762
Black	Female	0.003454	0.003331
	Male	0.003922	0.003971
Other	Female	0.002803	0.002593
	Male	0.003478	0.003310
White	Female	0.002969	0.002978
	Male	0.003074	0.003025

In [68]:
```
mlt_seris.unstack().unstack()
```

Out[68]:

income	<=50K		>50K	
sex	Female	Male	Female	Male
race				
Amer-Indian-Eskimo	0.001764	0.002046	0.002395	0.001954
Asian-Pac-Islander	0.002398	0.002652	0.002305	0.002762
Black	0.003454	0.003922	0.003331	0.003971
Other	0.002803	0.003478	0.002593	0.003310
White	0.002969	0.003074	0.002978	0.003025

In [69]:
```
mlt_seris.unstack().unstack()
```

Out[69]:

income	<=50K		>50K	
sex	Female	Male	Female	Male
race				
Amer-Indian-Eskimo	0.001764	0.002046	0.002395	0.001954
Asian-Pac-Islander	0.002398	0.002652	0.002305	0.002762
Black	0.003454	0.003922	0.003331	0.003971
Other	0.002803	0.003478	0.002593	0.003310
White	0.002969	0.003074	0.002978	0.003025

In [70]:
```
mlt_df= mlt_seris.unstack().unstack()
mlt_df.columns
```

Out[70]:
```
MultiIndex([('<=50K', 'Female'),
            ('<=50K',   'Male'),
            ( '>50K', 'Female'),
            ( '>50K',   'Male')],
           names=['income', 'sex'])
```

In [71]:
```
mlt_df.stack()
```

Out[71]:

	income	<=50K	>50K
race	sex		
Amer-Indian-Eskimo	Female	0.001764	0.002395
	Male	0.002046	0.001954
Asian-Pac-Islander	Female	0.002398	0.002305
	Male	0.002652	0.002762
Black	Female	0.003454	0.003331
	Male	0.003922	0.003971
Other	Female	0.002803	0.002593
	Male	0.003478	0.003310
White	Female	0.002969	0.002978
	Male	0.003074	0.003025

In [72]:
```
mlt_df.stack().stack()
```

Out[72]:
```
race                sex     income
Amer-Indian-Eskimo  Female  <=50K     0.001764
                            >50K      0.002395
                    Male    <=50K     0.002046
                            >50K      0.001954
Asian-Pac-Islander  Female  <=50K     0.002398
                            >50K      0.002305
                    Male    <=50K     0.002652
                            >50K      0.002762
Black               Female  <=50K     0.003454
                            >50K      0.003331
                    Male    <=50K     0.003922
                            >50K      0.003971
Other               Female  <=50K     0.002803
                            >50K      0.002593
                    Male    <=50K     0.003478
                            >50K      0.003310
White               Female  <=50K     0.002969
                            >50K      0.002978
                    Male    <=50K     0.003074
                            >50K      0.003025
dtype: float64
```

	ReadingDateTime	NO	NO2	NOX	PM10	PM2.5
0	01/01/2017 00:00	3.5	30.8	36.2	35.7	31.0
1	01/01/2017 01:00	3.6	31.5	37.0	28.5	31.0
2	01/01/2017 02:00	2.2	27.3	30.7	22.7	31.0

`wide_df`

	ReadingDateTime	Species	Value
0	01/01/2017 00:00	NO	3.5
1	01/01/2017 01:00	NO	3.6
2	01/01/2017 02:00	NO	2.2
3	01/01/2017 00:00	NO2	30.8
4	01/01/2017 01:00	NO2	31.5
5	01/01/2017 02:00	NO2	27.3
6	01/01/2017 00:00	NOX	36.2
7	01/01/2017 01:00	NOX	37.0
8	01/01/2017 02:00	NOX	30.7
9	01/01/2017 00:00	PM10	35.7
10	01/01/2017 01:00	PM10	28.5
11	01/01/2017 02:00	PM10	22.7
12	01/01/2017 00:00	PM2.5	31.0
13	01/01/2017 01:00	PM2.5	31.0
14	01/01/2017 02:00	PM2.5	31.0

`long_df`

In [73]:

```
wide_df = pd.read_csv('wide.csv')
wide_df
```

Out[73]:

	ReadingDateTime	NO	NO2	NOX	PM10	PM2.5
0	01/01/2017 00:00	3.5	30.8	36.2	35.7	31.0
1	01/01/2017 01:00	3.6	31.5	37.0	28.5	31.0
2	01/01/2017 02:00	2.2	27.3	30.7	22.7	31.0

In [74]:

```
wide_df.melt(id_vars='ReadingDateTime',
             value_vars=['NO','NO2','NOX','PM10','PM2.5'],
             var_name='Species',
             value_name='Value')
```

Out[74]:

	ReadingDateTime	Species	Value
0	01/01/2017 00:00	NO	3.5
1	01/01/2017 01:00	NO	3.6
2	01/01/2017 02:00	NO	2.2
3	01/01/2017 00:00	NO2	30.8
4	01/01/2017 01:00	NO2	31.5
5	01/01/2017 02:00	NO2	27.3
6	01/01/2017 00:00	NOX	36.2
7	01/01/2017 01:00	NOX	37.0
8	01/01/2017 02:00	NOX	30.7
9	01/01/2017 00:00	PM10	35.7
10	01/01/2017 01:00	PM10	28.5
11	01/01/2017 02:00	PM10	22.7
12	01/01/2017 00:00	PM2.5	31.0
13	01/01/2017 01:00	PM2.5	31.0
14	01/01/2017 02:00	PM2.5	31.0

In [75]:

```
long_df = pd.read_csv('long.csv')
long_df
```

Out[75]:

	ReadingDateTime	Species	Value
0	01/01/2017 00:00	NO	3.5
1	01/01/2017 01:00	NO	3.6
2	01/01/2017 02:00	NO	2.2
3	01/01/2017 00:00	NO2	30.8
4	01/01/2017 01:00	NO2	31.5
5	01/01/2017 02:00	NO2	27.3
6	01/01/2017 00:00	NOX	36.2
7	01/01/2017 01:00	NOX	37.0
8	01/01/2017 02:00	NOX	30.7
9	01/01/2017 00:00	PM10	35.7
10	01/01/2017 01:00	PM10	28.5
11	01/01/2017 02:00	PM10	22.7
12	01/01/2017 00:00	PM2.5	31.0
13	01/01/2017 01:00	PM2.5	31.0
14	01/01/2017 02:00	PM2.5	31.0

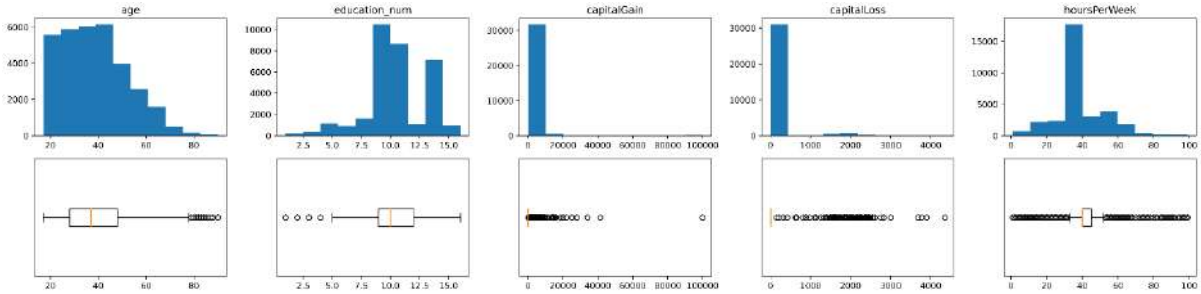
In [76]:

```
long_df.pivot(index='ReadingDateTime',
              columns='Species',
              values='Value')
```

Out[76]:

Species	NO	NO2	NOX	PM10	PM2.5
ReadingDateTime					
01/01/2017 00:00	3.5	30.8	36.2	35.7	31.0
01/01/2017 01:00	3.6	31.5	37.0	28.5	31.0
01/01/2017 02:00	2.2	27.3	30.7	22.7	31.0

Chapter 2: Review of Another Core Module – Matplotlib



In [1]:

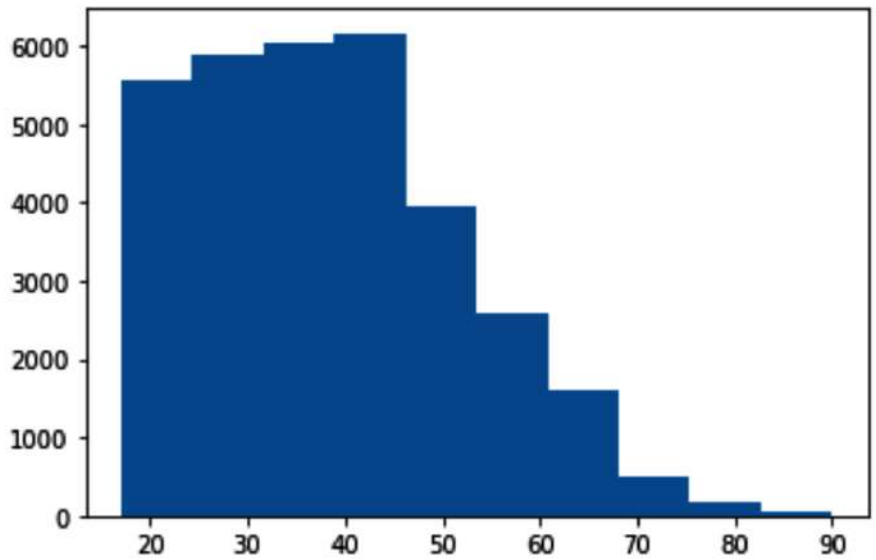
```
#from previous chapter

import pandas as pd
import numpy as np
adult_df = pd.read_csv('adult.csv')
```

In [2]:

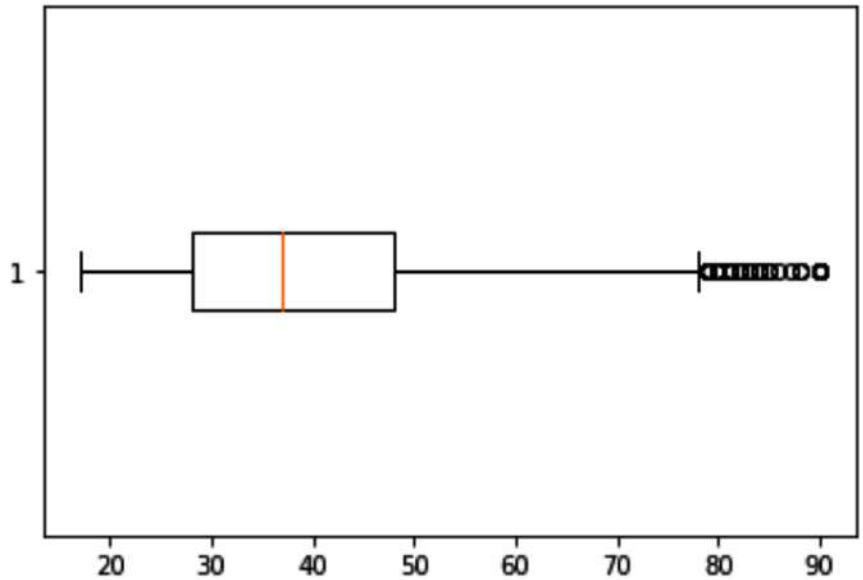
```
import matplotlib.pyplot as plt
```

In [3]:

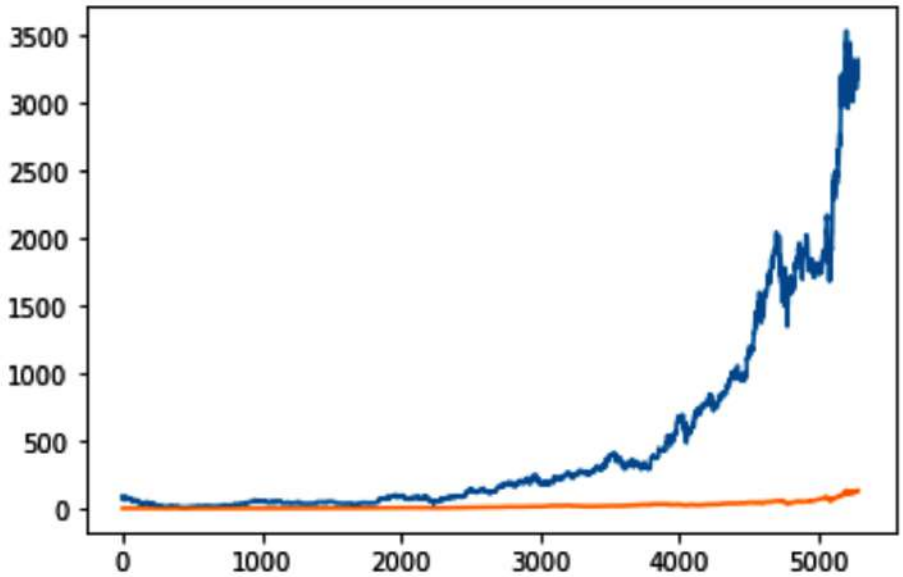
```
plt.hist(adult_df.age)
plt.show()
```



In [4]:
```
plt.boxplot(adult_df.age, vert=False)
plt.show()
```

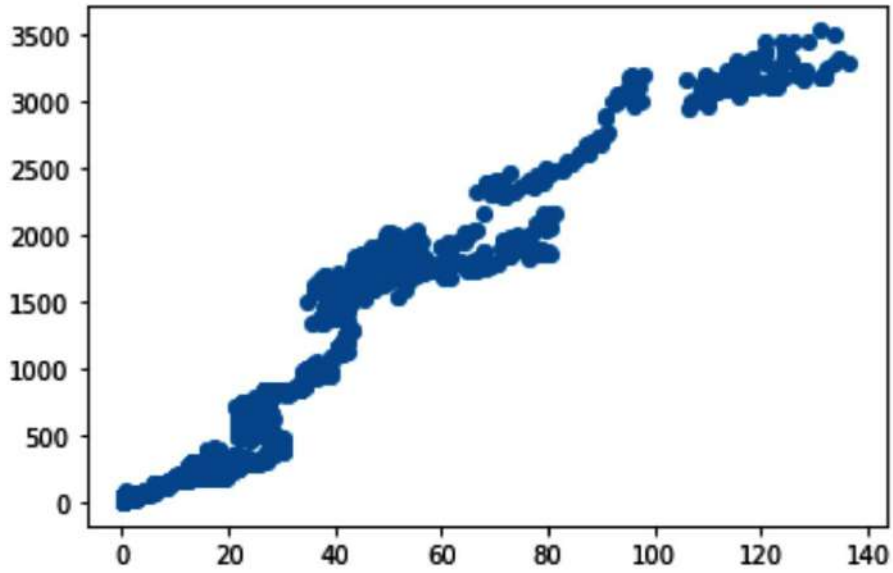


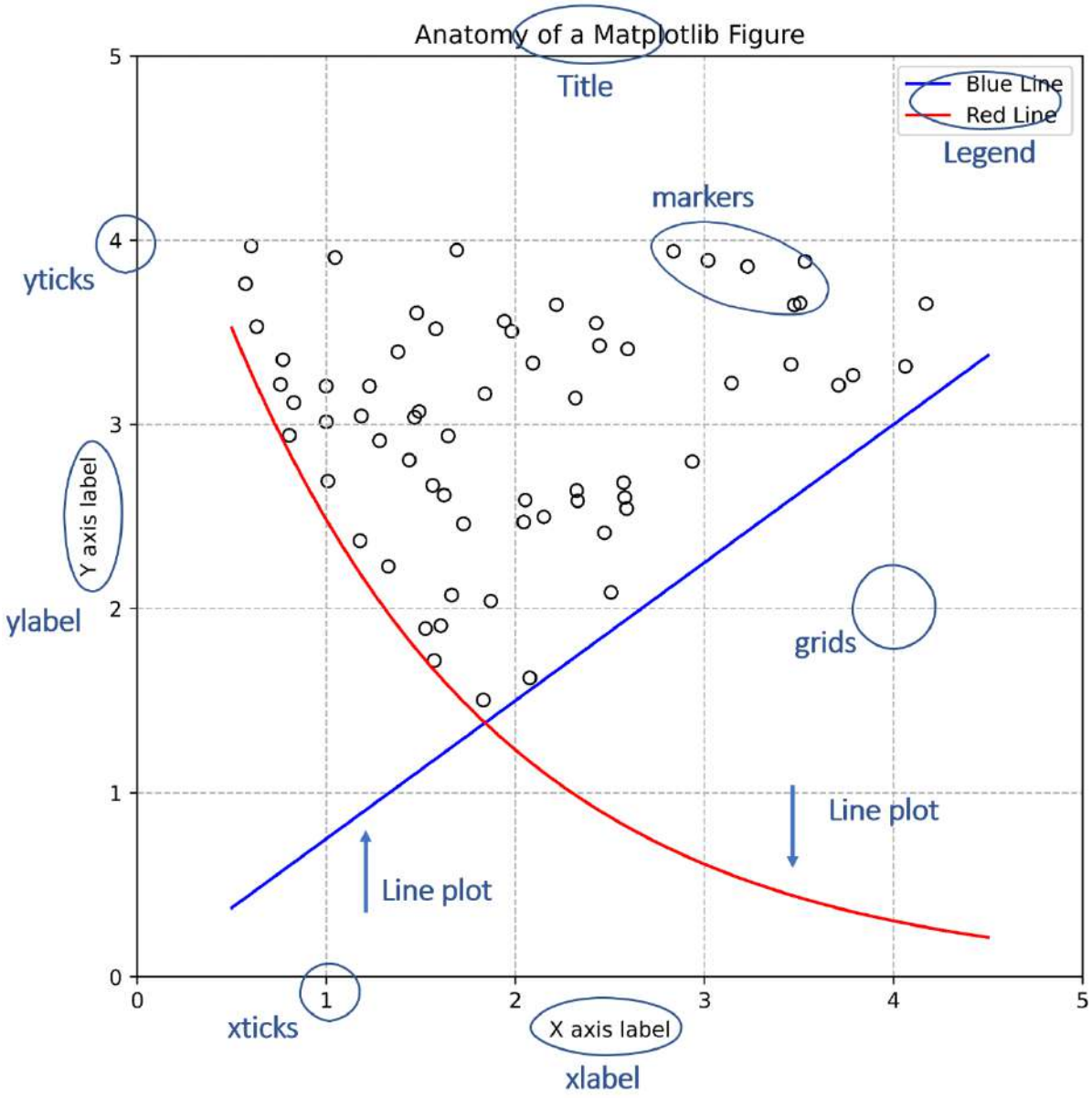
In [6]:
```
plt.plot(amz_df.Close)
plt.plot(apl_df.Close)
plt.show()
```



In [7]:

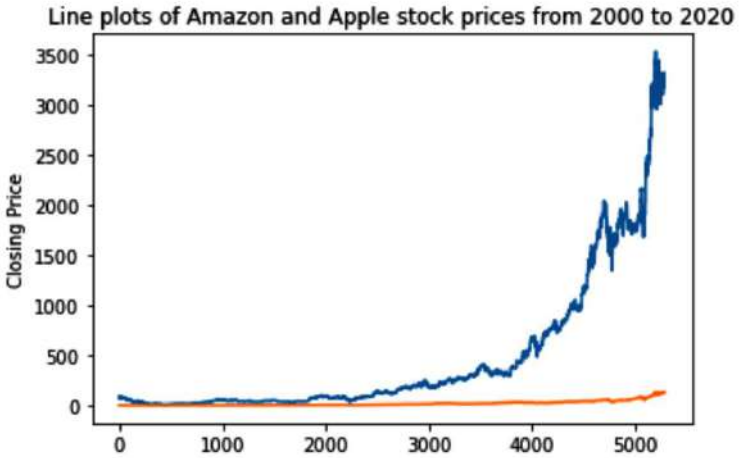
```
plt.scatter(apl_df.Close,amz_df.Close)
plt.show()
```




Anatomy of a Matplotlib Figure
Title
Blue Line
Red Line
Legend
markers
yticks
Y axis label
ylabel
grids
Line plot
Line plot
xticks
X axis label
xlabel
0
1
2
3
4
5

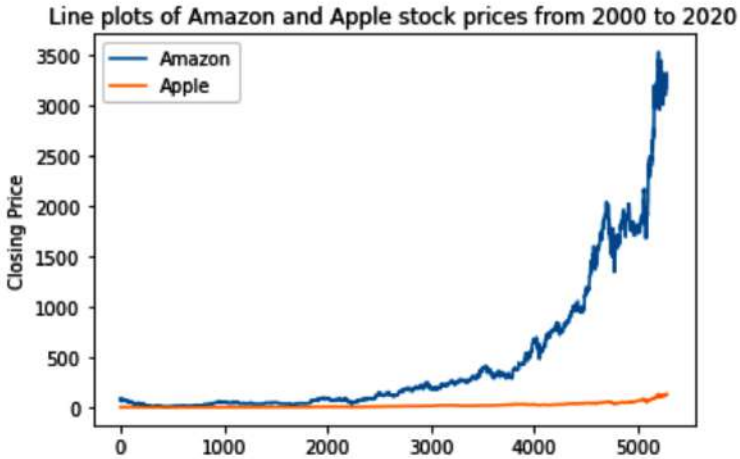
In [8]:

```
plt.plot(amz_df.Close)
plt.plot(apl_df.Close)
plt.title('Line plots of Amazon and Apple stock prices from 2000 to 2020')
plt.ylabel('Closing Price')
plt.show()
```



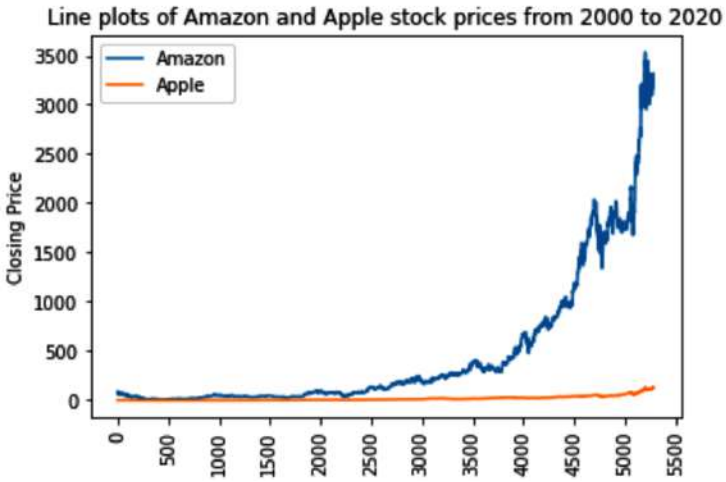
In [9]:

```
plt.plot(amz_df.Close, label='Amazon')
plt.plot(apl_df.Close, label='Apple')
plt.title('Line plots of Amazon and Apple stock prices from 2000 to 2020')
plt.ylabel('Closing Price')
plt.legend()
plt.show()
```



In [10]:

```
plt.plot(amz_df.Close, label='Amazon')
plt.plot(apl_df.Close, label='Apple')
plt.title('Line plots of Amazon and Apple stock prices from 2000 to 2020')
plt.ylabel('Closing Price')
plt.xticks([0,500,1000,1500,2000,2500,3000,3500,4000,4500,5000,5500],
           rotation=90)
plt.legend()
plt.show()
```

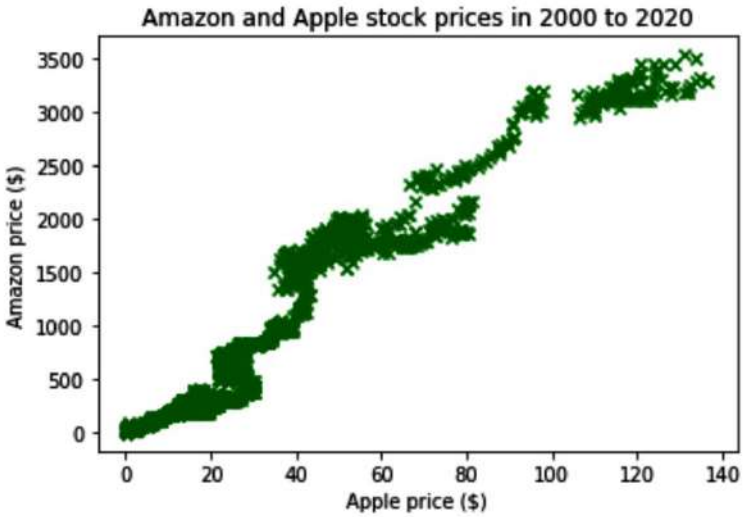


In [11]:

```
plt.plot(amz_df.Close, label='Amazon')
plt.plot(apl_df.Close, label='Apple')
plt.title('Line plots of Amazon and Apple stock prices from 2000 to 2020')
plt.ylabel('Closing Price')
plt.legend()
plt.xticks(np.arange(0,len(amz_df),250),amz_df.Date[0:len(amz_df):250],
           rotation=90)
plt.show()
```

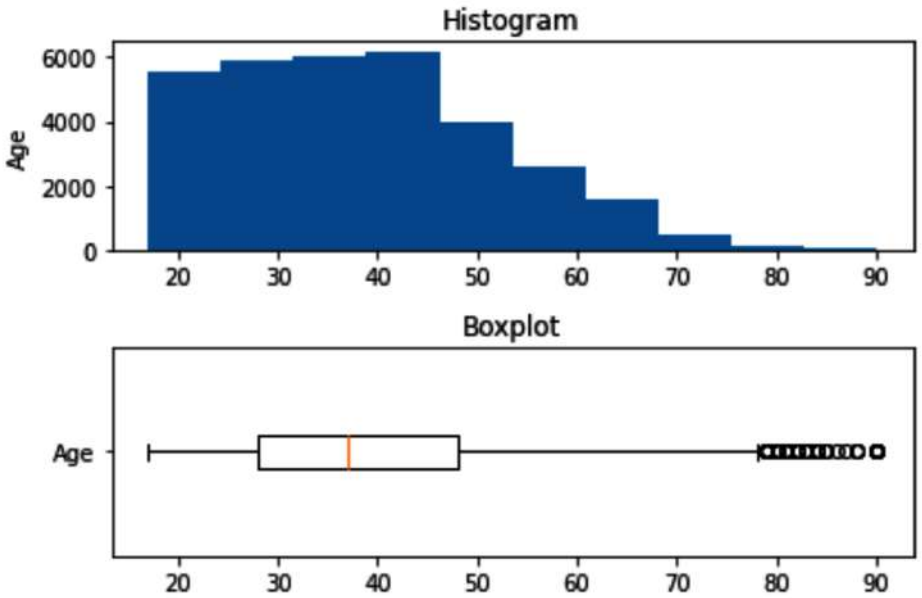
In [12]:

```
plt.scatter(apl_df.Close,amz_df.Close, marker = 'x', color='green')
plt.title('Amazon and Apple stock prices in 2000 to 2020')
plt.xlabel('Apple price ($)')
plt.ylabel('Amazon price ($)')
plt.show()
```



In [13]:

```
plt.subplot(2,1,1)
plt.hist(adult_df.age)
plt.title('Histogram')
plt.ylabel('Age')

plt.subplot(2,1,2)
plt.boxplot(adult_df.age, vert=False)
plt.title('Boxplot')
plt.yticks([1],['Age'])

plt.tight_layout()
plt.show()
```



Chapter 3: Data – What Is It Really?

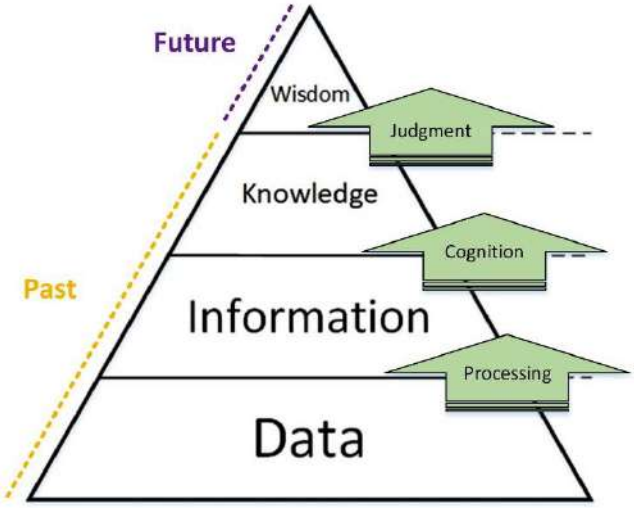
In [1]:
```
import pandas as pd

electricity_df = pd.read_csv('electricity_prediction.csv')
electricity_df.head()
```

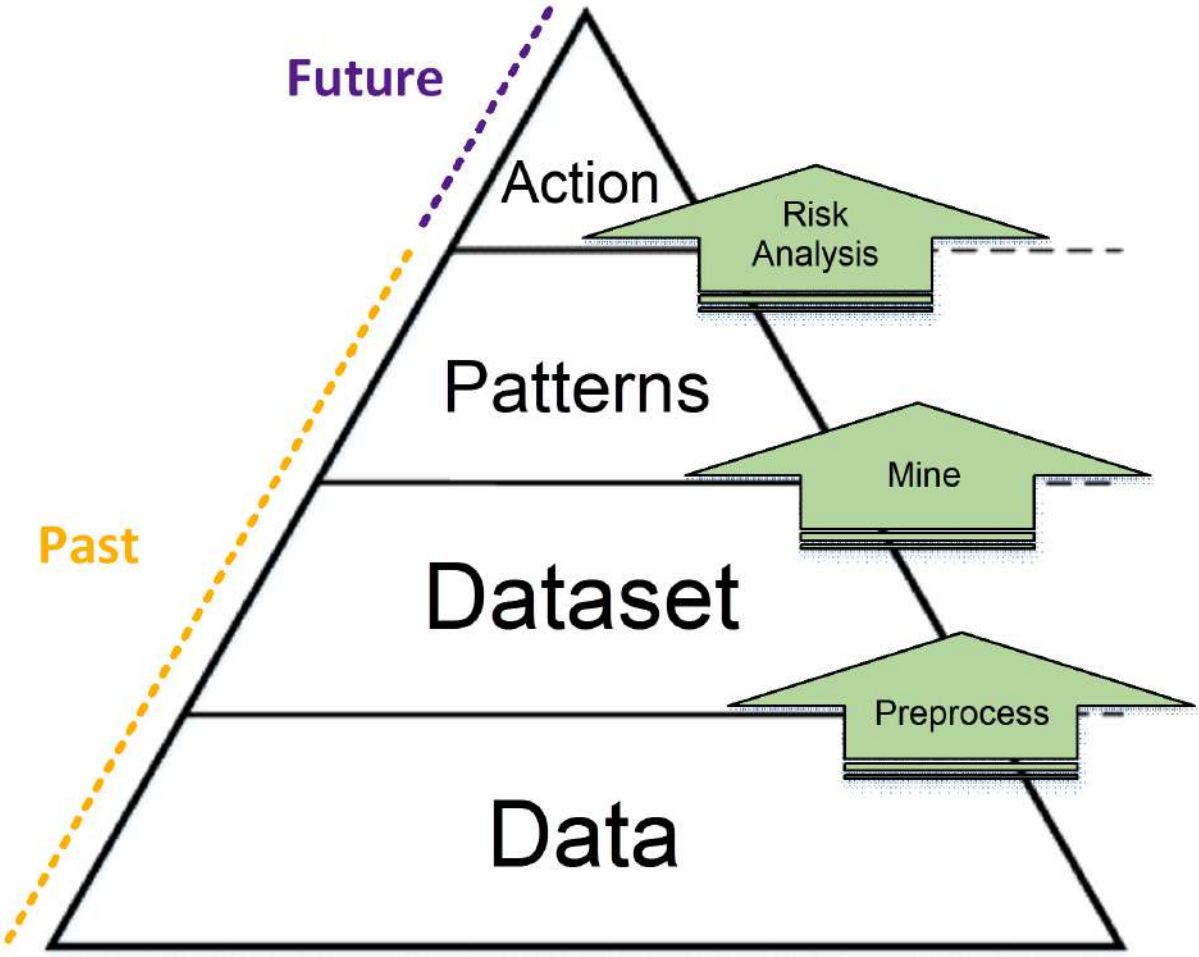
Out[1]:

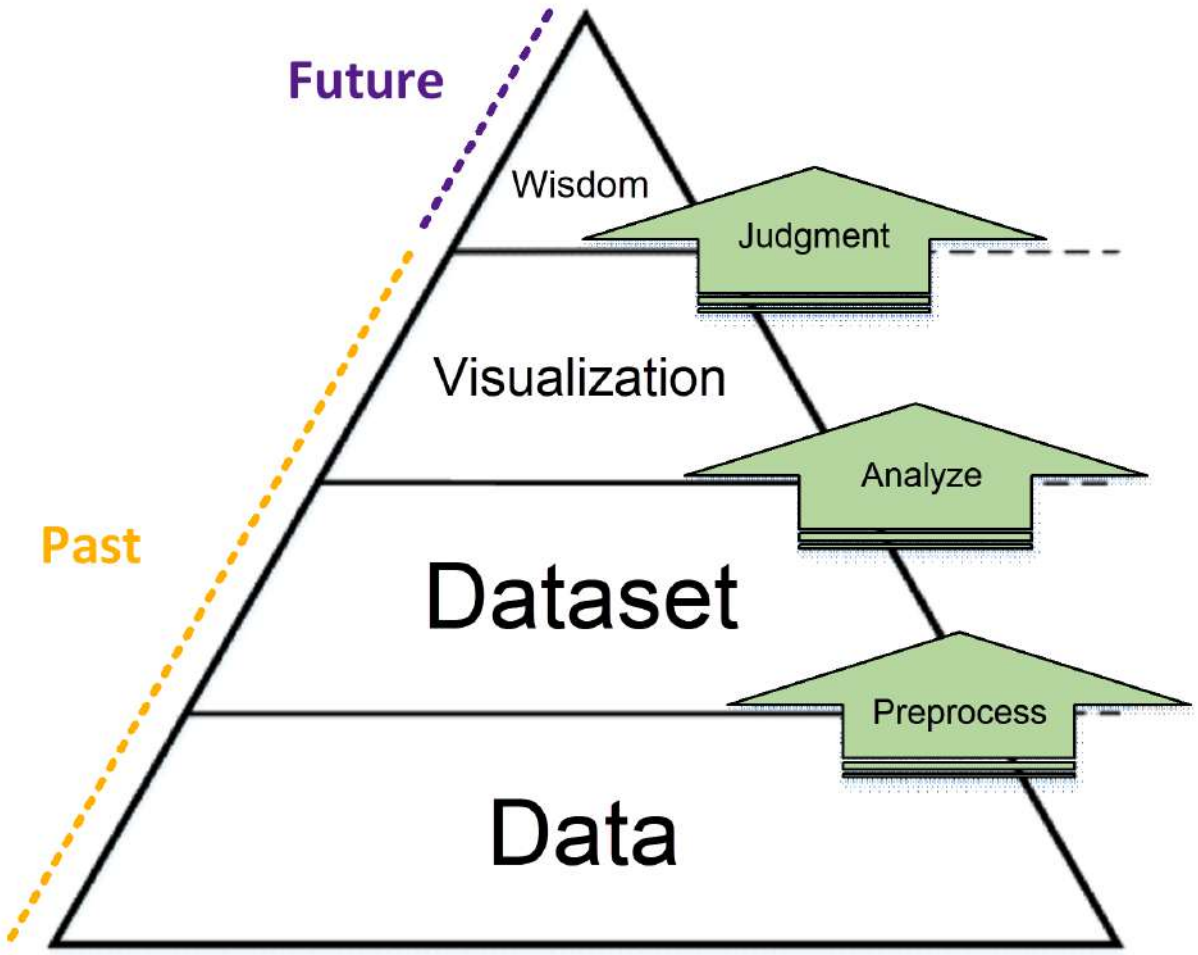
	Date	Weekday	Consumption	Average Temperature
0	1/1/2016	4	2581914	80
1	1/2/2016	5	2663011	77
2	1/3/2016	6	2725351	78
3	1/4/2016	0	3092978	80
4	1/5/2016	1	3231827	81

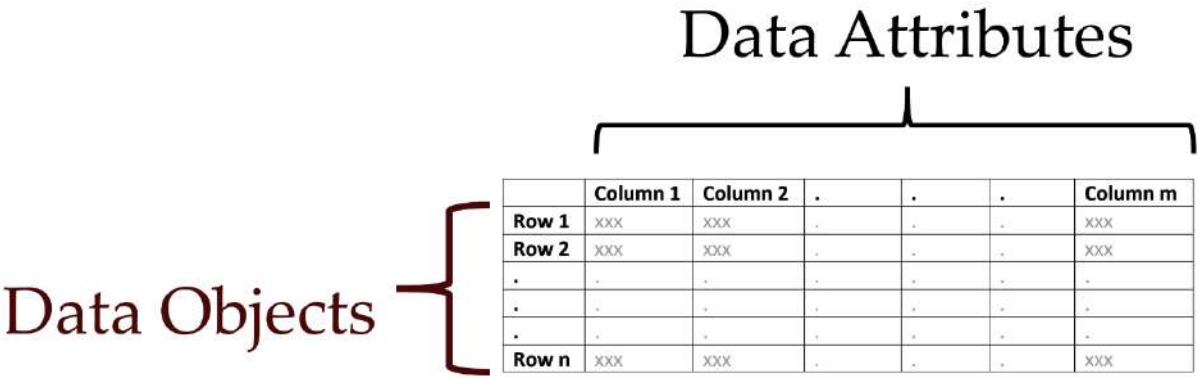
In [2]:
```
from sklearn.linear_model import LinearRegression

X = electricity_df[['Weekday', 'Average Temperature']]
y = electricity_df['Consumption']

lrm = LinearRegression()
lrm.fit(X, y)

print('intercept ', lrm.intercept_)
print(pd.DataFrame({'Predictor': X.columns, 'coefficient': lrm.coef_}))
```

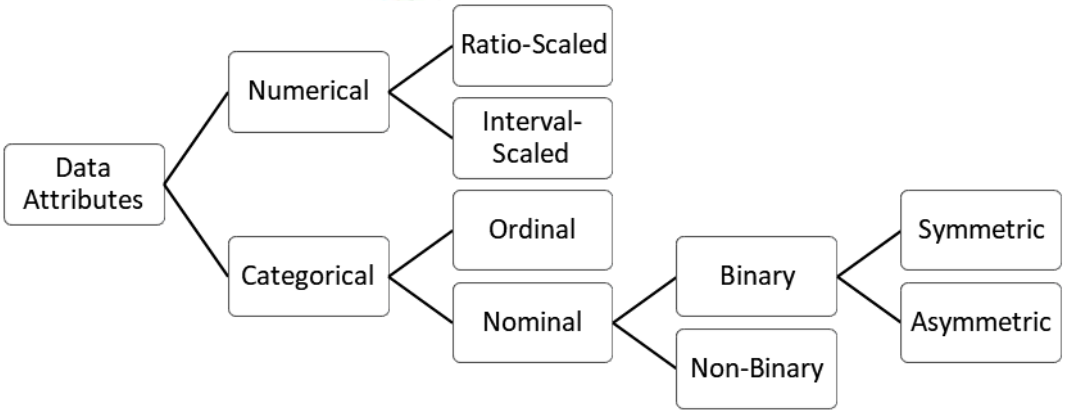
```
intercept  3074181.4950158806
             Predictor   coefficient
0              Weekday -55710.145405
1  Average Temperature  -3476.377056
```



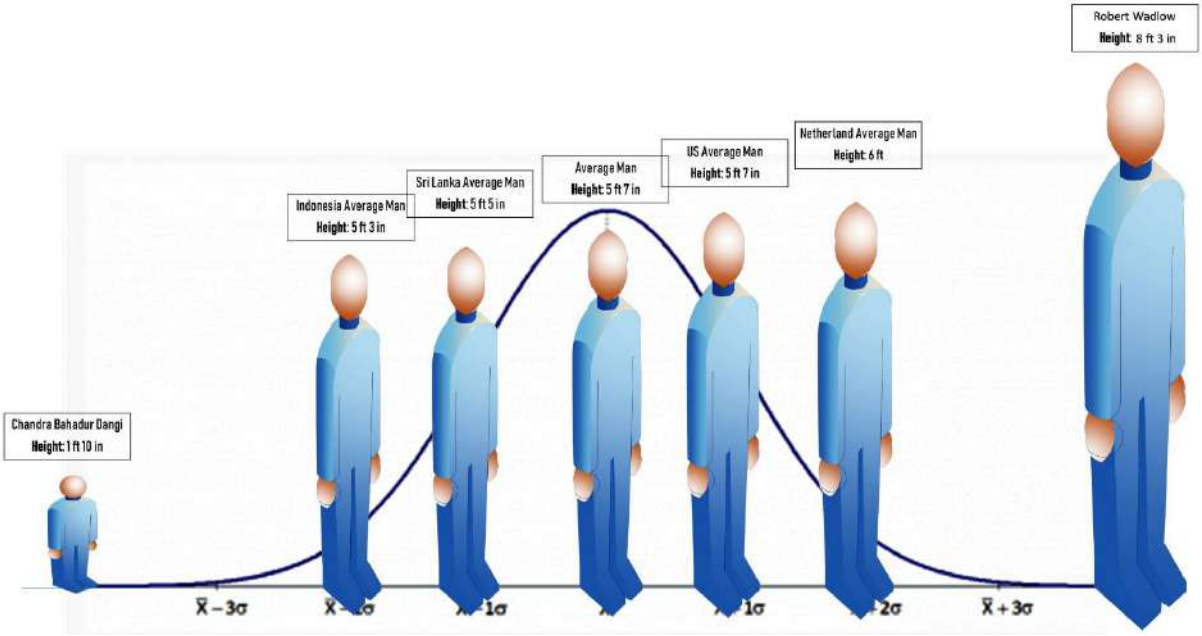

Future
Past
Action
Risk Analysis
Patterns
Mine
Dataset
Preprocess
Data


Future
Past
Wisdom
Judgment
Visualization
Analyze
Dataset
Preprocess
Data


Data Attributes
Data Objects
Column 1
Column 2
.
.
.
Column m
Row 1
Row 2
Row n
xxx


Data Attributes
Numerical
Ratio-Scaled
Interval-Scaled
Categorical
Ordinal
Nominal
Binary
Non-Binary
Symmetric
Asymmetric

Male	M	0	1	1
Female	F	1	0	2



Analytic Perspective		Programming Perspective
Nominal attributes	Binary	Booleans or strings
	Non-binary	
Ordinal attributes		Integers
Interval-scaled attributes		Integers or floating points
Ration-scaled attributes		

In [1]:
```python
import pandas as pd
customer_df = pd.read_excel('Customers Dataset.xlsx')
customer_df
```

Out[1]:

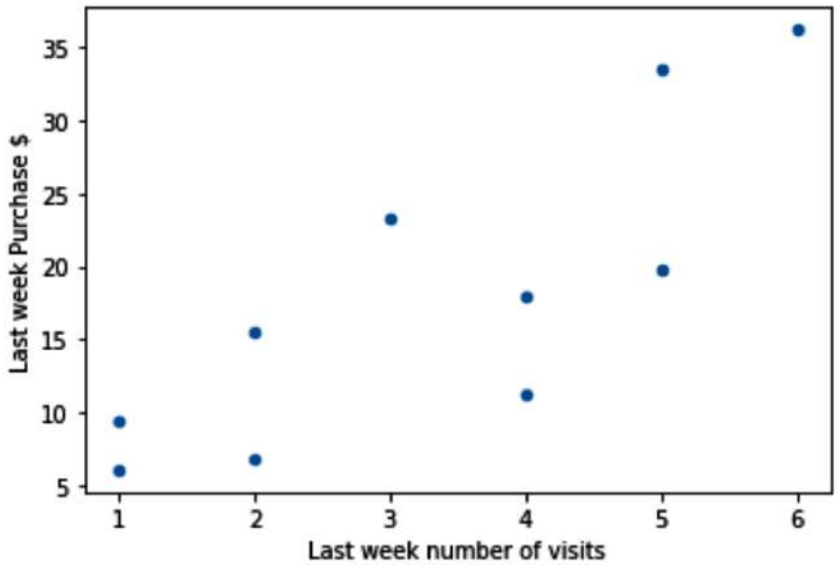
	Customer Name	Store	Last week number of visits	Last week Purchase $
0	Abu Irvine	Starbuck - Claremont Village	5	33.43
1	Colleen Melendez	Starbuck - Claremont Village	4	11.32
2	Lyla-Rose Ruiz	Starbuck - Claremont Village	1	9.48
3	Riley-Jay Manning	Starbuck - Claremont Village	2	15.50
4	Ieuan Carroll	Starbuck - Claremont Village	4	17.96
5	Renesmae Lawson	Starbuck - Claremont Village	5	19.84
6	Lawrence Medina	Starbuck - Claremont Village	3	23.21
7	Ben O'Connor	Starbuck - Claremont Village	1	6.12
8	Adnaan Kim	Starbuck - Claremont Village	6	36.16
9	Abbigail Dunlap	Starbuck - Claremont Village	2	6.88

In [2]:
```python
customer_df.var()
```

Out[2]:
```
Last week number of visits      3.122222
Last week Purchase $          109.628044
dtype: float64
```

In [3]:

```
import matplotlib.pyplot as plt
customer_df.plot.scatter(x='Last week number of visits',
                         y='Last week Purchase $')
plt.show()
```



In [4]:

```
customer_df.corr()
```

Out[4]:

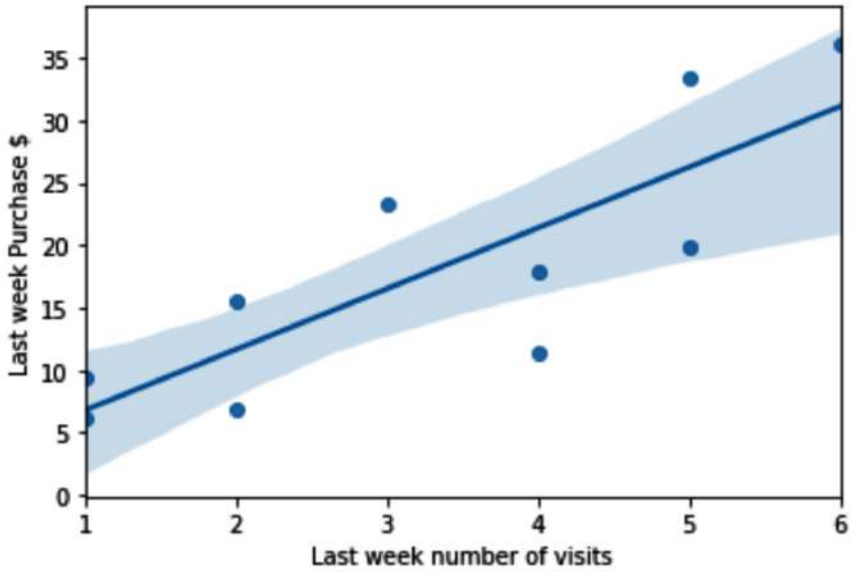
	Last week number of visits	Last week Purchase $
Last week number of visits	1.000000	0.821282
Last week Purchase $	0.821282	1.000000

In [5]:
```
customer2_df = pd.read_excel('Customers Dataset 2.xlsx')
customer2_df
```

Out[5]:

	Customer Name	Store	Last week number of visits	Last week Purchase $
0	Nelson Rivera	Starbuck - Claremont Village	3	NaN
1	Abbigail Felix	Starbuck - Claremont Village	1	NaN
2	Kelly North	Starbuck - Claremont Village	2	NaN
3	Aneesa Moran	Starbuck - Claremont Village	5	NaN
4	Ammara Ritter	Starbuck - Claremont Village	7	NaN
5	Elise Valenzuela	Starbuck - Claremont Village	1	NaN
6	Jaidan Gay	Starbuck - Claremont Village	4	NaN
7	Alejandro Mercer	Starbuck - Claremont Village	3	NaN
8	Arisha Whittaker	Starbuck - Claremont Village	5	NaN
9	Mehmet Power	Starbuck - Claremont Village	2	NaN

In [6]:
```
import seaborn as sns
sns.regplot(x='Last week number of visits',
            y='Last week Purchase $',data=customer_df)
plt.show()
```



In [7]:

```
customer2_df['Last week Purchase $'] =  customer2_df[
    'Last week number of visits'].apply(lambda v:1.930 + 4.867 *v)
customer2_df
```

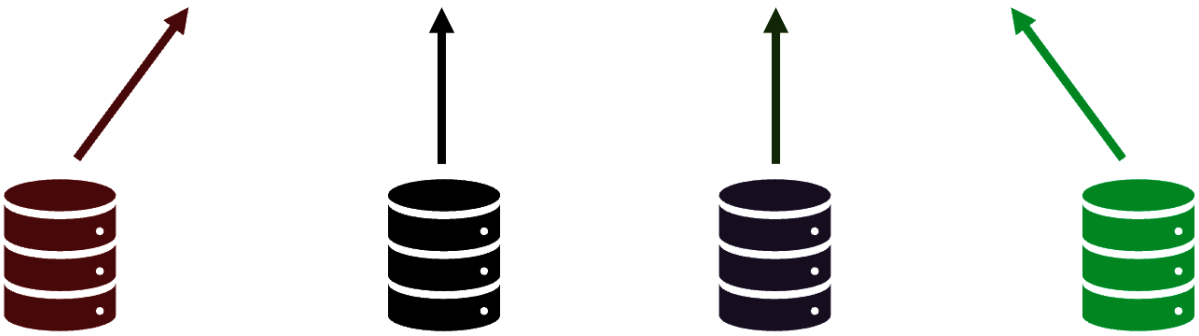
Out[7]:

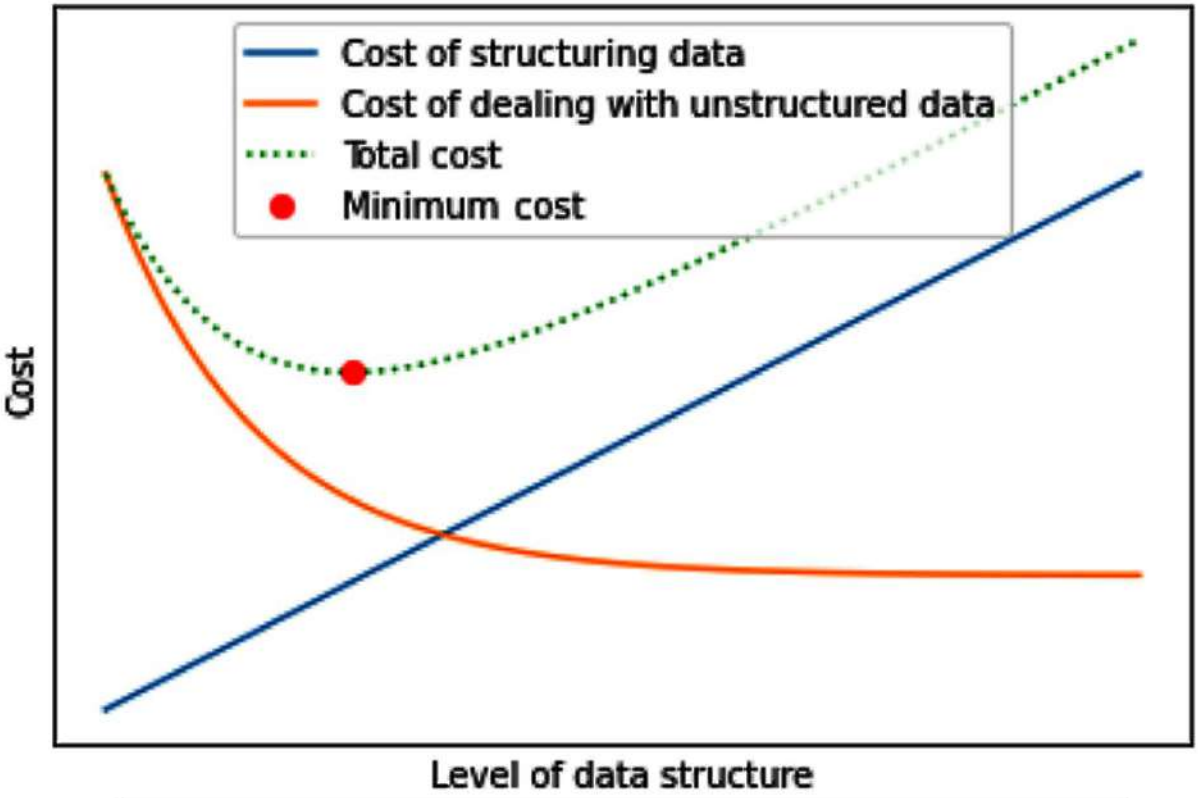
	Customer Name	Store	Last week number of visits	Last week Purchase $
0	Nelson Rivera	Starbuck - Claremont Village	3	16.531
1	Abbigail Felix	Starbuck - Claremont Village	1	6.797
2	Kelly North	Starbuck - Claremont Village	2	11.664
3	Aneesa Moran	Starbuck - Claremont Village	5	26.265
4	Ammara Ritter	Starbuck - Claremont Village	7	35.999
5	Elise Valenzuela	Starbuck - Claremont Village	1	6.797
6	Jaidan Gay	Starbuck - Claremont Village	4	21.398
7	Alejandro Mercer	Starbuck - Claremont Village	3	16.531
8	Arisha Whittaker	Starbuck - Claremont Village	5	26.265
9	Mehmet Power	Starbuck - Claremont Village	2	11.664

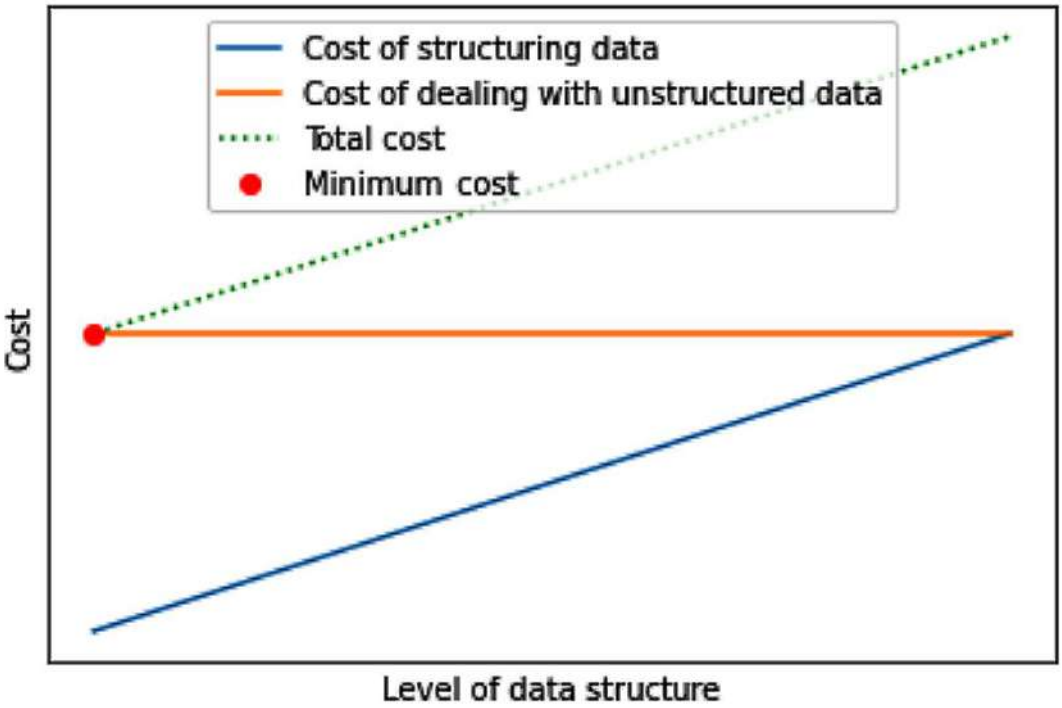
Chapter 4: Databases

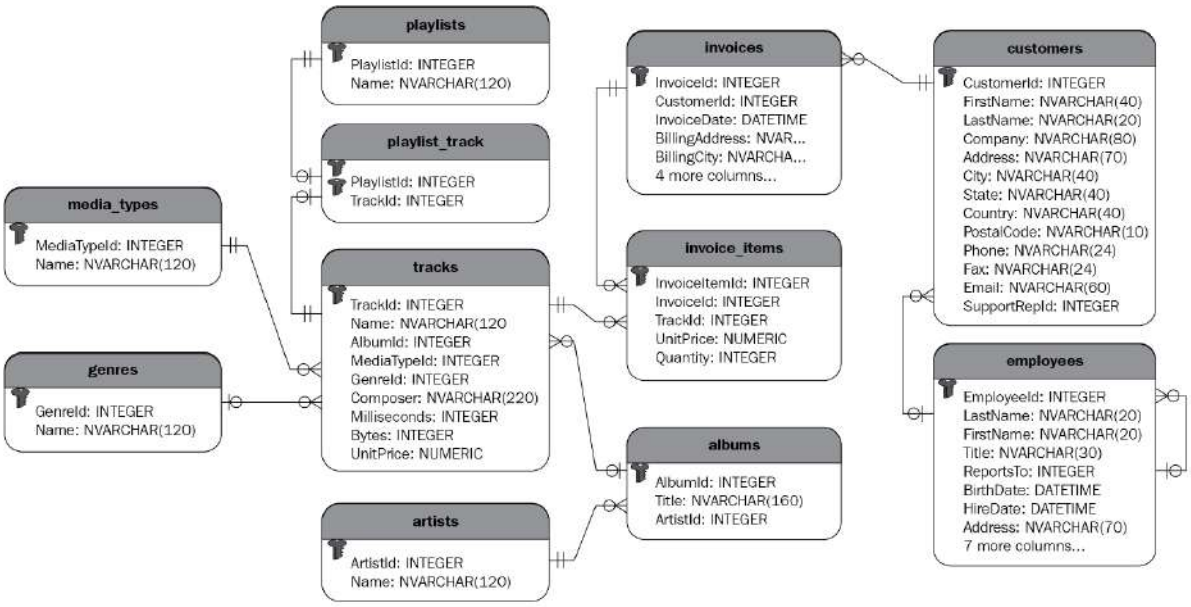
MusicTitleTyple	TotalSale
Negative	1.110000
Neither	1.174550
Positive	1.188667

	Column 1	Column 2	.	.	.	Column m
Row 1	xxx	xxx	.	.	.	xxx
Row 2	xxx	xxx	.	.	.	xxx
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Row n	xxx	xxx	.	.	.	xxx




Cost of structuring data
Cost of dealing with unstructured data
Total cost
Minimum cost
Cost
Level of data structure


Cost of structuring data
Cost of dealing with unstructured data
Total cost
Minimum cost
Cost
Level of data structure



In [1]:

```
import sqlite3
import pandas as pd
```

In [2]:

```
Connection = sqlite3.connect('chinook.db')
query_txt = "SELECT * FROM customers;"

df_customers = pd.read_sql_query(query_txt,Connection)
```

In [4]:

```
print(API_address)
```

```
https://finnhub.io/api/v1/stock/candle?symbol=AMZN&resolution=W&from=157786
5600&to=1609315200&token=bsiqli7rh5rc8orbnkqg
```

In [7]:

```
from datetime import datetime
import requests
import pandas

stk_ticker = 'AMZN'
data_resolution = 'W'
timestamp_from = 1577865600
timestamp_to = 1609315200
API_Key = 'bsiqli7rh5rc8orbnkqg'
Address_template = 'https://finnhub.io/api/v1/stock/candle?symbol={}&resolut

API_address = Address_template.format(stk_ticker,data_resolution,
                                      timestamp_from,timestamp_to,API_Key)

AMZN_df = pd.DataFrame(r.json())
AMZN_df.drop(columns=['s'],inplace=True)
AMZN_df.t = AMZN_df.t.apply(datetime.fromtimestamp)
AMZN_df.t = AMZN_df.t.apply(lambda v:v.date())
AMZN_df.set_index('t',drop=True,inplace=True)
AMZN_df.columns = ['Closing','High','Low','Opening','Volume']
AMZN_df.head()
```

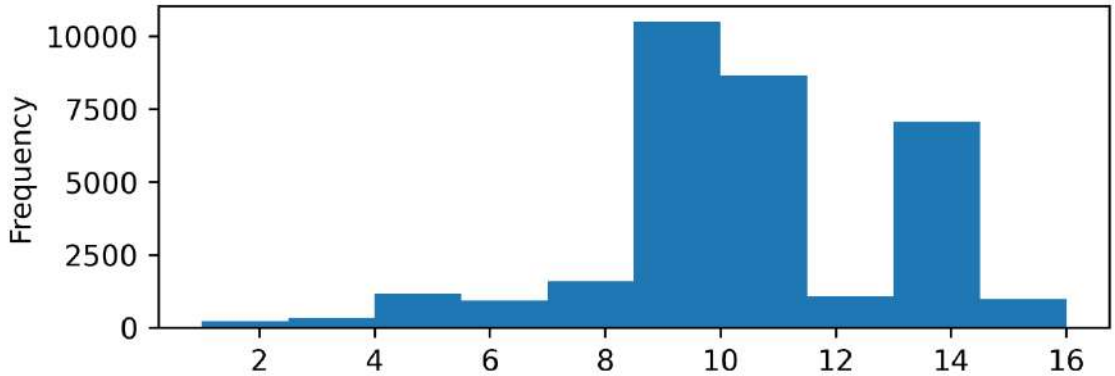
Out[7]:

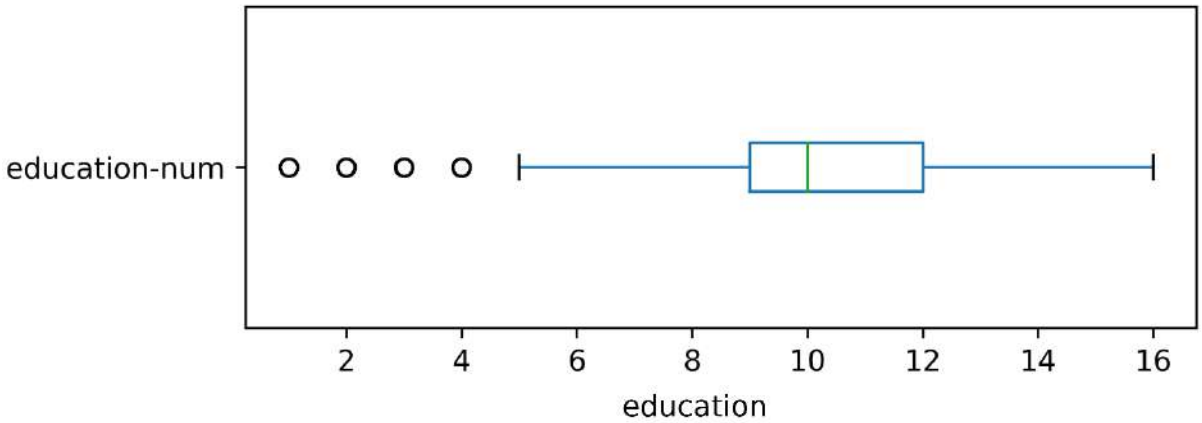
t	Closing	High	Low	Opening	Volume
2020-01-01	1891.97	1913.89	1860.00	1875.00	19514188
2020-01-08	1862.02	1917.82	1855.09	1909.89	15160738
2020-01-15	1887.46	1902.50	1857.25	1882.99	13580875
2020-01-22	1858.00	1894.99	1815.34	1885.11	14688733
2020-01-29	2039.87	2071.02	1850.61	1858.00	37459327

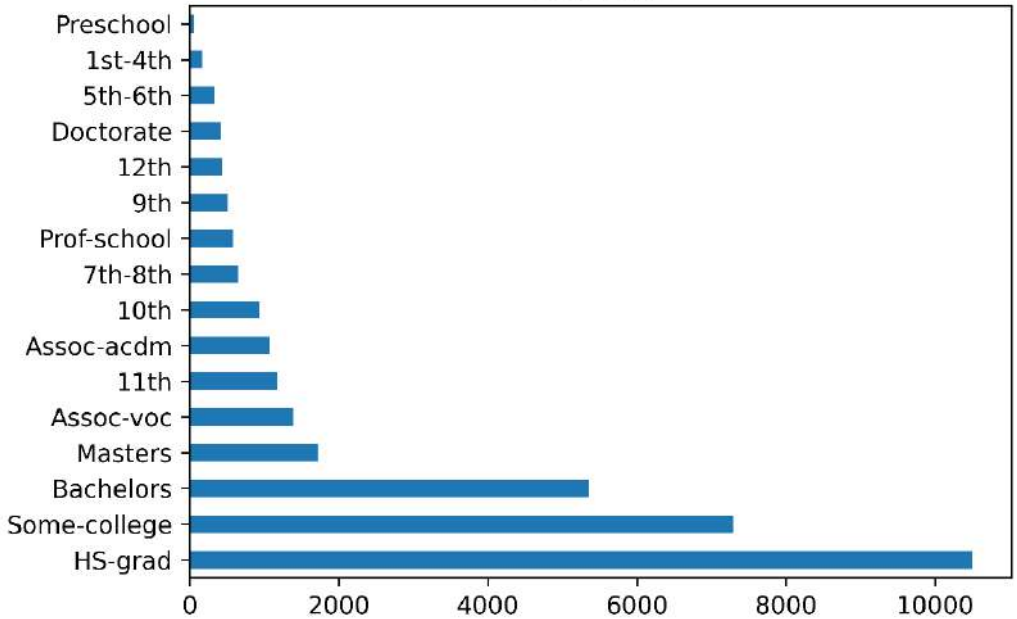
	Relational Databases	Unstructured Databases	Distributed databases	Blockchain
Ease of loading new data to the database	2	1	1 or 2	N/A
Ease of introducing new types of data to the database				
Data access reliability	2	2	1	N/A
Decentralized authority				
Ease of creating more detailed reports				
Ease of recording larger sizes of data				
Better manageability of operational costs				
Prone to political considerations	N/A	N/A	1	2
The need for more preprocessing for analytic purposes				

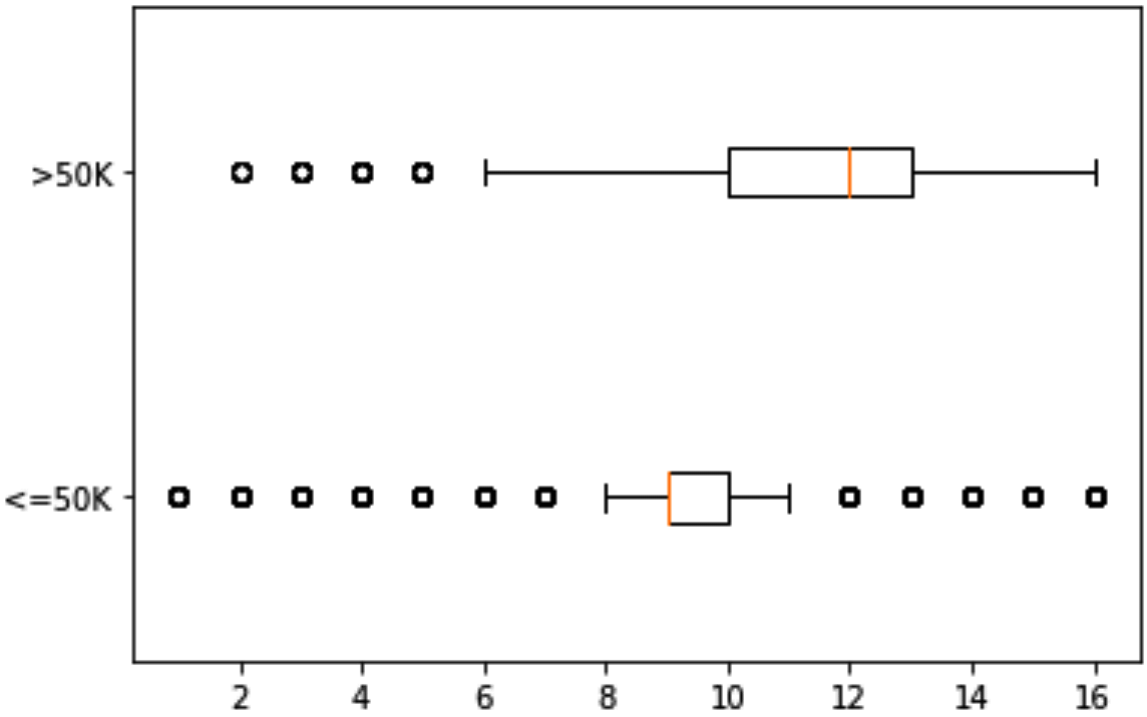
	Direct Connection	Web page Connection	API Connection	Request Connection	Publicly Shared
Flexibility of access	1	2	2	4	5
Prone to human miscommunications	5	5	5	1	5
Need for a high level of technical skillset	1	3	2	4	5
Need for knowing the database tables	1	5	5	5	5
Fastest access to the desired data	1	2	2	5	4
More code-friendly	1	5	2	5	5
Awareness of the possibilities in the database	1	3	3	2	5
Least time-consuming data pulling	1	3	2	5	4
Highest database security	4	2	2	3	1

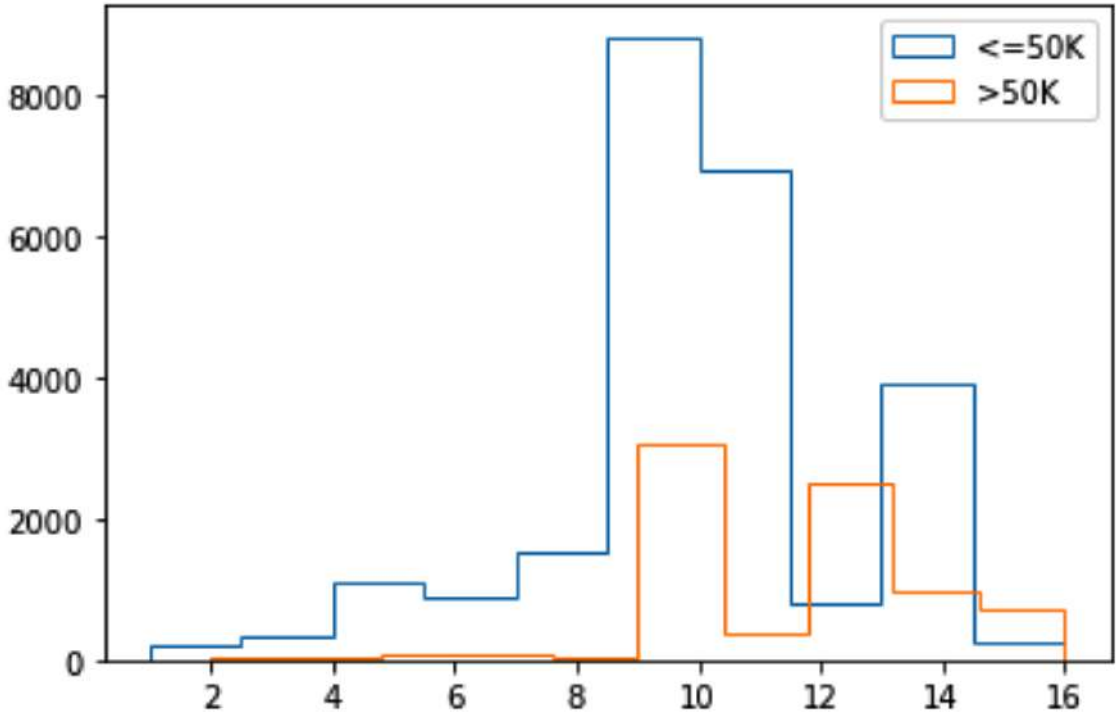
Chapter 5: Data Visualization

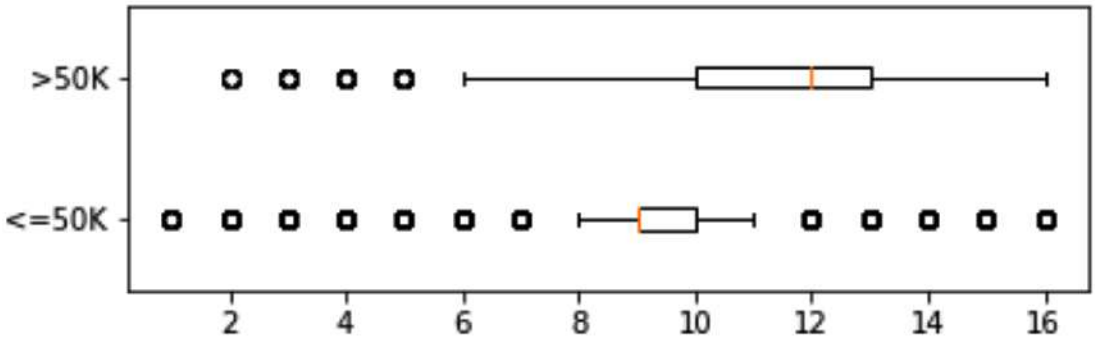


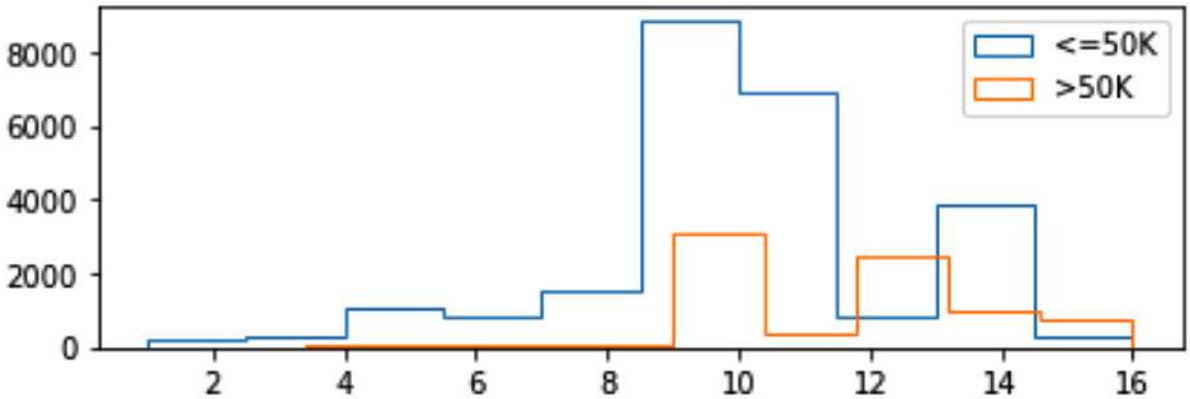





>50K
<=50K
2
4
6
8
10
12
14
16

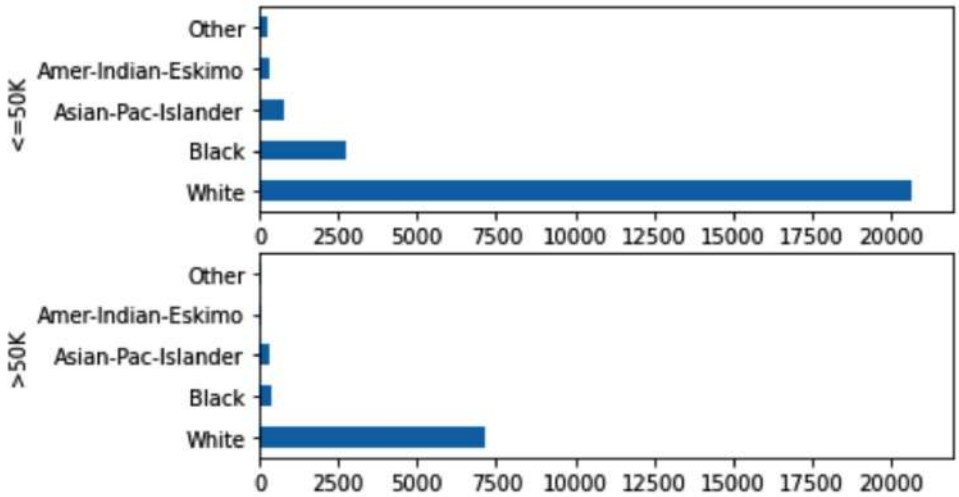

<=50K
>50K
8000
6000
4000
2000
0
2
4
6
8
10
12
14
16



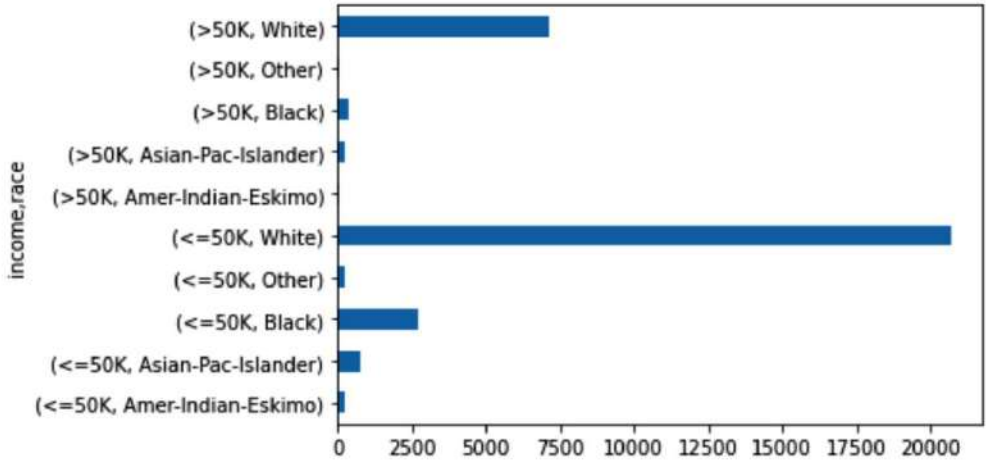


In [8]:

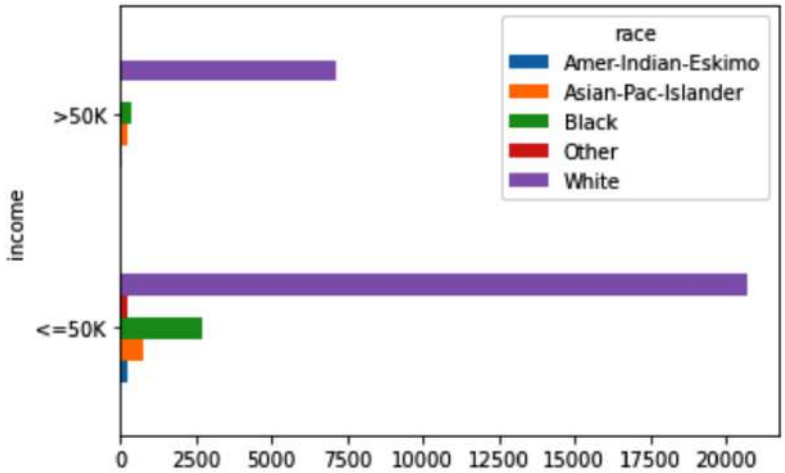
```
income_possibilities = adult_df.income.unique()

for i,poss in enumerate(income_possibilities):
    plt.subplot(2,1,i+1)
    BM = adult_df.income == poss
    adult_df[BM].race.value_counts().plot.barh()
    plt.xlim([0,22000])
    plt.ylabel(poss)
```



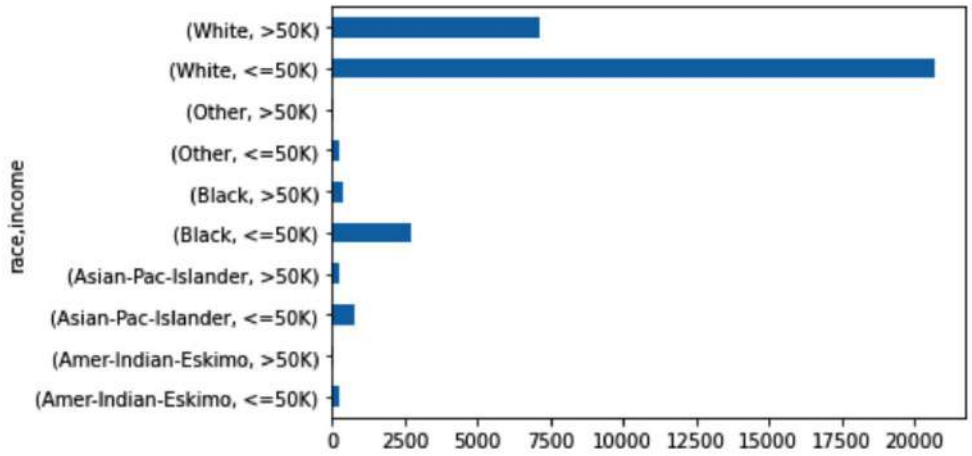
In [9]:
```
adult_df.groupby(['income','race']).size().plot.barh()
plt.show()
```



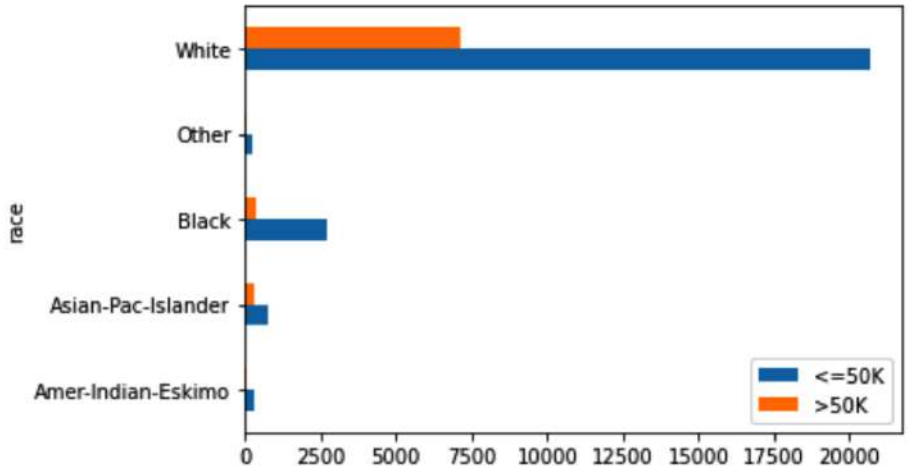
In [10]:
```
adult_df.groupby(['income','race']).size().unstack().plot.barh()
plt.show()
```



In [11]:
```
adult_df.groupby(['race','income']).size().plot.barh()
plt.show()
```

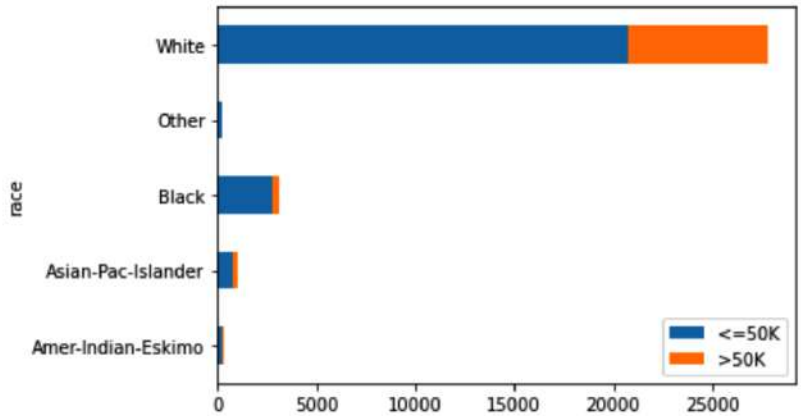


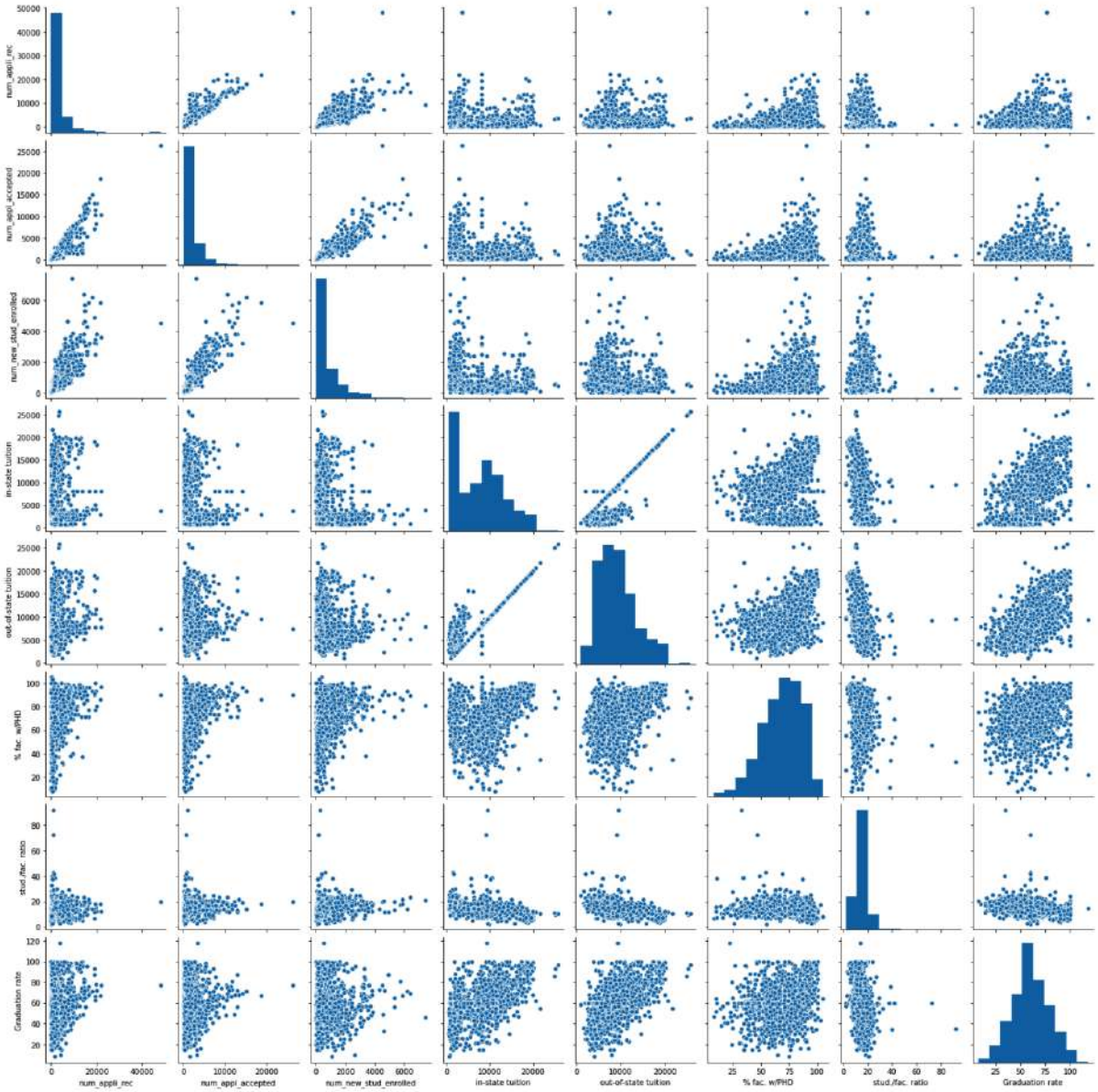
In [12]:
```
adult_df.groupby(['race','income']).size().unstack().plot.barh()
plt.legend(loc=4)
plt.show()
```



In [13]:
```
adult_df.groupby(['race','income']).size().unstack().plot.barh(stacked=True)
plt.legend(loc=4)
```

Out[13]: <matplotlib.legend.Legend at 0x2ac034b4970>





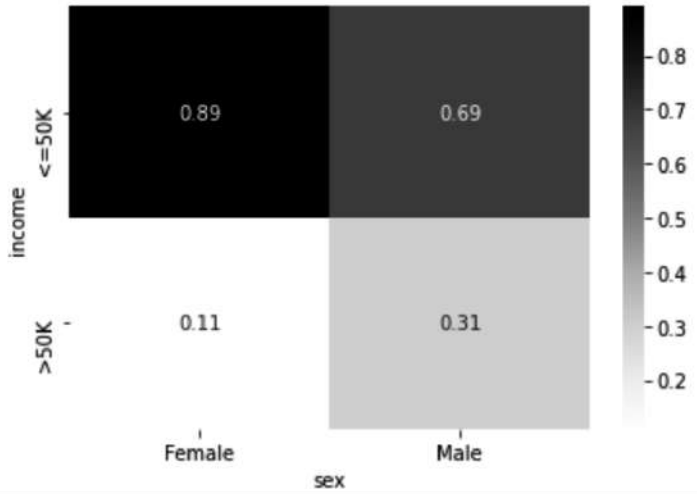
In [15]:
```
contingency_tbl = pd.crosstab(adult_df.income,adult_df.sex)
contingency_tbl
```

Out[15]:

sex	Female	Male
income		
<=50K	9592	15128
>50K	1179	6662

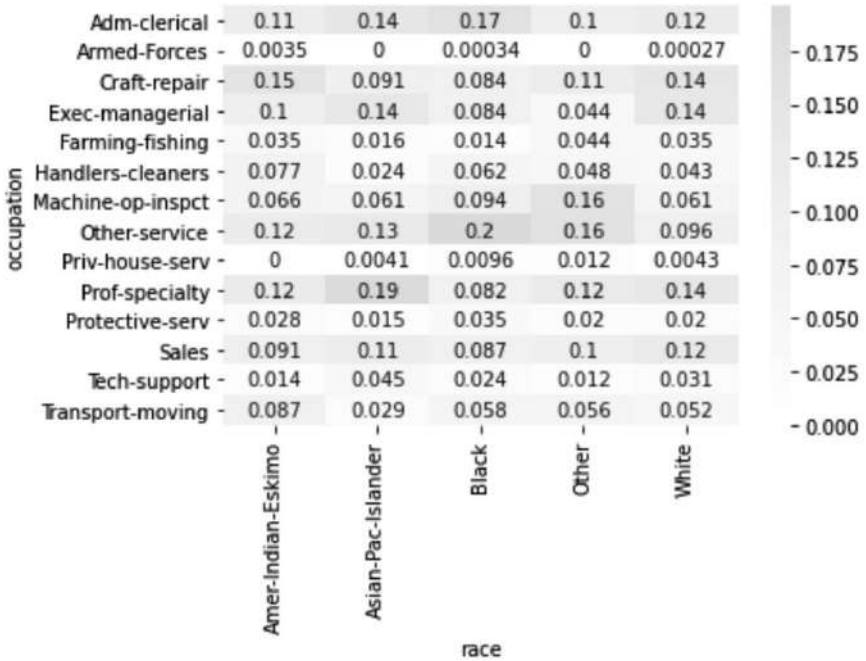
In [16]:

```
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



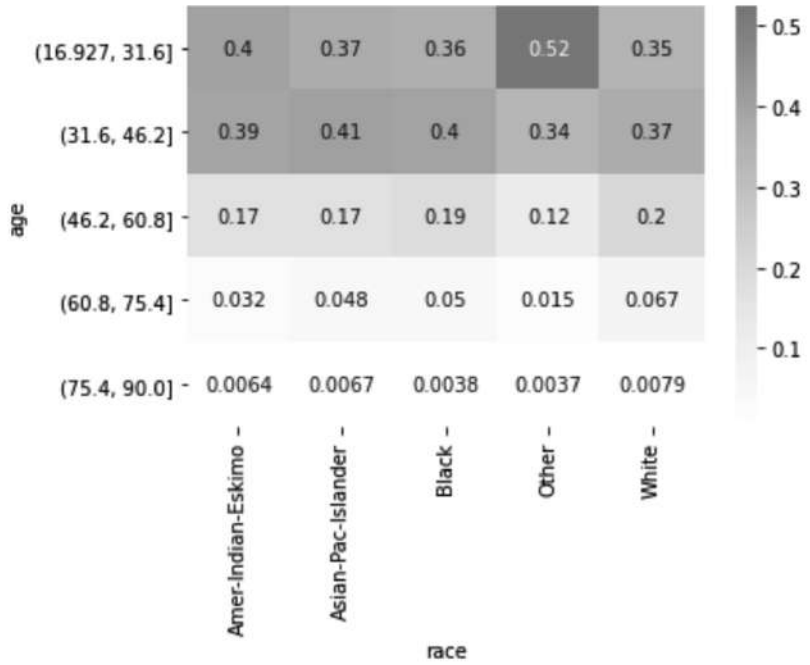
In [17]:

```
contingency_tbl = pd.crosstab(adult_df.occupation,adult_df.race)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



In [18]:

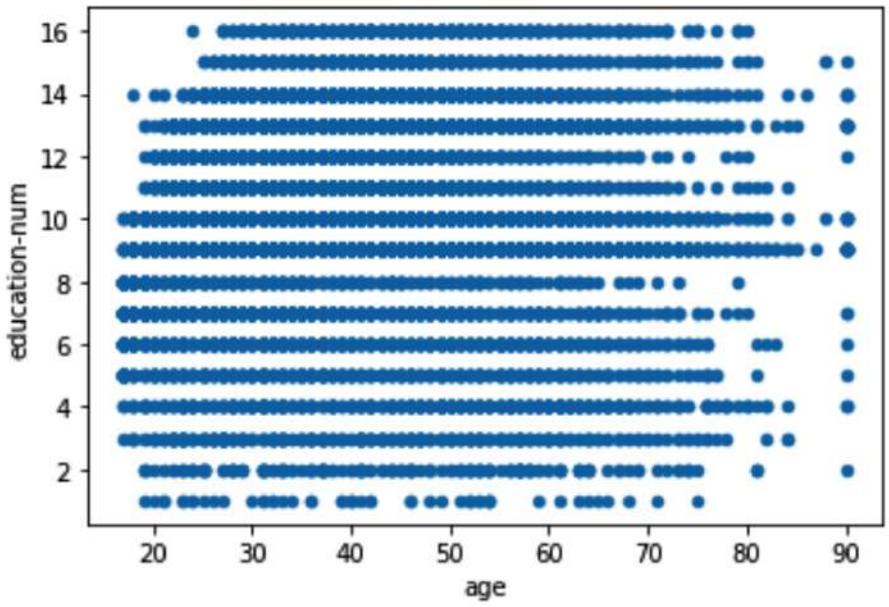
```
age_discretized = pd.cut(adult_df.age, bins = 5)
contingency_tbl = pd.crosstab(age_discretized,adult_df.race)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



education	Preschool	1st-4th	5th-6th	7th-8th	9th	10th	11th	12th	HS-grad	Some-college	Assoc-voc	Assoc-acdm	Bachelors	Masters	Prof-school	Doctorate
education-num	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16

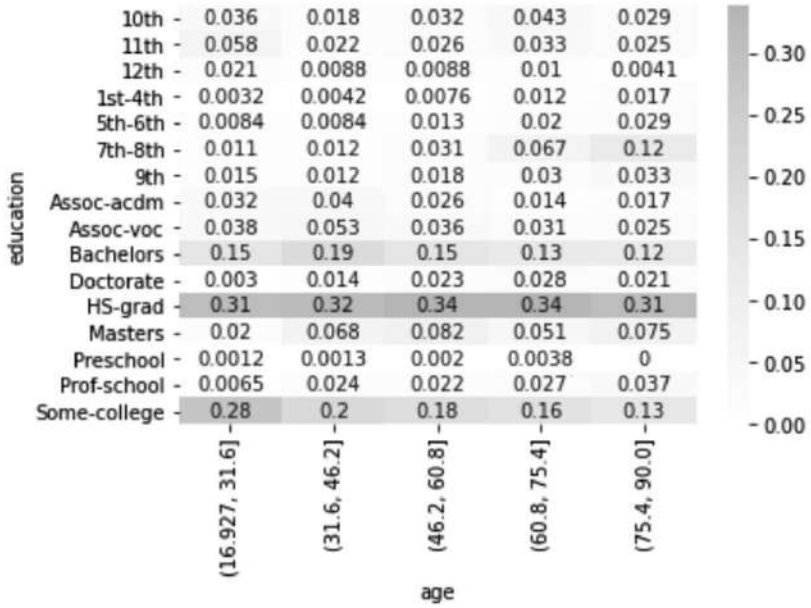
In [21]:

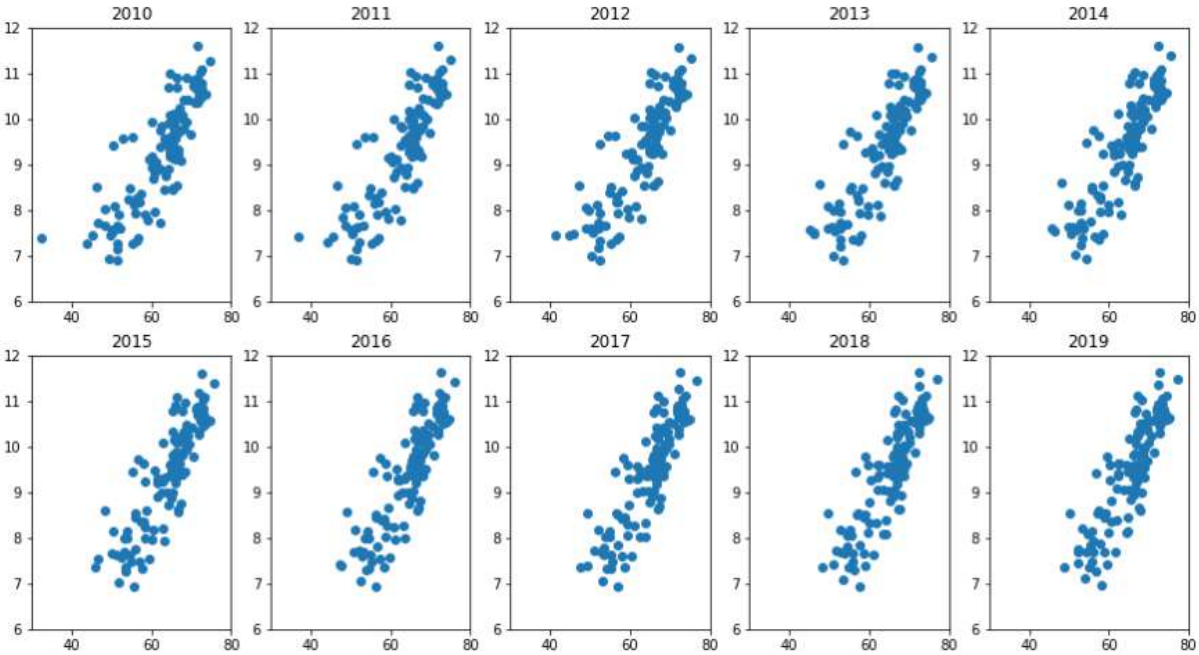
```
adult_df.plot.scatter(x='age',y='education-num')
plt.show()
```

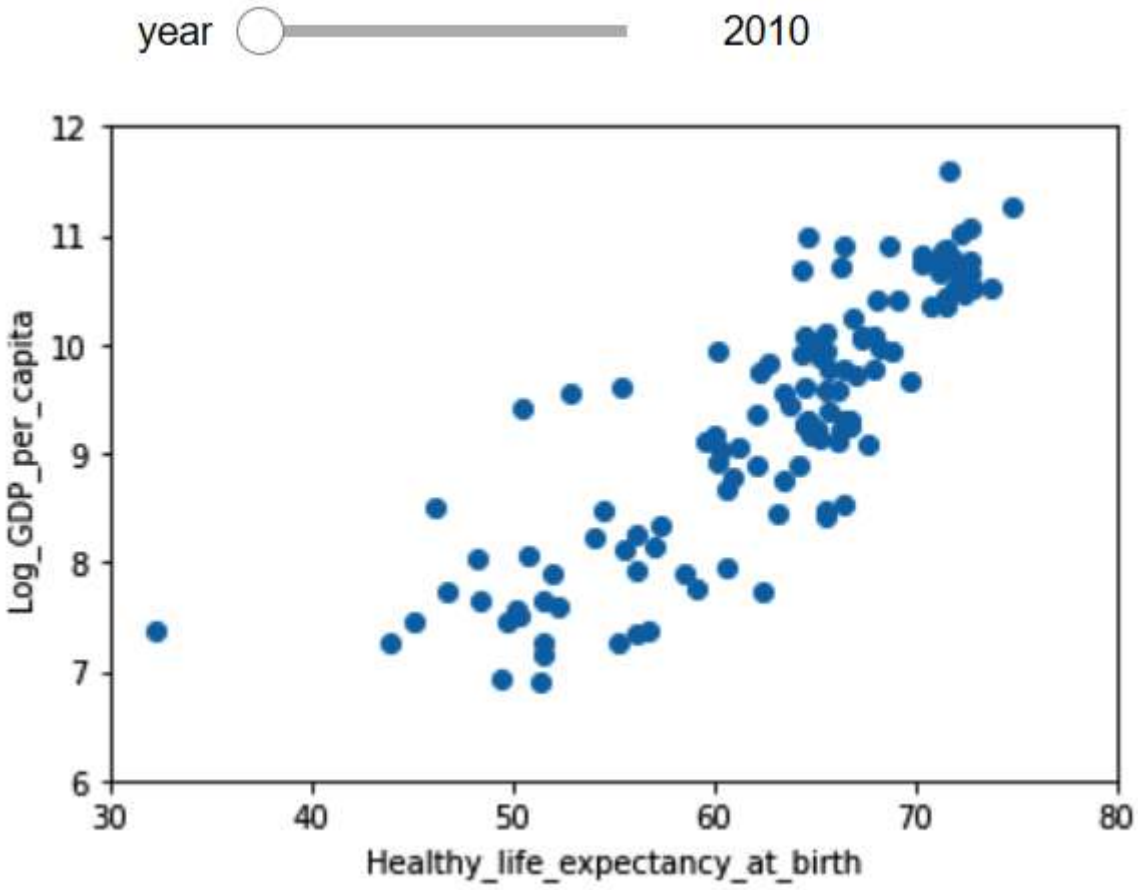


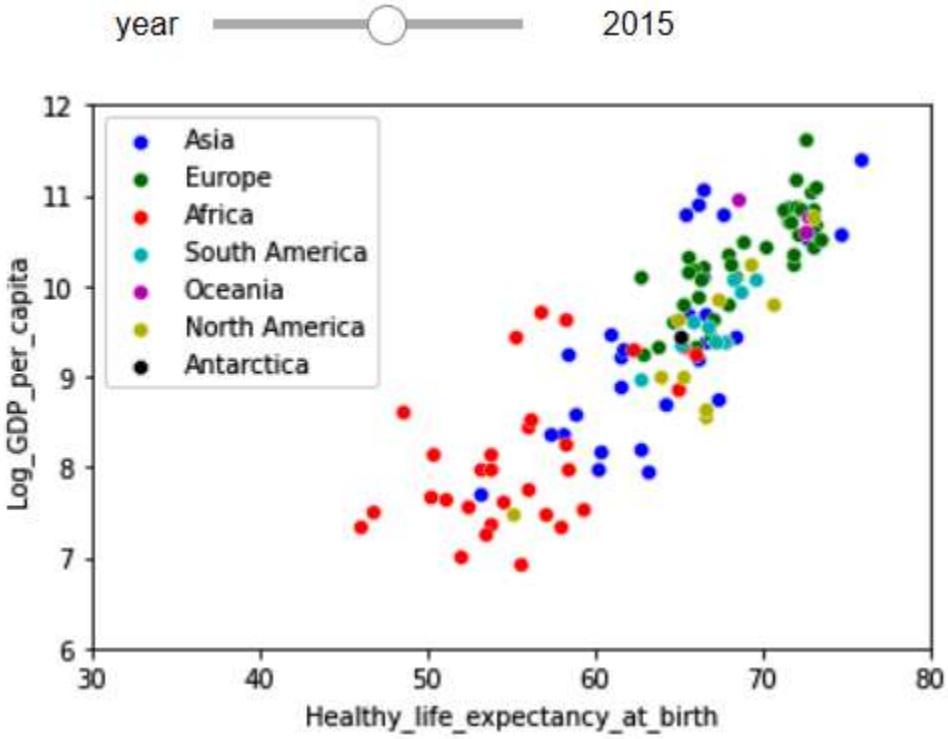
In [22]:

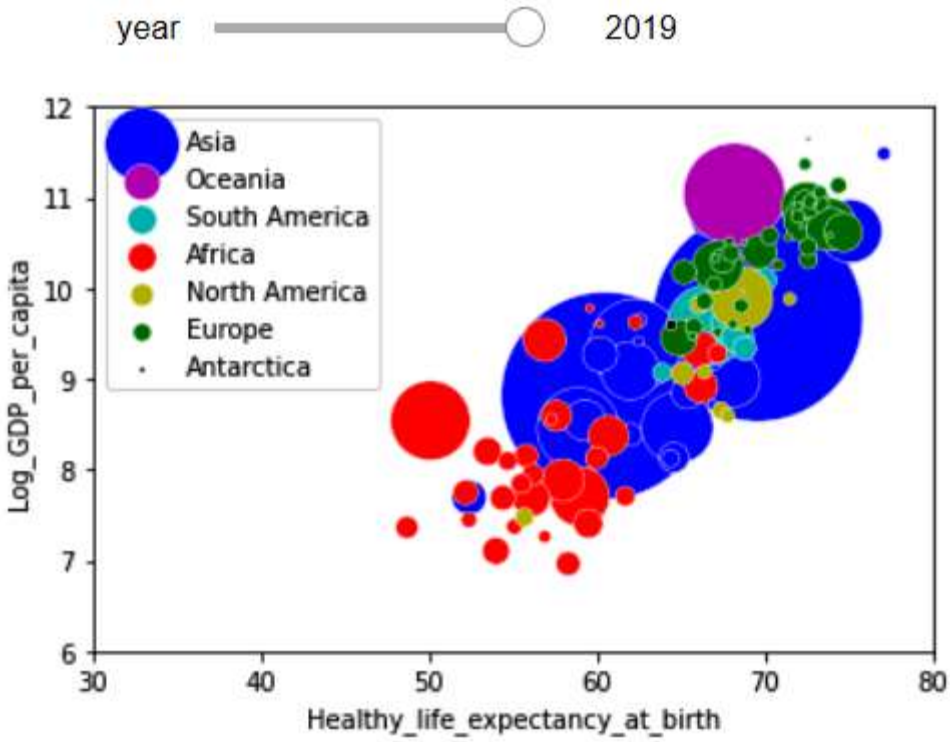
```
age_discretized = pd.cut(adult_df['age'], bins = 5)
contingency_tbl = pd.crosstab(adult_df.education,age_discretized)
probablity_tbl = contingency_tbl/ contingency_tbl.sum()
sns.heatmap(probablity_tbl, annot=True, center=0.5 ,cmap="Greys")
plt.show()
```



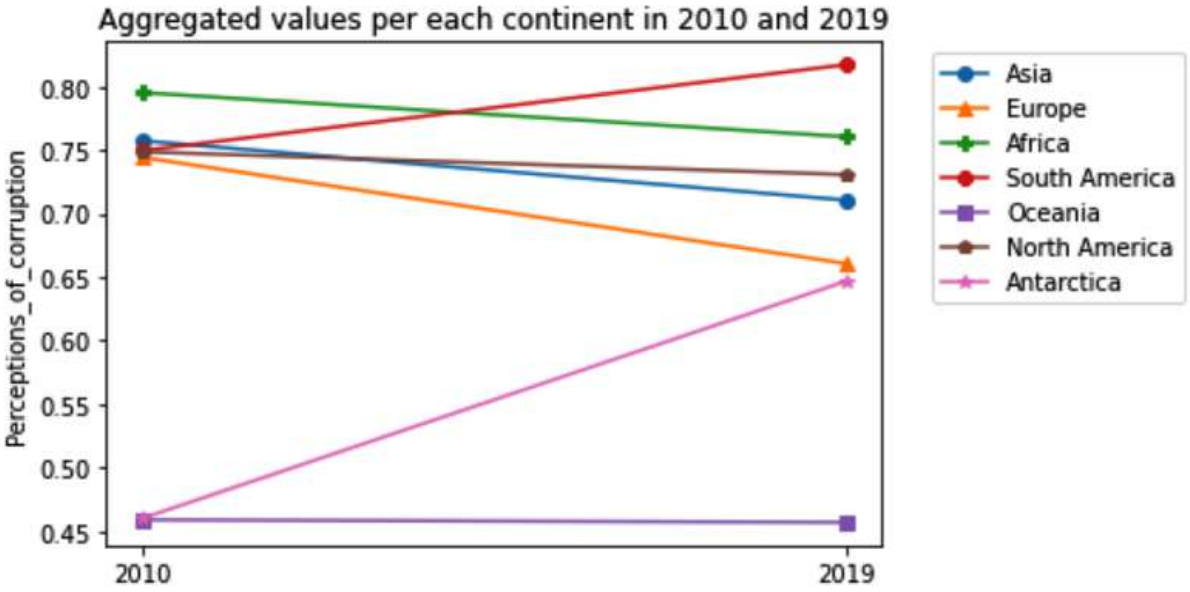


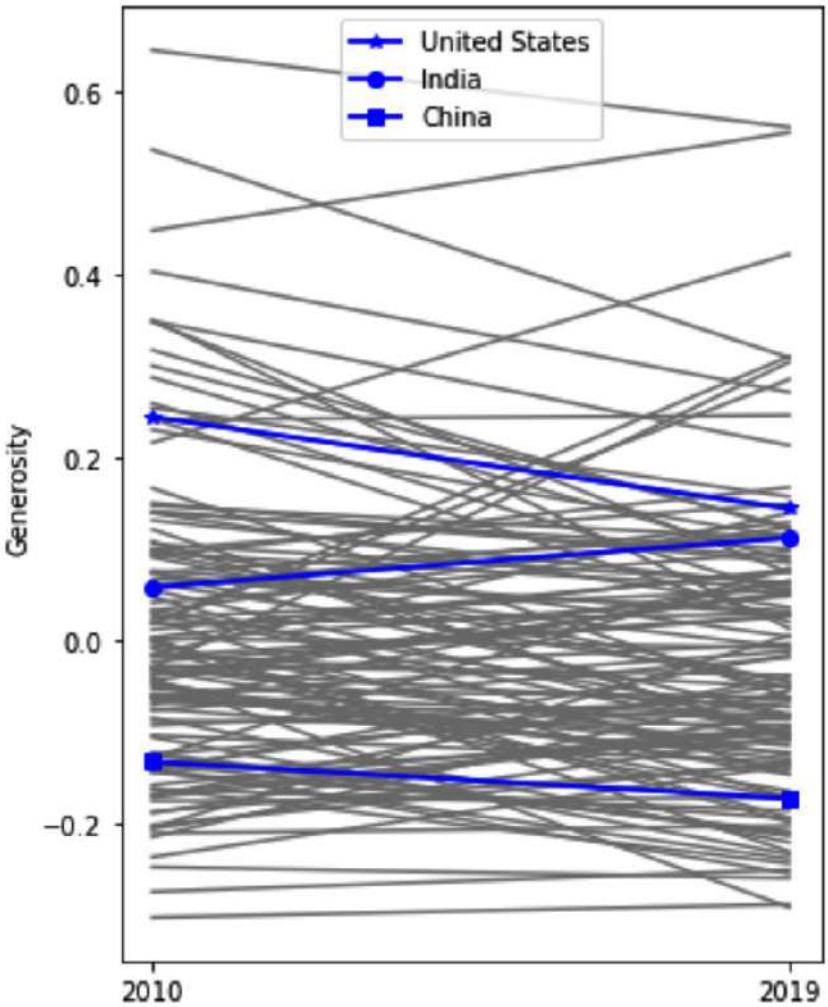

year
2010
Log_GDP_per_capita
Healthy_life_expectancy_at_birth


year
2015
Asia
Europe
Africa
South America
Oceania
North America
Antarctica
Log_GDP_per_capita
Healthy_life_expectancy_at_birth

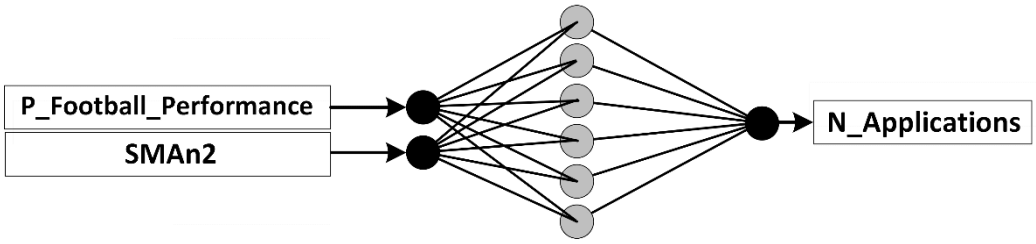


Date	1/2/2020	1/3/2020	1/6/2020	1/7/2020	1/8/2020	1/9/2020	1/10/2020	1/13/2020	1/14/2020	1/15/2020
Amazon	1898.01	1874.97	1902.88	1906.86	1891.97	1901.05	1883.16	1891.3	1869.44	1862.02
Apple	74.3335	73.6108	74.1974	73.8484	75.0364	76.6302	76.8035	78.4443	77.3851	77.0534

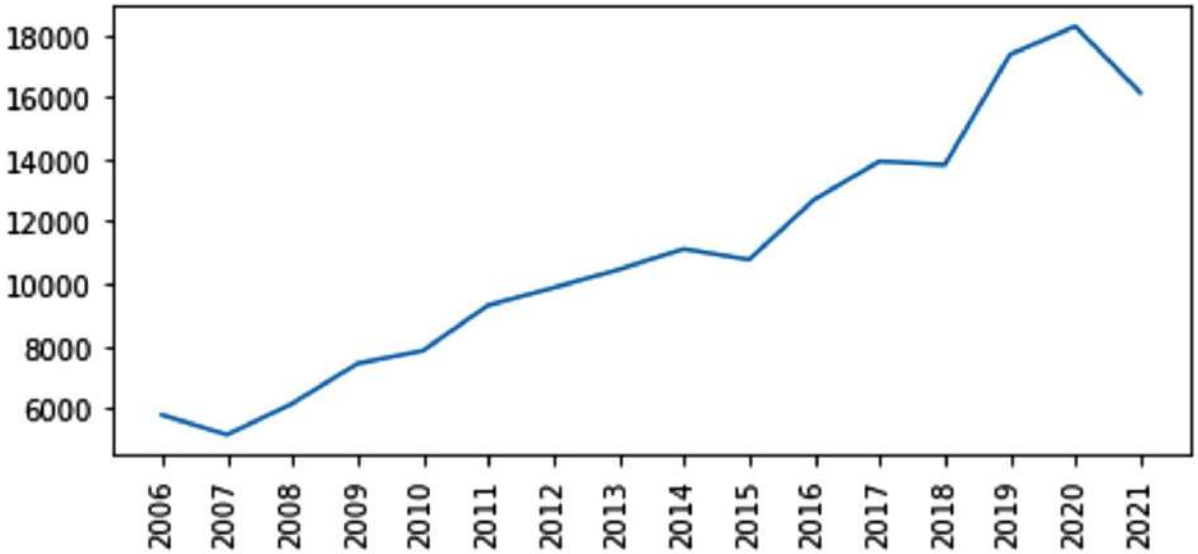

Aggregated values per each continent in 2010 and 2019
Perceptions_of_corruption
0.80
0.75
0.70
0.65
0.60
0.55
0.50
0.45
2010
2019
Asia
Europe
Africa
South America
Oceania
North America
Antarctica

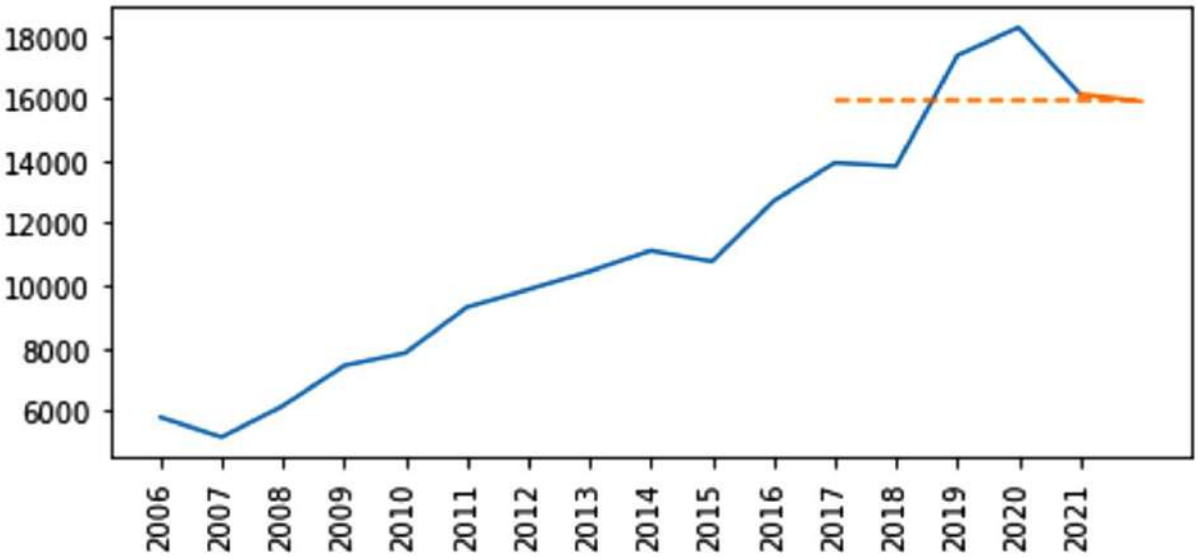

United States
India
China
Generosity
0.6
0.4
0.2
0.0
−0.2
2010
2019

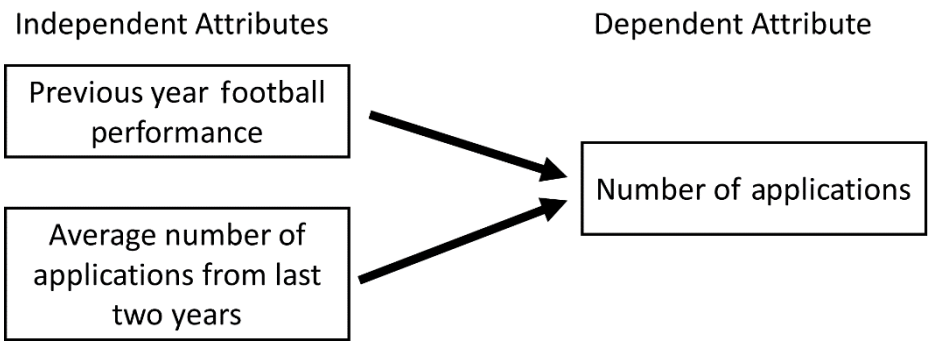
Chapter 6: Prediction



Year	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021
N_Applications	5778	5140	6141	7429	7839	9300	9864	10449	11117	10766	12701	13930	13817	17363	18269	16127







In [1]:

```
import pandas as pd
msu_df = pd.read_csv('MSU applications.csv')
msu_df.set_index('Year',drop=True,inplace=True)
msu_df
```

Out[1]:

	P_Football_Performance	SMAn2	N_Applications
Year			
2006	0.273	5778.0	5778
2007	0.273	5778.0	5140
2008	0.250	5459.0	6141
2009	0.615	5640.5	7429
2010	0.333	6785.0	7839
2011	0.417	7634.0	9300
2012	0.692	8569.5	9864
2013	0.538	9582.0	10449
2014	0.615	10156.5	11117
2015	0.538	10783.0	10766
2016	0.769	10941.5	12701
2017	0.692	11733.5	13930
2018	0.462	13315.5	13817
2019	0.692	13873.5	17363
2020	0.615	15590.0	18269
2021	0.462	17816.0	16127

In [2]:

```
from sklearn.linear_model import LinearRegression

X = ['P_Football_Performance','SMAn2']
y = 'N_Applications'

data_X = msu_df[X]
data_y = msu_df[y]

lm = LinearRegression()
lm.fit(data_X, data_y)

print('intercept (b0) ', lm.intercept_)
coef_names = ['b1','b2']
print(pd.DataFrame({'Predictor': data_X.columns,
                    'coefficient Name':coef_names,
                    'coefficient Value': lm.coef_}))
```

```
intercept (b0)  -890.7106225983407
                Predictor coefficient Name  coefficient Value
0  P_Football_Performance               b1        5544.961933
1                   SMAn2               b2           0.907032
```

In [3]:

```
newData = pd.DataFrame({'P_Football_Performance':0.364,'SMAn2':17198},
                       index=[2022])
newData
```

Out[3]:

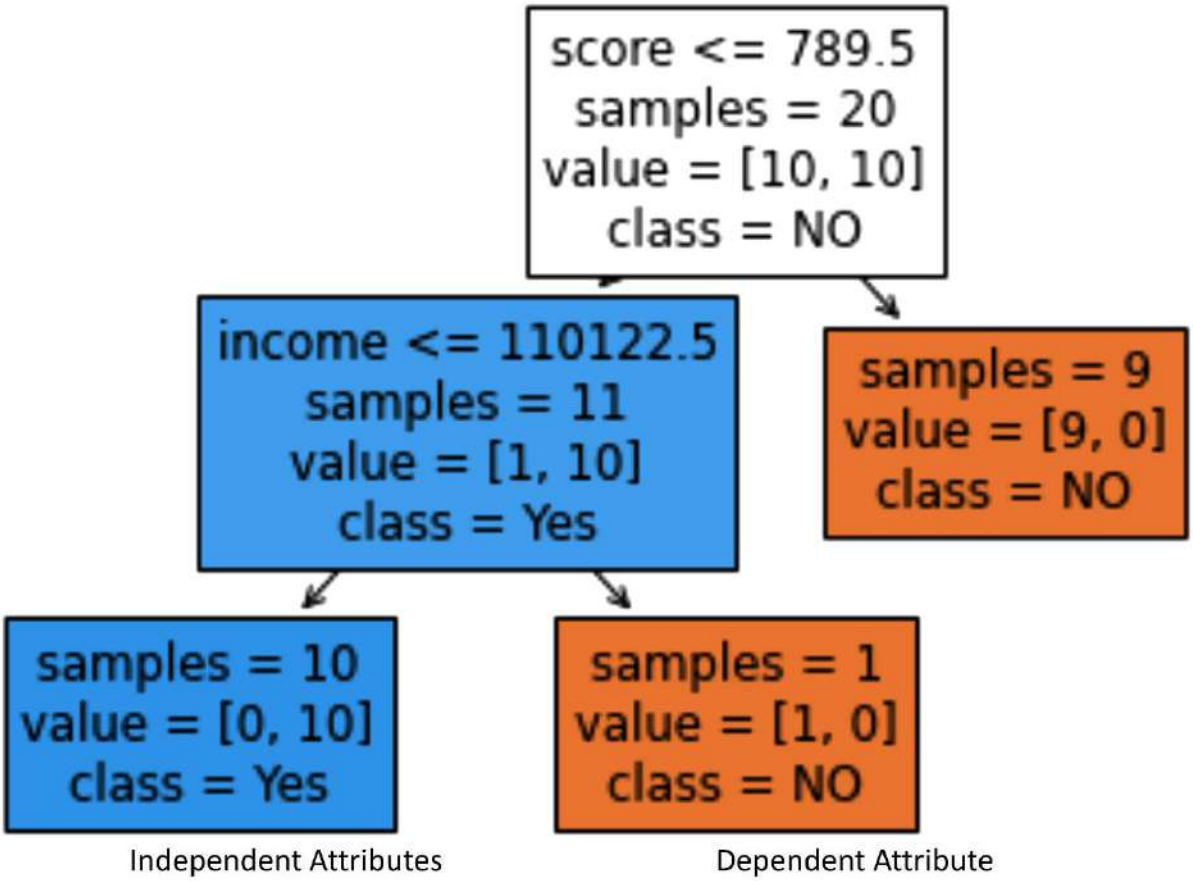
	P_Football_Performance	SMAn2
2022	0.364	17198

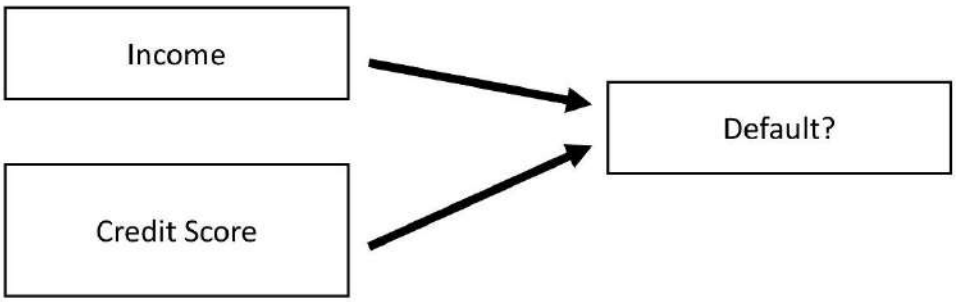
In [4]:

```
lm.predict(newData)
```

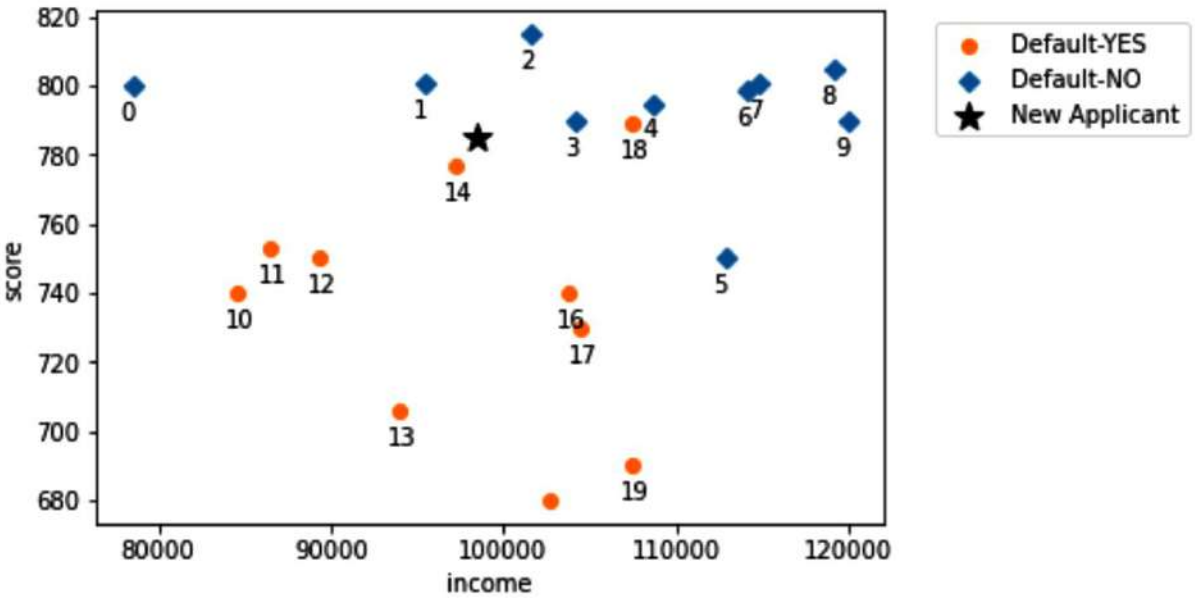
Out[4]: array([16726.78787061])

Chapter 7: Classification





	income	score	default
0	78479	800	NO
1	95483	801	NO
2	101641	815	NO
3	104234	790	NO
4	108726	795	NO
5	112845	750	NO
6	114114	799	NO
7	114799	801	NO
8	119147	805	NO
9	119976	790	NO
10	84519	740	Yes
11	86504	753	Yes
12	89292	750	Yes
13	93941	706	Yes
14	97262	777	Yes
15	102658	680	Yes
16	103760	740	Yes
17	104451	730	Yes
18	107388	789	Yes
19	107400	690	Yes


Default-YES
Default-NO
New Applicant
score
income
820
800
780
760
740
720
700
680
80000
90000
100000
110000
120000
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
16
17
18
19

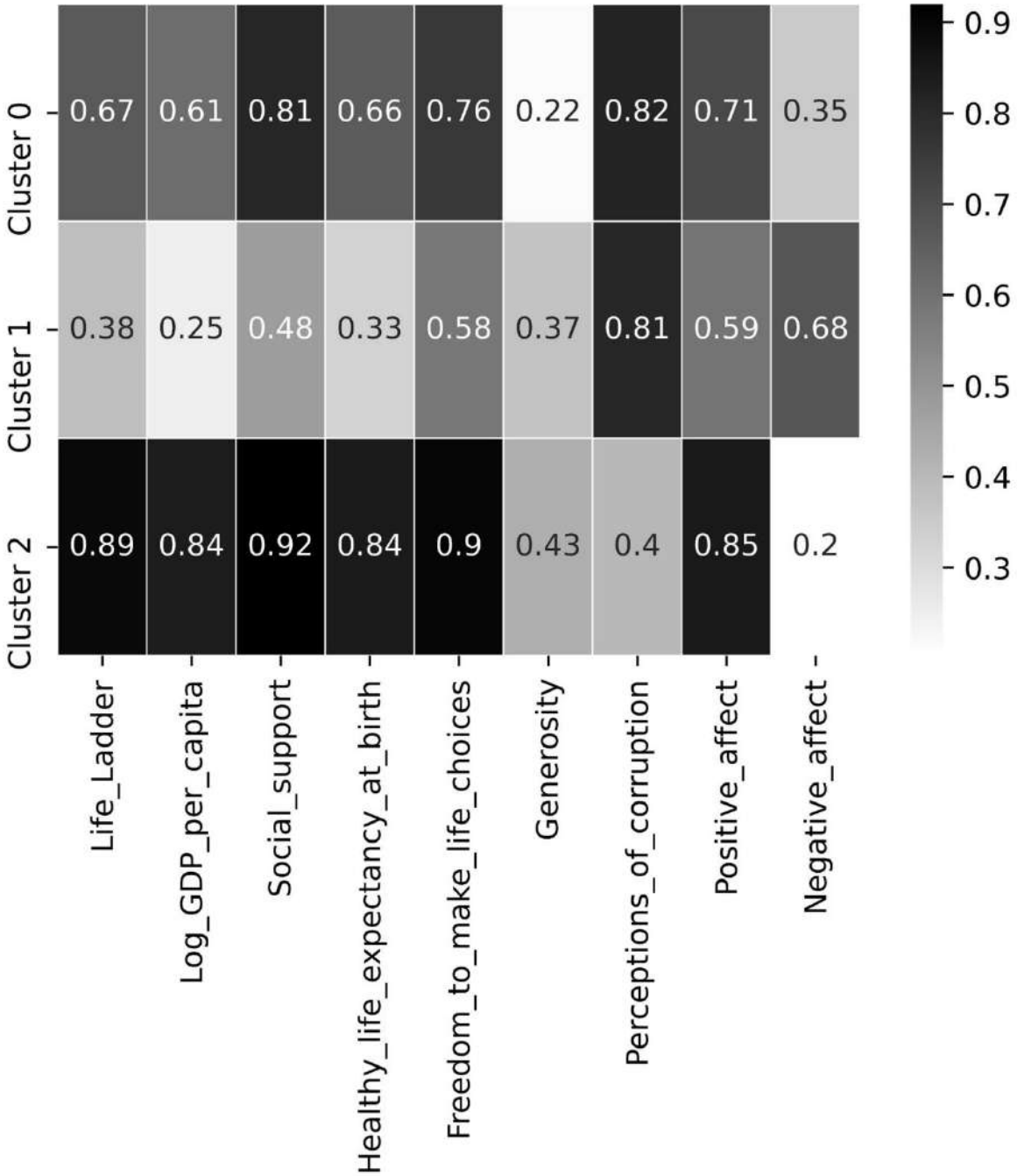
	income	score	default	income_Normalized	score_Normalized
0	78479	800	NO	0.000000	0.888889
1	95483	801	NO	0.409765	0.896296
2	101641	815	NO	0.558161	1.000000
3	104234	790	NO	0.620647	0.814815
4	108726	795	NO	0.728896	0.851852
5	112845	750	NO	0.828156	0.518519
6	114114	799	NO	0.858737	0.881481
7	114799	801	NO	0.875244	0.896296
8	119147	805	NO	0.980023	0.925926
9	119976	790	NO	1.000000	0.814815
10	84519	740	Yes	0.145553	0.444444
11	86504	753	Yes	0.193387	0.540741
12	89292	750	Yes	0.260573	0.518519
13	93941	706	Yes	0.372605	0.192593
14	97262	777	Yes	0.452635	0.718519
15	102658	680	Yes	0.582669	0.000000
16	103760	740	Yes	0.609225	0.444444
17	104451	730	Yes	0.625877	0.370370
18	107388	789	Yes	0.696653	0.807407
19	107400	690	Yes	0.696942	0.074074
20	98487	785	NaN	0.482155	0.777778

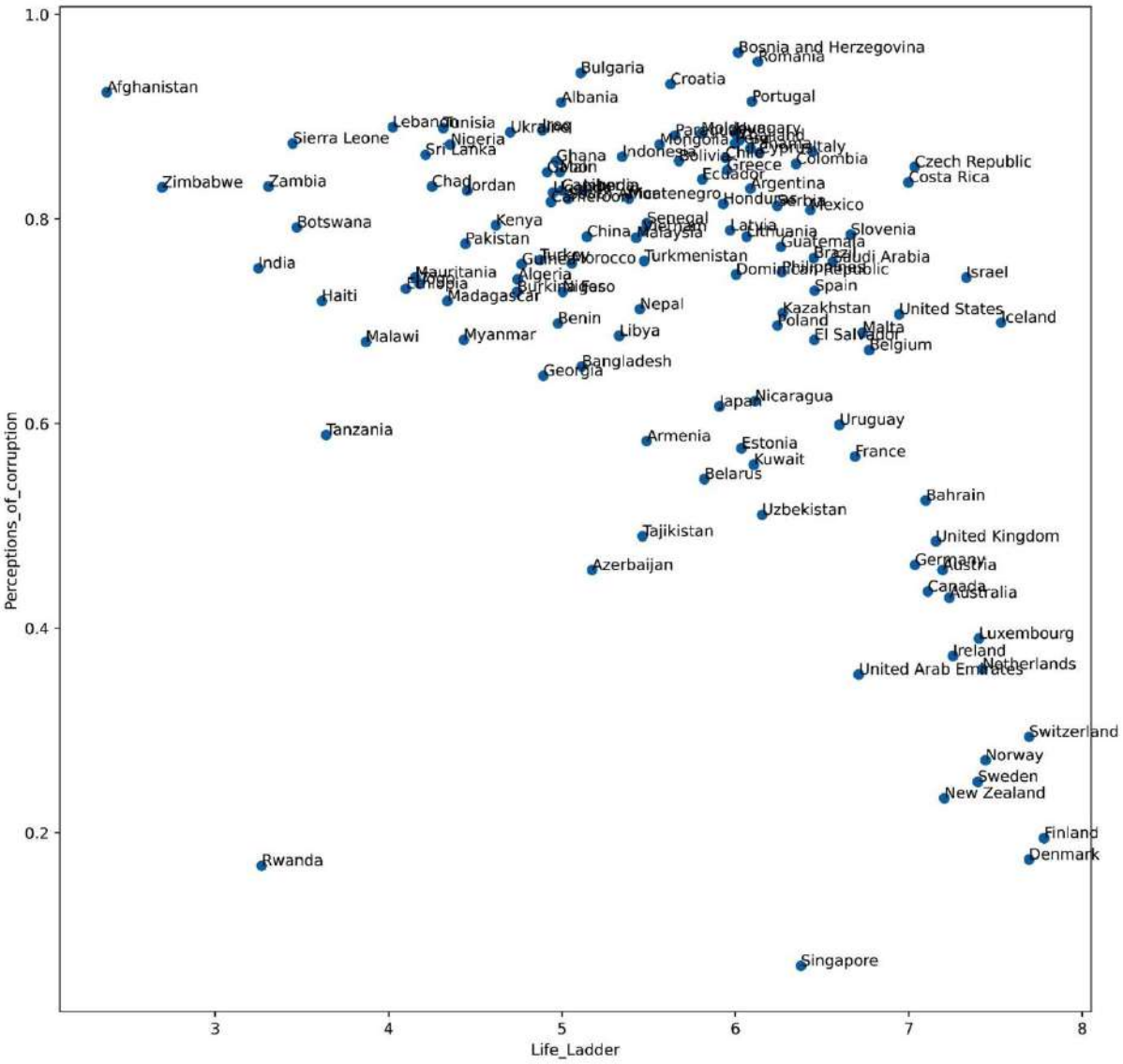
In [7]:

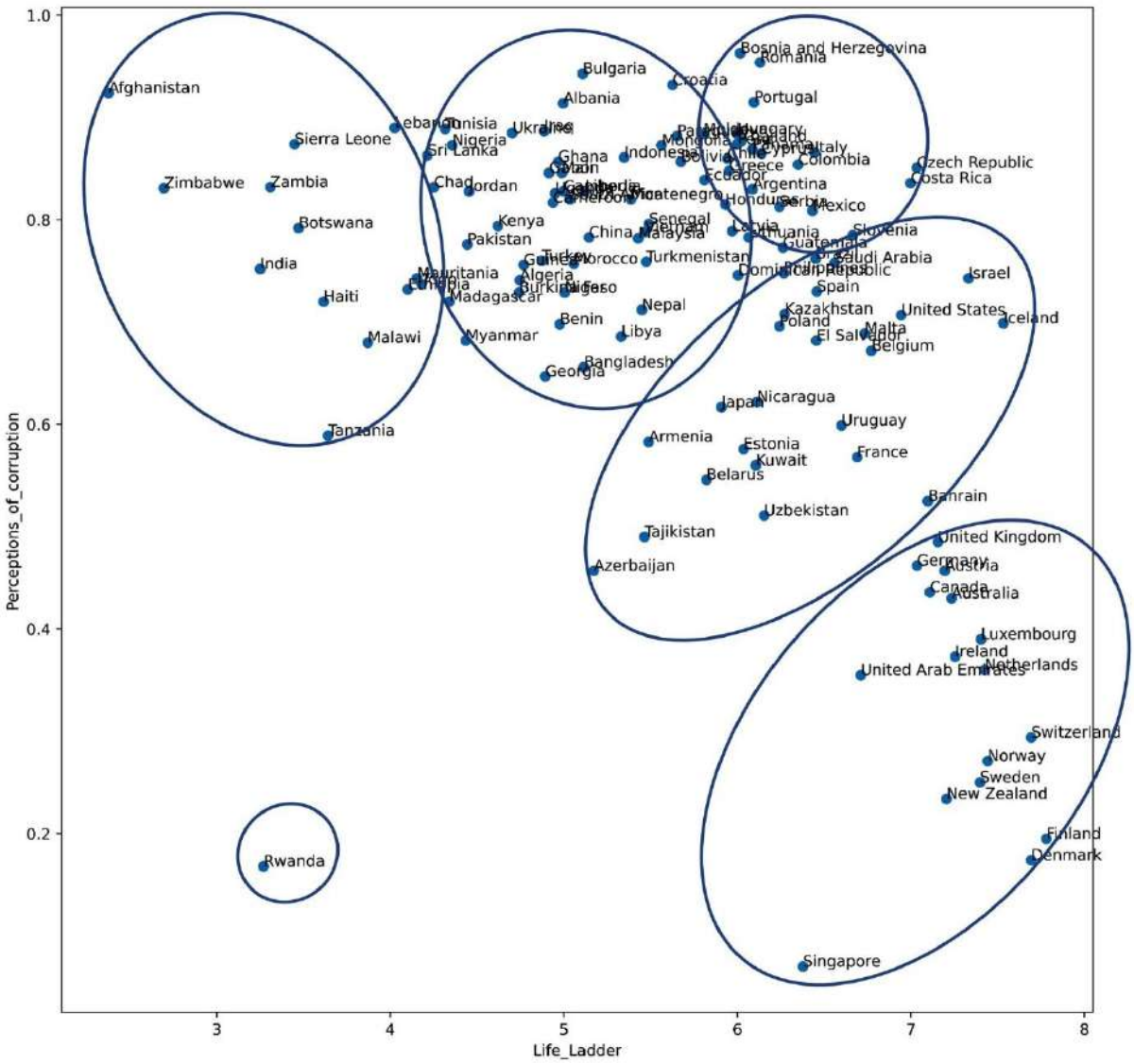
```
from sklearn.neighbors import KNeighborsClassifier

predictors = ['income_Normalized','score_Normalized']
target = 'default'

Xs = applicant_df[predictors].drop(index=[20])
y= applicant_df[target].drop(index=[20])

knn = KNeighborsClassifier(n_neighbors=4)
knn.fit(Xs, y)

newApplicant = pd.DataFrame({'income_Normalized':
                              applicant_df.iloc[20].income_Normalized,
                              'score_Normalized':
                              applicant_df.iloc[20].score_Normalized},
                             index = [20])
predict_y = knn.predict(newApplicant)
print(predict_y)
```

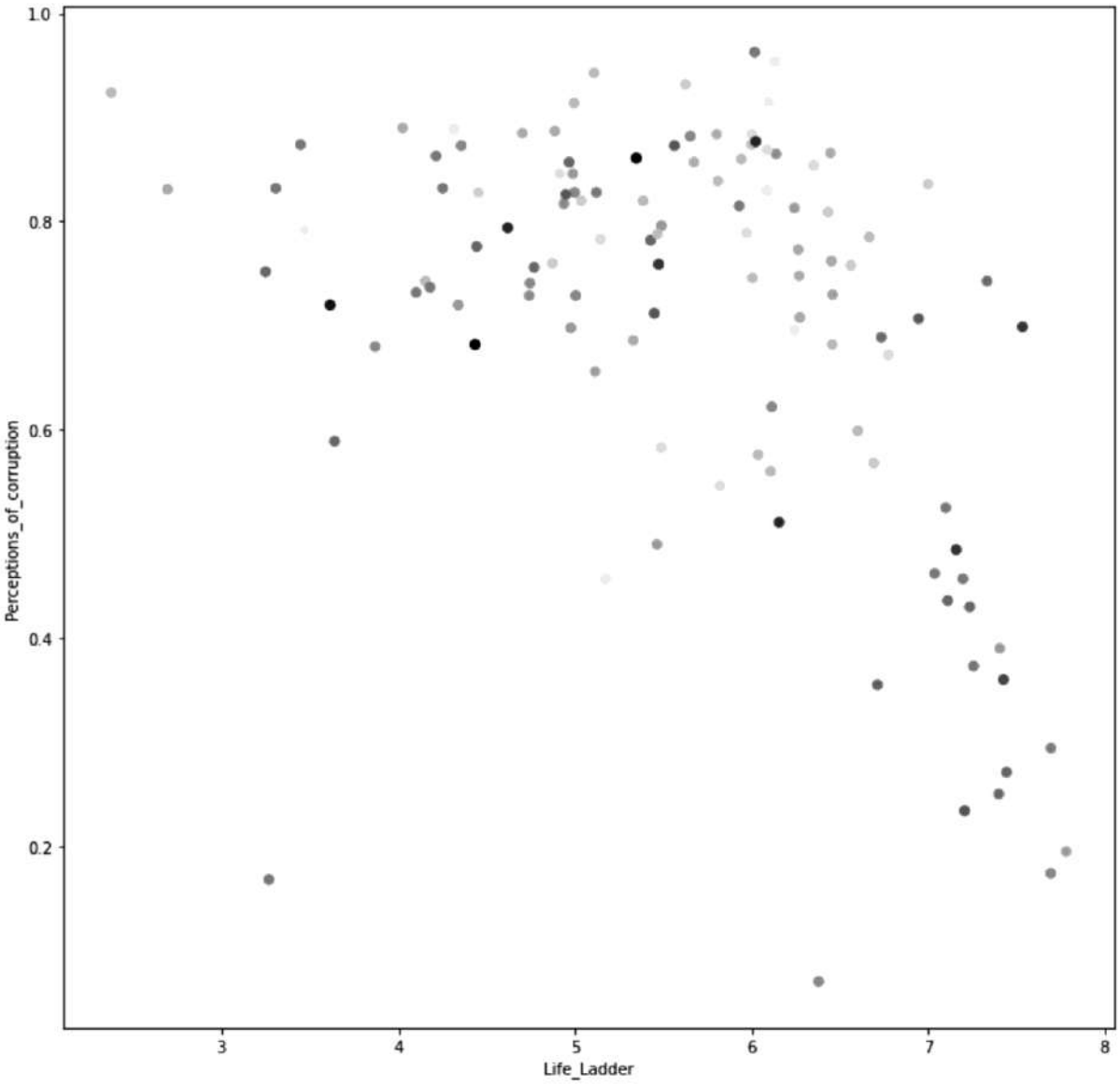
```
['NO']
```

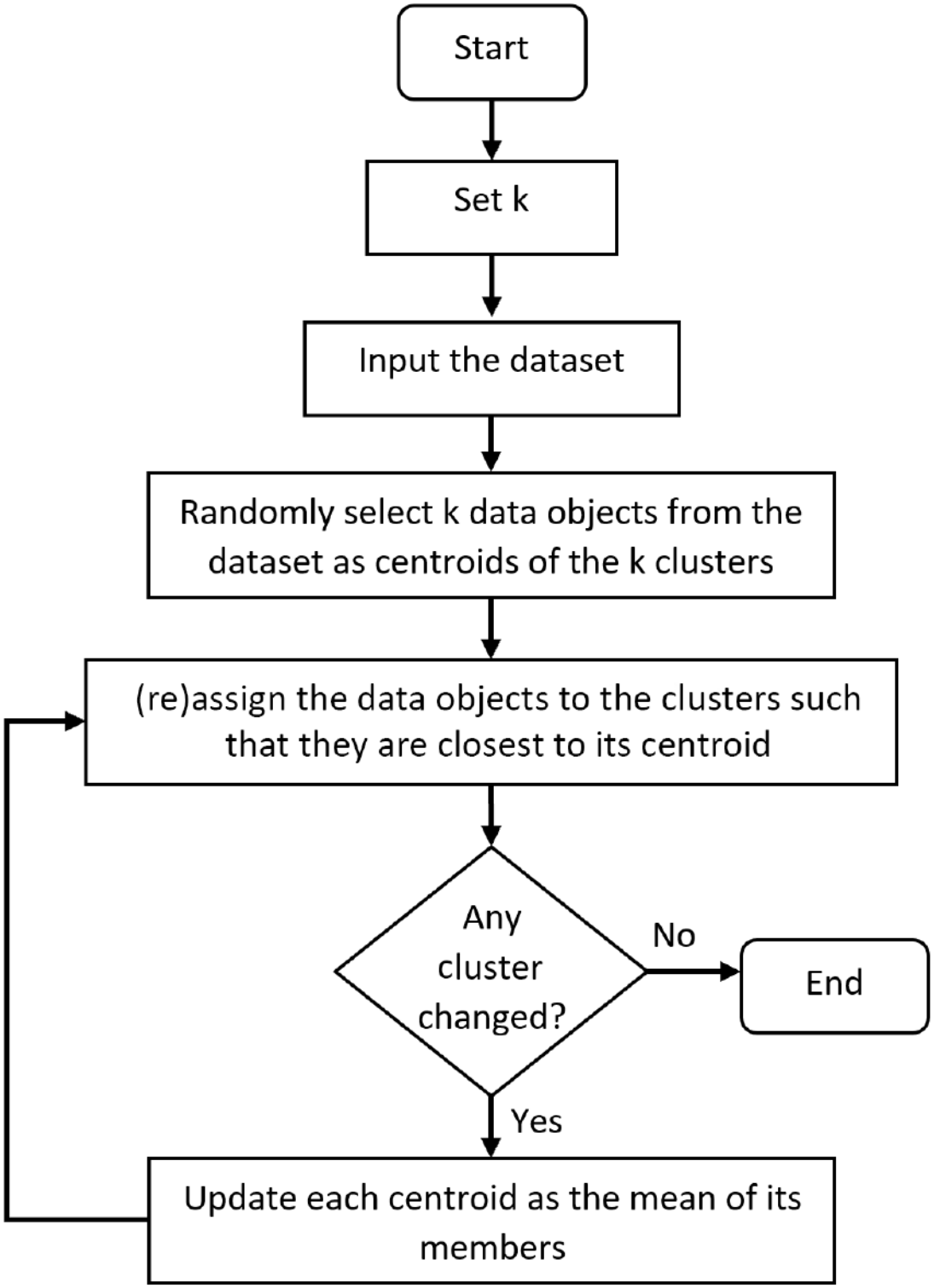
Chapter 8: Clustering Analysis




Perceptions_of_corruption
Life_Ladder
1.0
0.8
0.6
0.4
0.2
3
4
5
6
7
8
Bosnia and Herzegovina
Romania
Bulgaria
Croatia
Afghanistan
Albania
Portugal
Sierra Leone
Nigeria
Sri Lanka
Ghana
Indonesia
Italy
Colombia
Czech Republic
Costa Rica
Zimbabwe
Zambia
Chad
Jordan
Ecuador
Argentina
Honduras
Mexico
Botswana
Kenya
Senegal
Latvia
Lithuania
Slovenia
Pakistan
China
Malaysia
Guatemala
India
Turkmenistan
Brazil
Saudi Arabia
Israel
Mauritania
Algeria
Spain
Haiti
Madagascar
Nepal
Kazakhstan
United States
Iceland
Benin
Poland
Malta
Malawi
Myanmar
Libya
El Salvador
Belgium
Bangladesh
Georgia
Nicaragua
Japan
Uruguay
Tanzania
Armenia
Estonia
France
Kuwait
Belarus
Bahrain
Uzbekistan
Tajikistan
United Kingdom
Germany
Austria
Azerbaijan
Canada
Australia
Luxembourg
Ireland
Netherlands
United Arab Emirates
Switzerland
Norway
Sweden
New Zealand
Finland
Denmark
Rwanda
Singapore


Perceptions_of_corruption
Life_Ladder
1.0
0.8
0.6
0.4
0.2
3
4
5
6
7
8
Afghanistan
Zimbabwe
Zambia
Botswana
India
Haiti
Malawi
Tanzania
Sierra Leone
Rwanda
Bosnia and Herzegovina
Romania
Bulgaria
Croatia
Albania
Portugal
Czech Republic
Costa Rica
Israel
Iceland
United States
Spain
Kazakhstan
Poland
Malta
Belgium
Nicaragua
Uruguay
Armenia
Estonia
Kuwait
France
Belarus
Bahrain
Uzbekistan
Tajikistan
Azerbaijan
United Kingdom
Germany
Austria
Canada
Australia
Luxembourg
Ireland
Netherlands
United Arab Emirates
Switzerland
Norway
Sweden
New Zealand
Finland
Denmark
Singapore
Nepal
Benin
Libya
Bangladesh
Georgia
Myanmar
Madagascar
Mauritania
Algeria
Pakistan
Kenya
China
Malaysia
Senegal
Turkmenistan
Morocco
Jordan
Chad
Nigeria
Sri Lanka
Ghana
Indonesia
Bolivia
Chile
Cyprus
Italy
Colombia
Greece
Ecuador
Argentina
Honduras
Serbia
Mexico
Latvia
Lithuania
Slovenia
Guatemala
Saudi Arabia
Dominican Republic
Philippines
Tunisia
Ukraine
Lebanon
Hungary
Moldova
Mongolia
Cameroon
Montenegro
Senegal

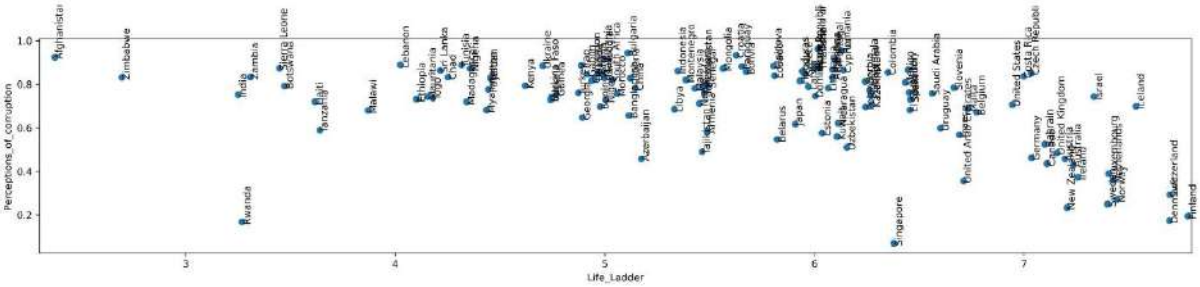

1.0
0.8
0.6
0.4
0.2
Perceptions_of_corruption
3
4
5
6
7
8
Life_Ladder


Start
Set k
Input the dataset
Randomly select k data objects from the dataset as centroids of the k clusters
(re)assign the data objects to the clusters such that they are closest to its centroid
Any cluster changed?
No
End
Yes
Update each centroid as the mean of its members

In [10]:

```
from sklearn.cluster import KMeans
dimensions = ['Life_Ladder','Perceptions_of_corruption']
Xs = report2019_df[dimensions]
kmeans = KMeans(n_clusters=6)
kmeans.fit(Xs)

for i in range(6):
    BM = kmeans.labels_==i
    print('Cluster {}: {}'.format(i,report2019_df[BM].Name.values))
```

```
Cluster 0: ['Australia' 'Austria' 'Bahrain' 'Canada' 'Denmark' 'Finland' 'G
ermany'
 'Iceland' 'Ireland' 'Israel' 'Luxembourg' 'Netherlands' 'New Zealand'
 'Norway' 'Sweden' 'Switzerland' 'United Kingdom']
Cluster 1: ['Albania' 'Algeria' 'Armenia' 'Azerbaijan' 'Bangladesh' 'Benin'
 'Bulgaria' 'Burkina Faso' 'Cambodia' 'Cameroon' 'China' 'Gabon' 'Georgia'
 'Ghana' 'Guinea' 'Indonesia' 'Iraq' 'Liberia' 'Libya' 'Malaysia' 'Mali'
 'Mongolia' 'Montenegro' 'Morocco' 'Nepal' 'Niger' 'Senegal'
 'South Africa' 'Tajikistan' 'Turkey' 'Turkmenistan' 'Uganda' 'Vietnam']
Cluster 2: ['Argentina' 'Belarus' 'Bolivia' 'Bosnia and Herzegovina' 'Chil
e'
 'Colombia' 'Croatia' 'Cyprus' 'Dominican Republic' 'Ecuador' 'Estonia'
 'Greece' 'Guatemala' 'Honduras' 'Hungary' 'Japan' 'Kazakhstan' 'Kuwait'
 'Latvia' 'Lithuania' 'Moldova' 'Nicaragua' 'Panama' 'Paraguay' 'Peru'
 'Philippines' 'Poland' 'Portugal' 'Romania' 'Serbia' 'Thailand'
 'Uzbekistan']
Cluster 3: ['Afghanistan' 'Botswana' 'Haiti' 'India' 'Rwanda' 'Sierra Leon
e'
 'Tanzania' 'Zambia' 'Zimbabwe']
Cluster 4: ['Belgium' 'Brazil' 'Costa Rica' 'Czech Republic' 'El Salvador'
'France'
 'Italy' 'Malta' 'Mexico' 'Saudi Arabia' 'Singapore' 'Slovenia' 'Spain'
 'United Arab Emirates' 'United States' 'Uruguay']
Cluster 5: ['Chad' 'Ethiopia' 'Jordan' 'Kenya' 'Lebanon' 'Madagascar' 'Mala
wi'
 'Mauritania' 'Myanmar' 'Nigeria' 'Pakistan' 'Sri Lanka' 'Togo' 'Tunisia'
 'Ukraine']
```

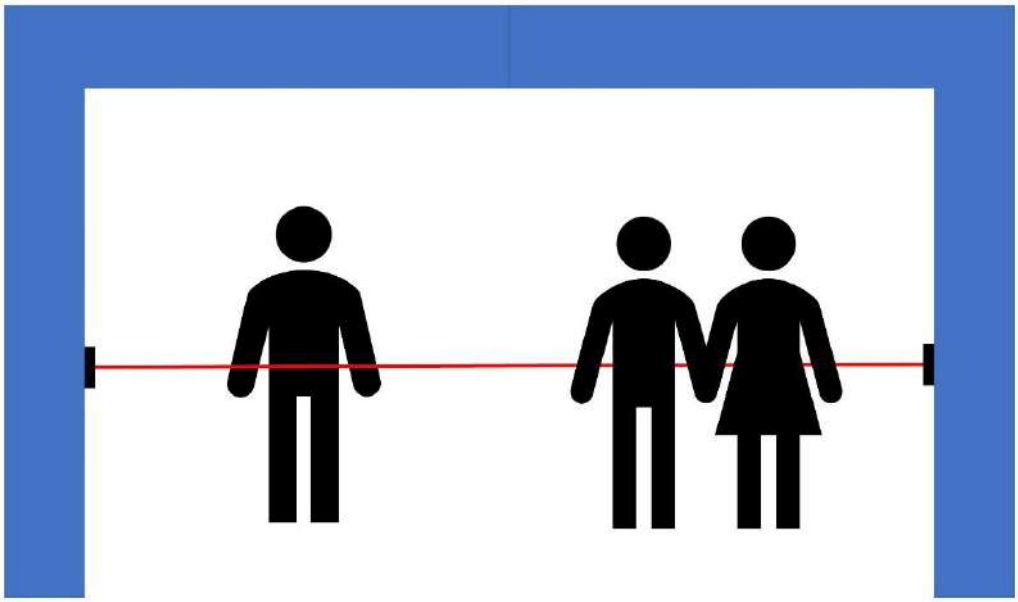


In [12]:

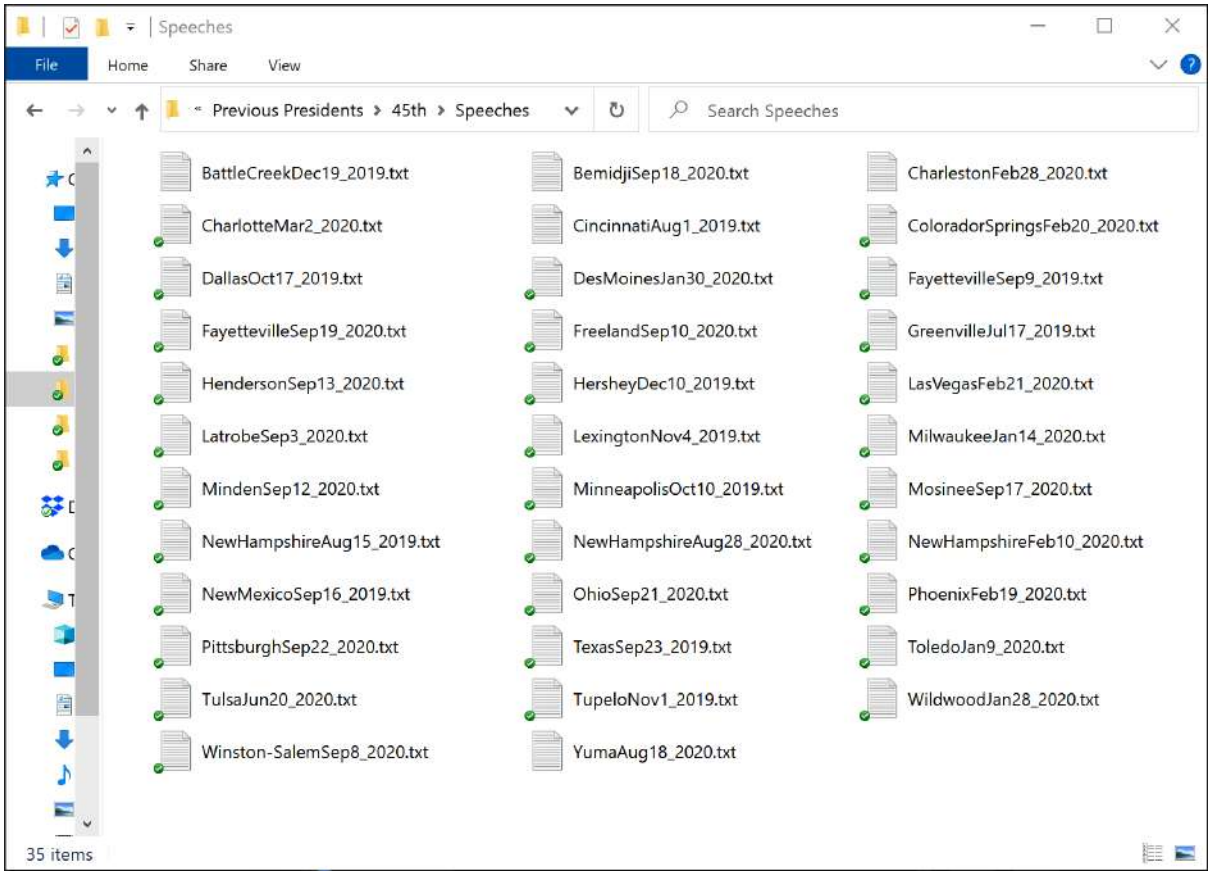
```
dimensions = ['Life_Ladder','Perceptions_of_corruption']
Xs = report2019_df[dimensions]
Xs = (Xs - Xs.min())/(Xs.max()-Xs.min())
kmeans = KMeans(n_clusters=6)
kmeans.fit(Xs)

for i in range(6):
    BM = kmeans.labels_==i
    print('Cluster {}: {}'.format(i,report2019_df[BM].Name.values))
```

```
Cluster 0: ['Australia' 'Austria' 'Canada' 'Denmark' 'Finland' 'Germany' 'I
reland'
 'Luxembourg' 'Netherlands' 'New Zealand' 'Norway' 'Singapore' 'Sweden'
 'Switzerland' 'United Arab Emirates' 'United Kingdom']
Cluster 1: ['Argentina' 'Bolivia' 'Bosnia and Herzegovina' 'Brazil' 'Chile'
 'Colombia' 'Costa Rica' 'Croatia' 'Cyprus' 'Czech Republic'
 'Dominican Republic' 'Ecuador' 'Greece' 'Guatemala' 'Honduras' 'Hungary'
 'Italy' 'Latvia' 'Lithuania' 'Mexico' 'Moldova' 'Mongolia' 'Panama'
 'Paraguay' 'Peru' 'Philippines' 'Portugal' 'Romania' 'Saudi Arabia'
 'Serbia' 'Slovenia' 'Thailand']
Cluster 2: ['Albania' 'Algeria' 'Bangladesh' 'Benin' 'Bulgaria' 'Burkina Fa
so'
 'Cambodia' 'Cameroon' 'China' 'Gabon' 'Georgia' 'Ghana' 'Guinea'
 'Indonesia' 'Iraq' 'Jordan' 'Kenya' 'Liberia' 'Libya' 'Malaysia' 'Mali'
 'Montenegro' 'Morocco' 'Myanmar' 'Nepal' 'Niger' 'Pakistan' 'Senegal'
 'South Africa' 'Turkey' 'Turkmenistan' 'Uganda' 'Ukraine' 'Vietnam']
Cluster 3: ['Afghanistan' 'Botswana' 'Chad' 'Ethiopia' 'Haiti' 'India' 'Leb
anon'
 'Madagascar' 'Malawi' 'Mauritania' 'Nigeria' 'Sierra Leone' 'Sri Lanka'
 'Tanzania' 'Togo' 'Tunisia' 'Zambia' 'Zimbabwe']
Cluster 4: ['Armenia' 'Azerbaijan' 'Bahrain' 'Belarus' 'Belgium' 'El Salvad
or'
 'Estonia' 'France' 'Iceland' 'Israel' 'Japan' 'Kazakhstan' 'Kuwait'
 'Malta' 'Nicaragua' 'Poland' 'Spain' 'Tajikistan' 'United States'
 'Uruguay' 'Uzbekistan']
Cluster 5: ['Rwanda']
```

Chapter 9: Data Cleaning Level I - Cleaning Up the Table



	Generals cleaning steps	Steps for the analytic tools	Steps for analytic goals
Level I -	A standard data structure Attributes have intuitive and codable names Rows have a unique identifier		
Level II -		Change data structure Restructure the dataset	Restructure the dataset
Level III -			Deal with data errors Deal with missing values Detect and handle outliers


Speeches
File
Home
Share
View
Previous Presidents › 45th › Speeches
Search Speeches
BattleCreekDec19_2019.txt
BemidjiSep18_2020.txt
CharlestonFeb28_2020.txt
CharlotteMar2_2020.txt
CincinnatiAug1_2019.txt
ColoradorSpringsFeb20_2020.txt
DallasOct17_2019.txt
DesMoinesJan30_2020.txt
FayettevilleSep9_2019.txt
FayettevilleSep19_2020.txt
FreelandSep10_2020.txt
GreenvilleJul17_2019.txt
HendersonSep13_2020.txt
HersheyDec10_2019.txt
LasVegasFeb21_2020.txt
LatrobeSep3_2020.txt
LexingtonNov4_2019.txt
MilwaukeeJan14_2020.txt
MindenSep12_2020.txt
MinneapolisOct10_2019.txt
MosineeSep17_2020.txt
NewHampshireAug15_2019.txt
NewHampshireAug28_2020.txt
NewHampshireFeb10_2020.txt
NewMexicoSep16_2019.txt
OhioSep21_2020.txt
PhoenixFeb19_2020.txt
PittsburghSep22_2020.txt
TexasSep23_2019.txt
ToledoJan9_2020.txt
TulsaJun20_2020.txt
TupeloNov1_2019.txt
WildwoodJan28_2020.txt
Winston-SalemSep8_2020.txt
YumaAug18_2020.txt
35 items

In [7]:

```
air_df = pd.read_csv('TempData.csv')
air_df
```

Out[7]:

	Temp	Year	Month	Day	Time
0	79.0	2016	1	1	00:00:00
1	79.0	2016	1	1	00:30:00
2	79.0	2016	1	1	01:00:00
3	77.0	2016	1	1	01:30:00
4	78.0	2016	1	1	02:00:00
...	...	...	...	...	...
20448	77.0	2016	12	31	22:00:00
20449	77.0	2016	12	31	22:30:00
20450	77.0	2016	12	31	23:00:00
20451	77.0	2016	12	31	23:00:00
20452	77.0	2016	12	31	23:30:00

20453 rows × 5 columns

			Temp
Month	**Day**	**Time**	
1	**1**	**00:00:00**	79.0
		00:30:00	79.0
		01:00:00	79.0
		01:30:00	77.0
		02:00:00	78.0
...	**...**	**...**	...
12	**31**	**22:00:00**	77.0
		22:30:00	77.0
		23:00:00	77.0
		23:00:00	77.0
		23:30:00	77.0

20453 rows × 1 columns

In [12]:

```
response_df = pd.read_csv('OSMI Mental Health in Tech Survey 2019.csv')
response_df.head(1)
```

Out[12]:

	*Are you self-employed? *	How many employees does your company or organization have?	Is your employer primarily a tech company/organization?	Is your primary role within your company related to tech/IT?	Does your employer provide mental health benefits as part of healthcare coverage?	Do you know the options for mental health care available under your employer-provided health coverage?	empl d men (for ex ca oth commun
0	False	26-100	True	True	I don't know	No	

1 rows × 82 columns

In [17]:

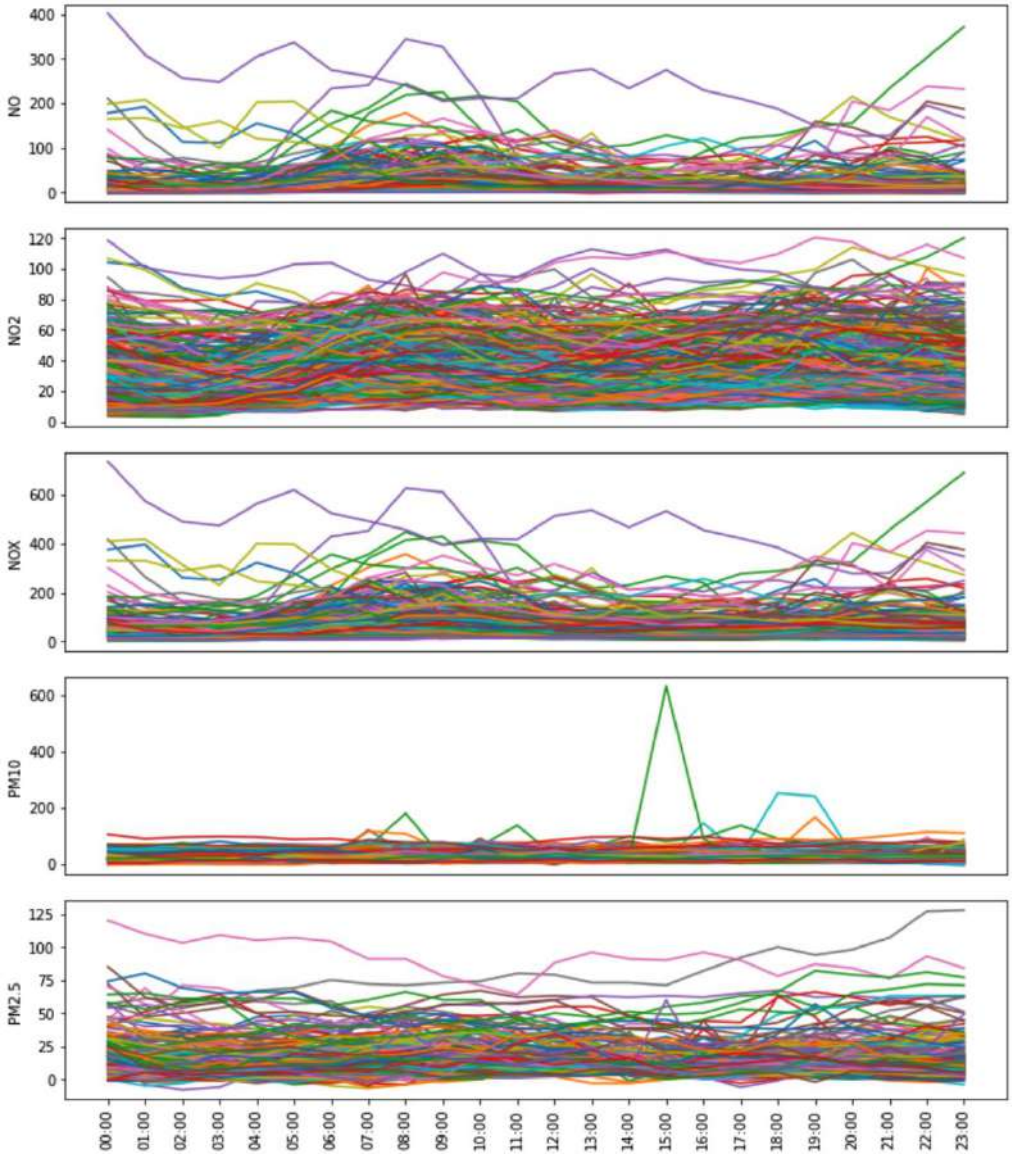
```
response_df.head(1)
```

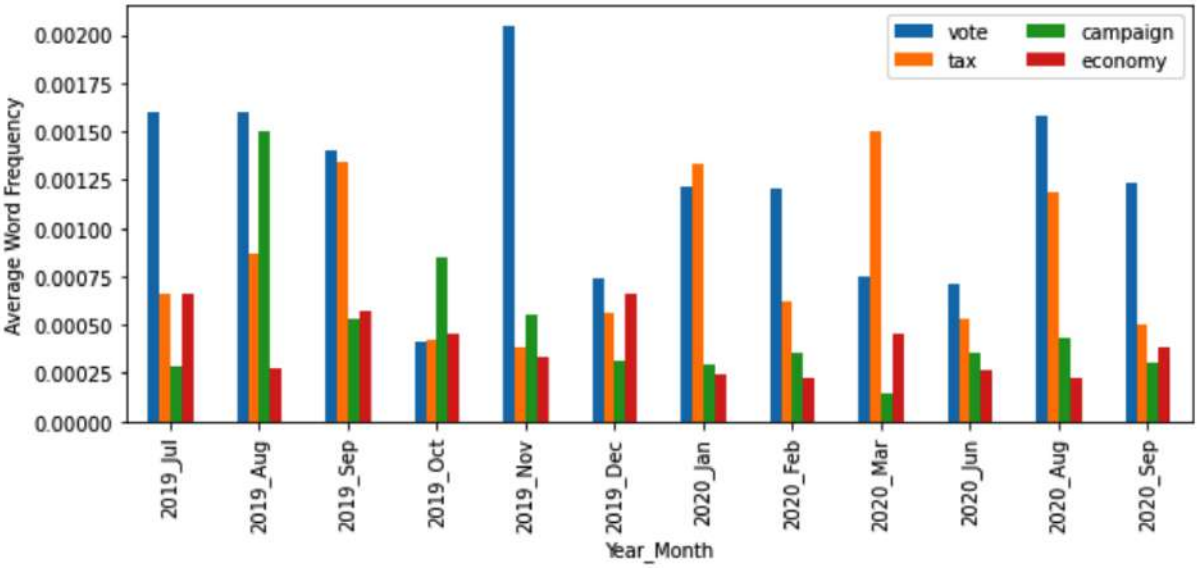
Out[17]:

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	...	Q73	Q74	Q75	Q76	Q77
0	False	26-100	True	True	I don't know	No	Yes	Yes	I don't know	Very easy	...	NaN	NaN	False	25	Male

1 rows × 82 columns

Chapter 10: Data Cleaning Level II - Unpacking, Restructuring, and Reformulating the Table





In [25]:
```
speech_df.head()
```

Out[25]:

	Content	City	Date	Day	Month	Year
0	Thank you. Thank you. Thank you to Vice Presid...	BattleCreek	2019-12-19	19	12	2019
1	There's a lot of people. That's great. Thank y...	Bemidji	2020-09-18	18	9	2020
2	Thank you. Thank you. Thank you. All I can say...	Charleston	2020-02-28	28	2	2020
3	I want to thank you very much. North Carolina,...	Charlotte	2020-03-02	2	3	2020
4	Thank you all. Thank you very much. Thank you ...	Cincinnati	2019-08-01	1	8	2019

In [27]:
```
speech_df.head()
```

Out[27]:

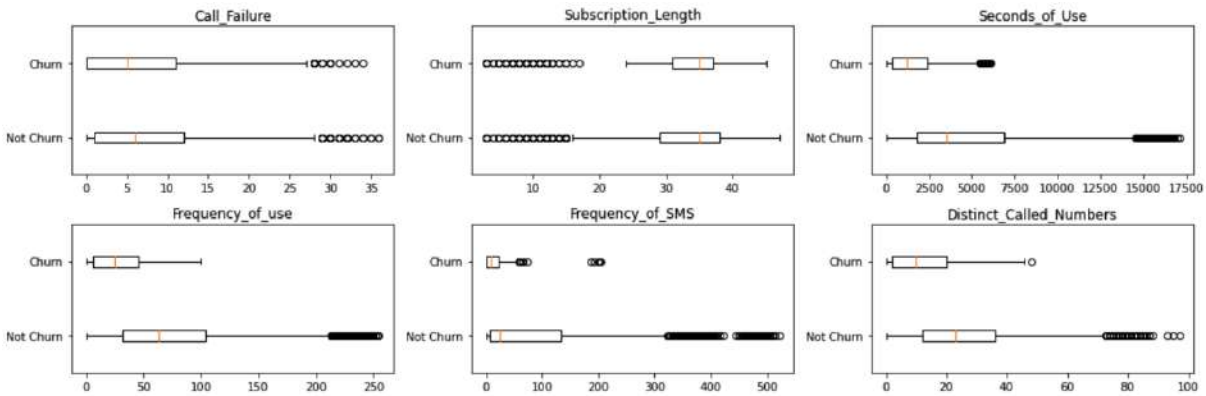
	Content	City	Date	Day	Month	Year	r_vote	r_tax	r_campaign	r_econon
0	Thank you. Thank you. Thank you to Vice Presid...	BattleCreek	2019-12-19	19	12	2019	0.000561	0.000505	0.000224	0.0006
1	There's a lot of people. That's great. Thank y...	Bemidji	2020-09-18	18	9	2020	0.000710	0.000237	0.000533	0.0000
2	Thank you. Thank you. Thank you. All I can say...	Charleston	2020-02-28	28	2	2020	0.000950	0.000317	0.000106	0.0000
3	I want to thank you very much. North Carolina,...	Charlotte	2020-03-02	2	3	2020	0.000750	0.001500	0.000150	0.0004
4	Thank you all. Thank you very much. Thank you ...	Cincinnati	2019-08-01	1	8	2019	0.001713	0.000857	0.001224	0.0002

Y_M	r_campaign	r_economy	r_tax	r_vote
2019_Aug	0.001499	0.000270	0.000872	0.001596
2019_Dec	0.000316	0.000665	0.000558	0.000739
2019_Jul	0.000283	0.000660	0.000660	0.001603
2019_Nov	0.000551	0.000333	0.000385	0.002048
2019_Oct	0.000533	0.000572	0.001340	0.001398
2019_Sep	0.000843	0.000448	0.000419	0.000409
2020_Aug	0.000428	0.000222	0.001189	0.001577
2020_Feb	0.000353	0.000224	0.000625	0.001206
2020_Jan	0.000299	0.000240	0.001331	0.001215
2020_Jun	0.000356	0.000267	0.000535	0.000713
2020_Mar	0.000150	0.000450	0.001500	0.000750
2020_Oct	0.000306	0.000386	0.000504	0.001235

In [34]:
```
customer_df = pd.read_csv('Customer Churn.csv')
customer_df.head(1)
```

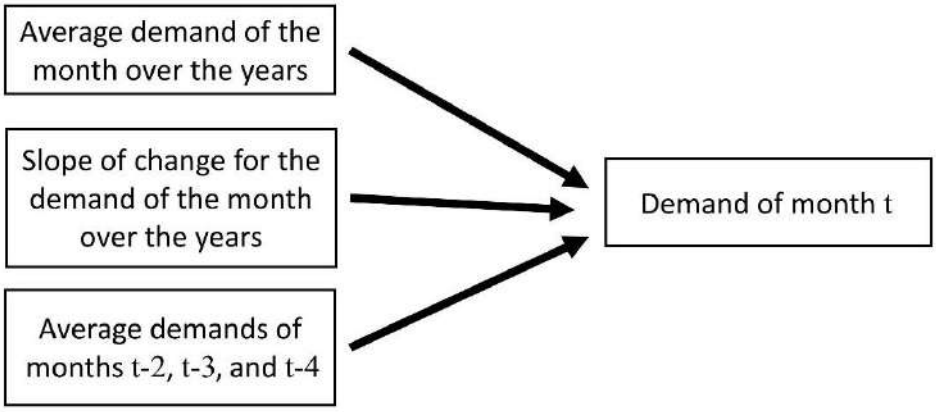
Out[34]:

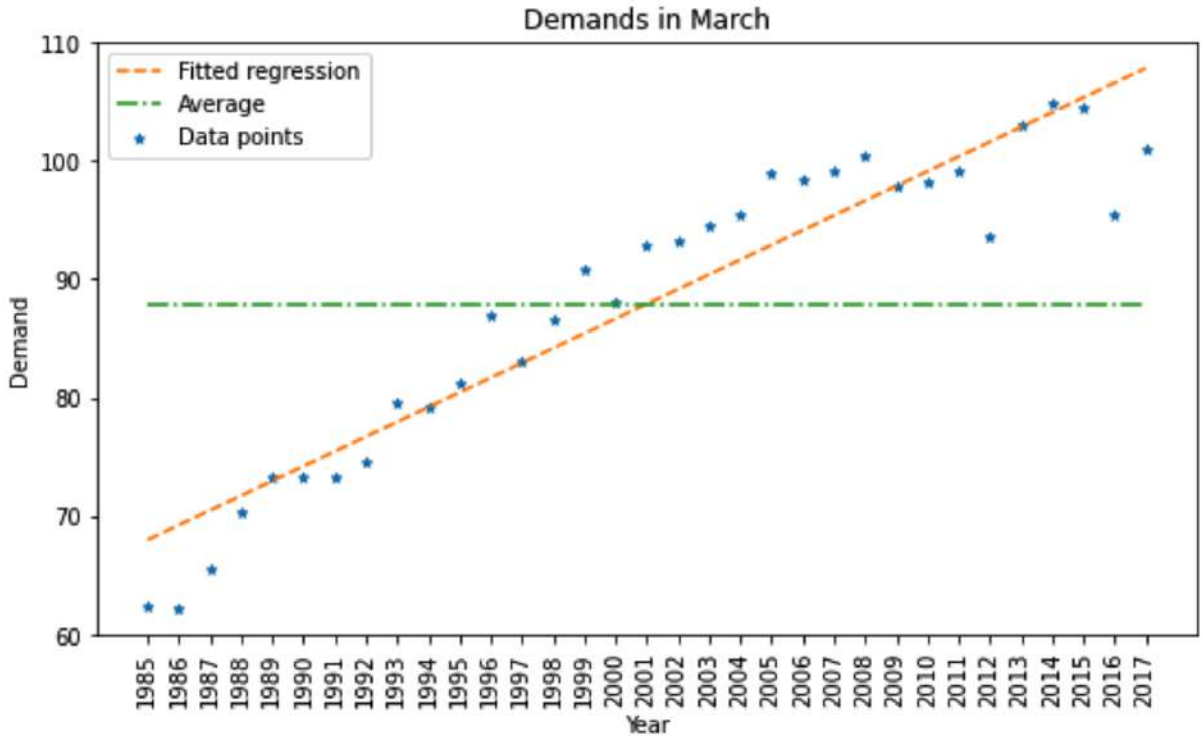
	Call Failure	Complains	Subscription Length	Seconds of Use	Frequency of use	Frequency of SMS	Distinct Called Numbers	Status	Churn
0	8	0	38	4370	71	5	17	1	



Independent Attributes

Dependent Attribute





In [39]:

```
month_df = pd.read_csv('Electric_Production.csv')
month_df
```

Out[39]:

	DATE	IPG2211A2N
0	1/1/1985	72.5052
1	2/1/1985	70.6720
2	3/1/1985	62.4502
3	4/1/1985	57.4714
4	5/1/1985	55.3151
...	...	...
392	9/1/2017	98.6154
393	10/1/2017	93.6137
394	11/1/2017	97.3359
395	12/1/2017	114.7212
396	1/1/2018	129.4048

397 rows × 2 columns

Date	Demand
1985-01-01	72.5052
1985-02-01	70.6720
1985-03-01	62.4502
1985-04-01	57.4714
1985-05-01	55.3151
...	...
2017-09-01	98.6154
2017-10-01	93.6137
2017-11-01	97.3359
2017-12-01	114.7212
2018-01-01	129.4048

397 rows × 1 columns

Date	IA1	IA2	IA3	DA
1987-01-01	NaN	NaN	NaN	NaN
1987-02-01	NaN	NaN	NaN	NaN
1987-03-01	NaN	NaN	NaN	NaN
1987-04-01	NaN	NaN	NaN	NaN
1987-05-01	NaN	NaN	NaN	NaN
...	...	...	...	...
2017-09-01	NaN	NaN	NaN	NaN
2017-10-01	NaN	NaN	NaN	NaN
2017-11-01	NaN	NaN	NaN	NaN
2017-12-01	NaN	NaN	NaN	NaN
2018-01-01	NaN	NaN	NaN	NaN

373 rows × 4 columns

In [44]:

```
predict_df.DA = month_df.loc['1987-01-01':].Demand
predict_df
```

Out[44]:

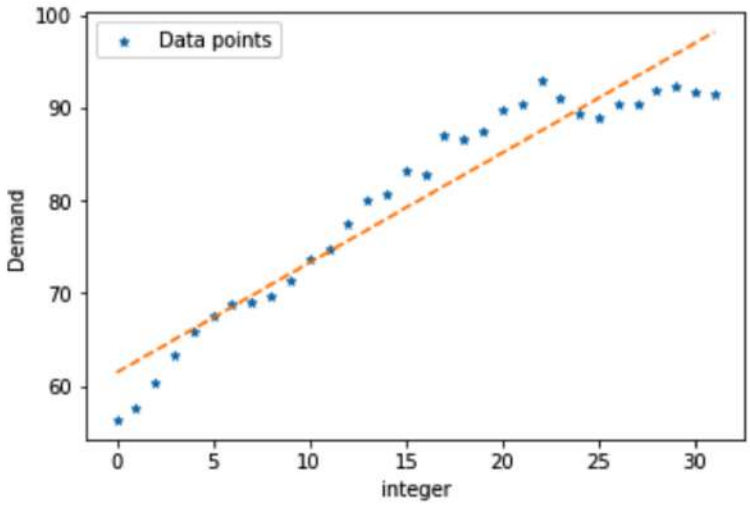
	IA1	IA2	IA3	DA
Date				
1987-01-01	NaN	NaN	NaN	73.8152
1987-02-01	NaN	NaN	NaN	70.0620
1987-03-01	NaN	NaN	NaN	65.6100
1987-04-01	NaN	NaN	NaN	60.1586
1987-05-01	NaN	NaN	NaN	58.8734
...	...	...	...	...
2017-09-01	NaN	NaN	NaN	98.6154
2017-10-01	NaN	NaN	NaN	93.6137
2017-11-01	NaN	NaN	NaN	97.3359
2017-12-01	NaN	NaN	NaN	114.7212
2018-01-01	NaN	NaN	NaN	129.4048

373 rows × 4 columns

In [51]:

```
row_date = '2017-10-01'
wdf = month_df.loc[:row_date].iloc[:-1]
BM = wdf.Month == 10
wdf = wdf[BM]
wdf.reset_index(drop=True,inplace=True)
wdf.drop(columns = ['Month'],inplace=True)
wdf['integer'] = range(len(wdf))
wdf['ones'] = 1

lm = LinearRegression()
lm.fit(wdf.drop(columns=['Demand']), wdf.Demand)
print('Slope = {}'.format(lm.coef_[0]))

wdf.plot.scatter(x='integer',y='Demand',marker='*',
                 label='Data points',c='C0')

b = lm.intercept_
a = lm.coef_[0]

X = wdf.integer
y = b + a*X

plt.plot(X,y,label = 'Fitted regression',linestyle='--',c='C1')
plt.show()
```

```
Slope = 1.1857728189149566
```

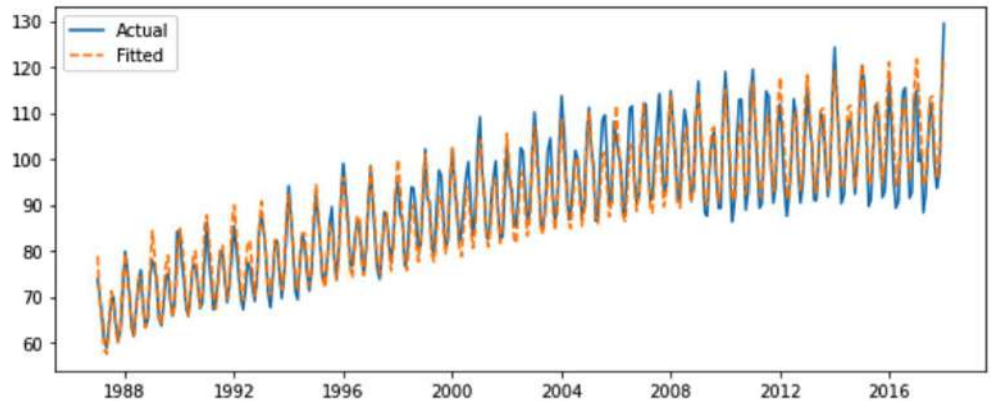


Date	IA1	IA2	IA3	DA
1987-01-01	72.905450	0.800500	59.291467	73.8152
1987-02-01	69.329450	-2.685100	61.669767	70.0620
1987-03-01	62.336150	-0.228100	67.097433	65.6100
1987-04-01	57.252150	-0.438500	70.670867	60.1586
1987-05-01	55.564400	0.498600	69.829067	58.8734
...	...	...	...	...
2017-09-01	86.105297	1.378406	102.129167	98.6154
2017-10-01	79.790228	1.185773	107.746067	93.6137
2017-11-01	82.692128	1.190510	106.566800	97.3359
2017-12-01	95.164994	1.421533	100.386767	114.7212
2018-01-01	101.272830	1.537419	96.521667	129.4048

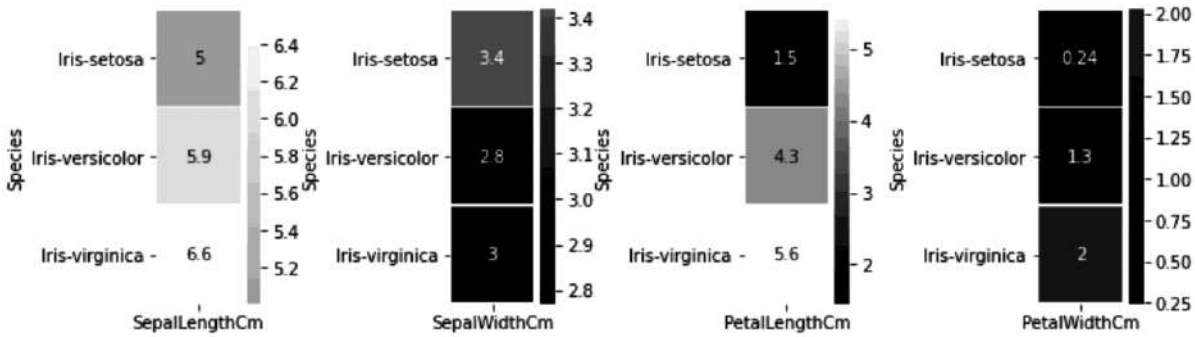
373 rows × 4 columns

In [269]:

```
plt.figure(figsize=(10,4))
plt.plot(X.index,y, label='Actual')
plt.plot(X.index,lm.predict(X),
         label = 'Fitted',linestyle='--')
plt.legend()
plt.show()
```



Chapter 11: Data Cleaning Level III- Missing Values, Outliers, and Errors



	Gender	Height	Year	GPA	Personality Type
1	1	190	Sophomore		ISTJ
2	1	189	Freshman	3.81	ESNJ
3	0	160	Freshman		ISTJ
4	1	181	Sophomore	3.95	INTP
5	1		Freshman	3.62	ISTJ
6	0	184	Freshman	3.87	
7	0	172	Junior	3.31	ISTP

	Gender	Height	Year	GPA	Personality Type
0	1	190.0	Sophomore	NaN	ISTJ
1	1	189.0	Freshman	3.81	ESNJ
2	0	160.0	Freshman	NaN	ISTJ
3	1	181.0	Sophomore	3.95	INTP
4	1	NaN	Freshman	3.62	ISTJ
5	0	184.0	Freshman	3.87	NaN
6	0	172.0	Junior	3.31	ISTP

In [3]:

```
air_df = pd.read_csv('Airdata.csv')
air_df
```

Out[3]:

	DateTime	Temperature	Humidity	Wind_Speed	Wind_Direction	NO2_Location_A	NO2_
0	1/1/2020 0:00	2.180529	87	1.484318	75.963760	39.23	
1	1/1/2020 1:00	1.490529	89	2.741678	113.198590	38.30	
2	1/1/2020 2:00	1.690529	85	3.563818	135.000000	NaN	
3	1/1/2020 3:00	1.430529	84	2.811690	129.805570	37.28	
4	1/1/2020 4:00	0.840529	86	1.800000	126.869896	29.97	
...	...	...	...	...	...	...	
8779	12/31/2020 19:00	4.920528	72	4.553679	251.565060	53.44	
8780	12/31/2020 20:00	4.990529	74	3.259938	186.340200	49.80	
8781	12/31/2020 21:00	4.360529	84	10.587917	252.181120	43.32	
8782	12/31/2020 22:00	3.820529	88	8.435069	219.805570	39.88	
8783	12/31/2020 23:00	3.170529	89	6.792466	212.005390	39.04	

8784 rows × 8 columns

In [4]:
```
air_df.info()
```

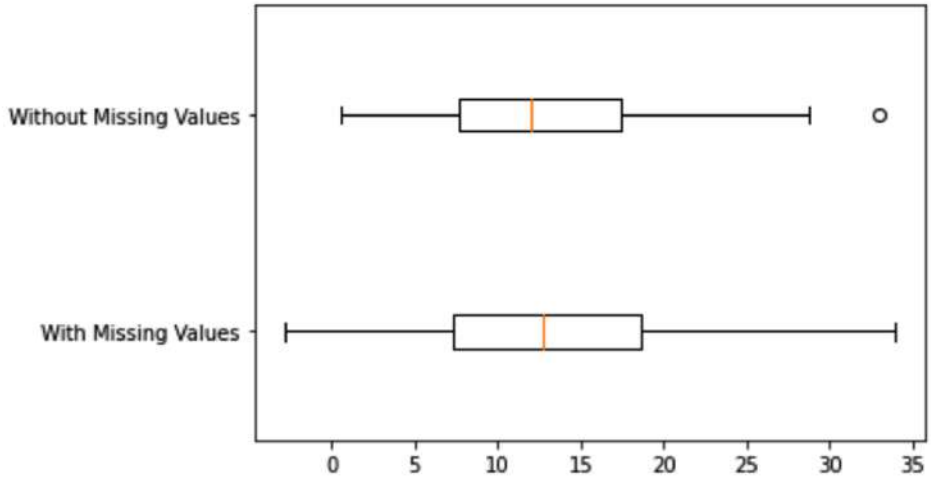
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8784 entries, 0 to 8783
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   DateTime        8784 non-null   object
 1   Temperature     8784 non-null   float64
 2   Humidity        8784 non-null   int64
 3   Wind_Speed      8784 non-null   float64
 4   Wind_Direction  8784 non-null   float64
 5   NO2_Location_A  8664 non-null   float64
 6   NO2_Location_B  8204 non-null   float64
 7   NO2_Location_C  8652 non-null   float64
dtypes: float64(6), int64(1), object(1)
memory usage: 549.1+ KB
```

In [5]:
```
print('Number of missing values:')
for col in air_df.columns:
    n_MV = sum(air_df[col].isna())
    print('{}:{}'.format(col,n_MV))
```

```
Number of missing values:
DateTime:0
Temperature:0
Humidity:0
Wind_Speed:0
Wind_Direction:0
NO2_Location_A:120
NO2_Location_B:580
NO2_Location_C:132
```

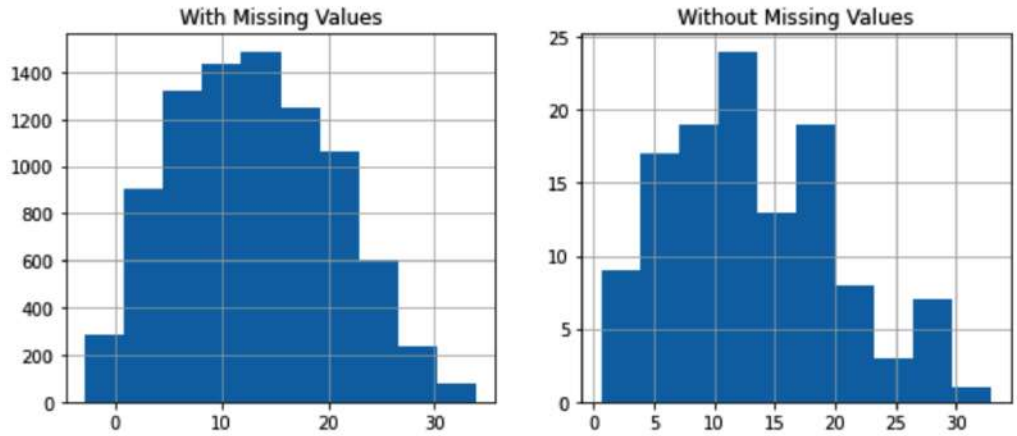
In [6]:

```
BM_MV = air_df.NO2_Location_A.isna()
MV_labels = ['With Missing Values','Without Missing Values']

box_sr = pd.Series('',index = BM_MV.unique())

for poss in BM_MV.unique():
    BM = BM_MV == poss
    box_sr[poss] = air_df[BM].Temperature

plt.boxplot(box_sr,vert=False)
plt.yticks([1,2],MV_labels)
plt.show()
```



In [7]:

```
BM_MV = air_df.NO2_Location_A.isna()
temp_range = (air_df.Temperature.min(),air_df.Temperature.max())
MV_labels = ['With Missing Values','Without Missing Values']

plt.figure(figsize=(10,4))

for i,poss in enumerate(BM_MV.unique()):
    plt.subplot(1,2,i+1)
    BM = BM_MV == poss
    air_df[BM].Temperature.hist()
    plt.xlim = temp_range
    plt.title(MV_labels[i])

plt.show()
```



In [8]:

```
from scipy.stats import ttest_ind
BM_MV = air_df.NO2_Location_A.isna()
ttest_ind(air_df[BM_MV].Temperature, air_df[~BM_MV].Temperature)
```

Out[8]:

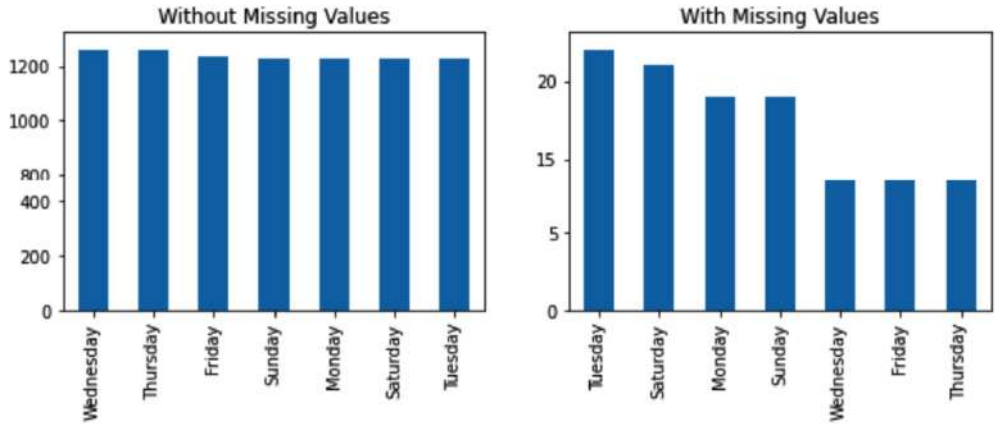
```
Ttest_indResult(statistic=0.05646499065315542, pvalue=0.9549726689684548)
```

In [10]:

```
from scipy.stats import ttest_ind
def Diagnose_MV_Numerical(df,str_att_name,BM_MV):
    MV_labels = {True:'With Missing Values',False:'Without Missing Values'}

    labels=[]
    box_sr = pd.Series('',index = BM_MV.unique())
    for poss in BM_MV.unique():
        BM = BM_MV == poss
        box_sr[poss] = df[BM][str_att_name].dropna()
        labels.append(MV_labels[poss])

    plt.boxplot(box_sr,vert=False)
    plt.yticks([1,2],labels)
    plt.xlabel(str_att_name)
    plt.show()

    plt.figure(figsize=(10,4))

    att_range = (df[str_att_name].min(),df[str_att_name].max())

    for i,poss in enumerate(BM_MV.unique()):
        plt.subplot(1,2,i+1)
        BM = BM_MV == poss
        df[BM][str_att_name].hist()
        plt.xlim = att_range
        plt.xlabel(str_att_name)
        plt.title(MV_labels[poss])

    plt.show()

    group_1_data = df[BM_MV][str_att_name].dropna()
    group_2_data = df[~BM_MV][str_att_name].dropna()

    p_value = ttest_ind(group_1_data,group_2_data).pvalue

    print('p-value of t-test: {}'.format(p_value))
```

In [14]:

```
BM_MV = air_df.NO2_Location_A.isna()
MV_labels = ['Without Missing Values','With Missing Values']

plt.figure(figsize=(10,4))
for i,poss in enumerate(BM_MV.unique()):
    plt.subplot(1,2,i+1)
    BM = BM_MV == poss
    air_df[BM].weekday.value_counts().plot.bar()
    plt.title(MV_labels[i])
plt.show()
```



In [15]:

```
from scipy.stats import chi2_contingency
BM_MV = air_df.NO2_Location_A.isna()
contigency_table = pd.crosstab(BM_MV,air_df.weekday)
contigency_table
```

Out[15]:

weekday	Friday	Monday	Saturday	Sunday	Thursday	Tuesday	Wednesday
NO2_Location_A							
False	1235	1229	1227	1229	1259	1226	1259
True	13	19	21	19	13	22	13

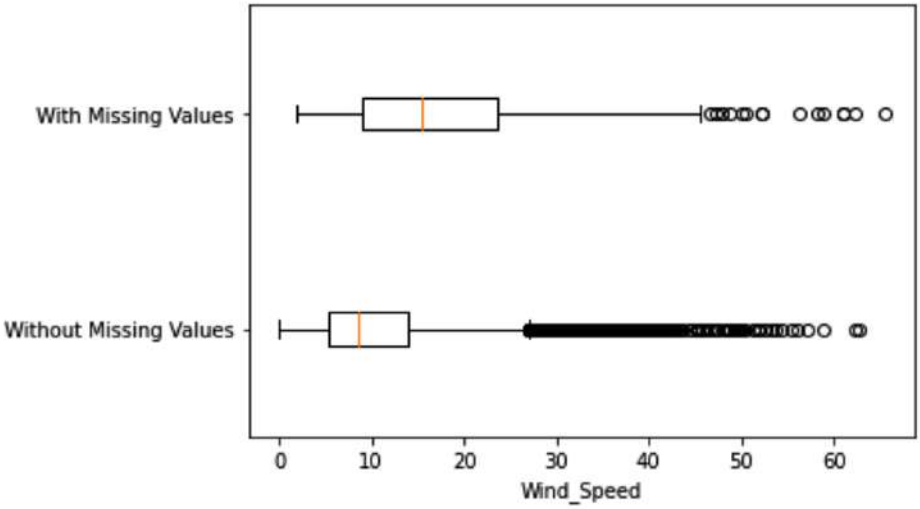
In [16]:

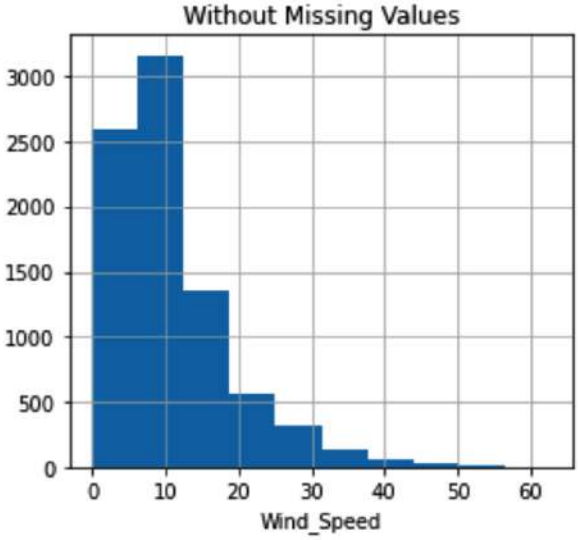
```
chi2_contingency(contigency_table)
```

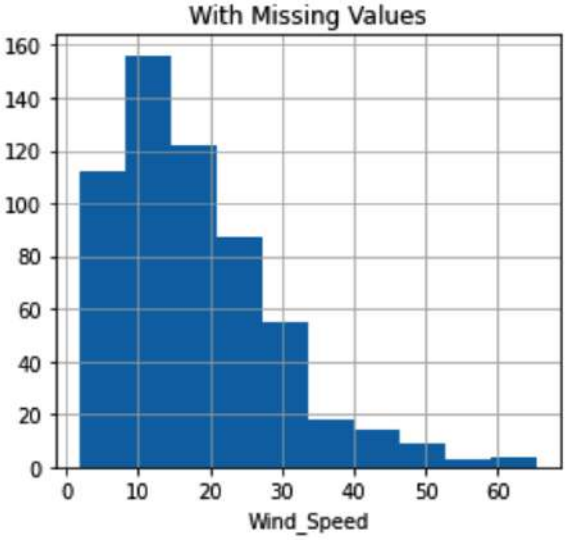
Out[16]:

```
(6.048964133655503,
 0.41772751510388023,
 6,
 array([[1230.95081967, 1230.95081967, 1230.95081967, 1230.95081967,
         1254.62295082, 1230.95081967, 1254.62295082],
        [  17.04918033,   17.04918033,   17.04918033,   17.04918033,
           17.37704918,   17.04918033,   17.37704918]]))
```

Diagnosis Analysis of Missing Values for Wind_Speed:

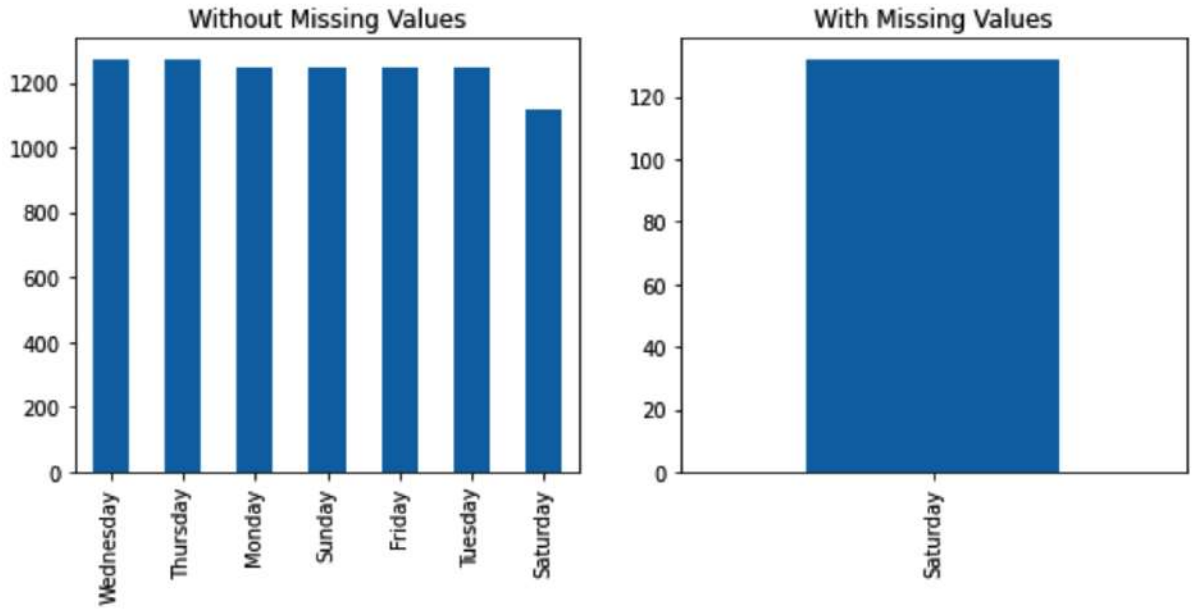




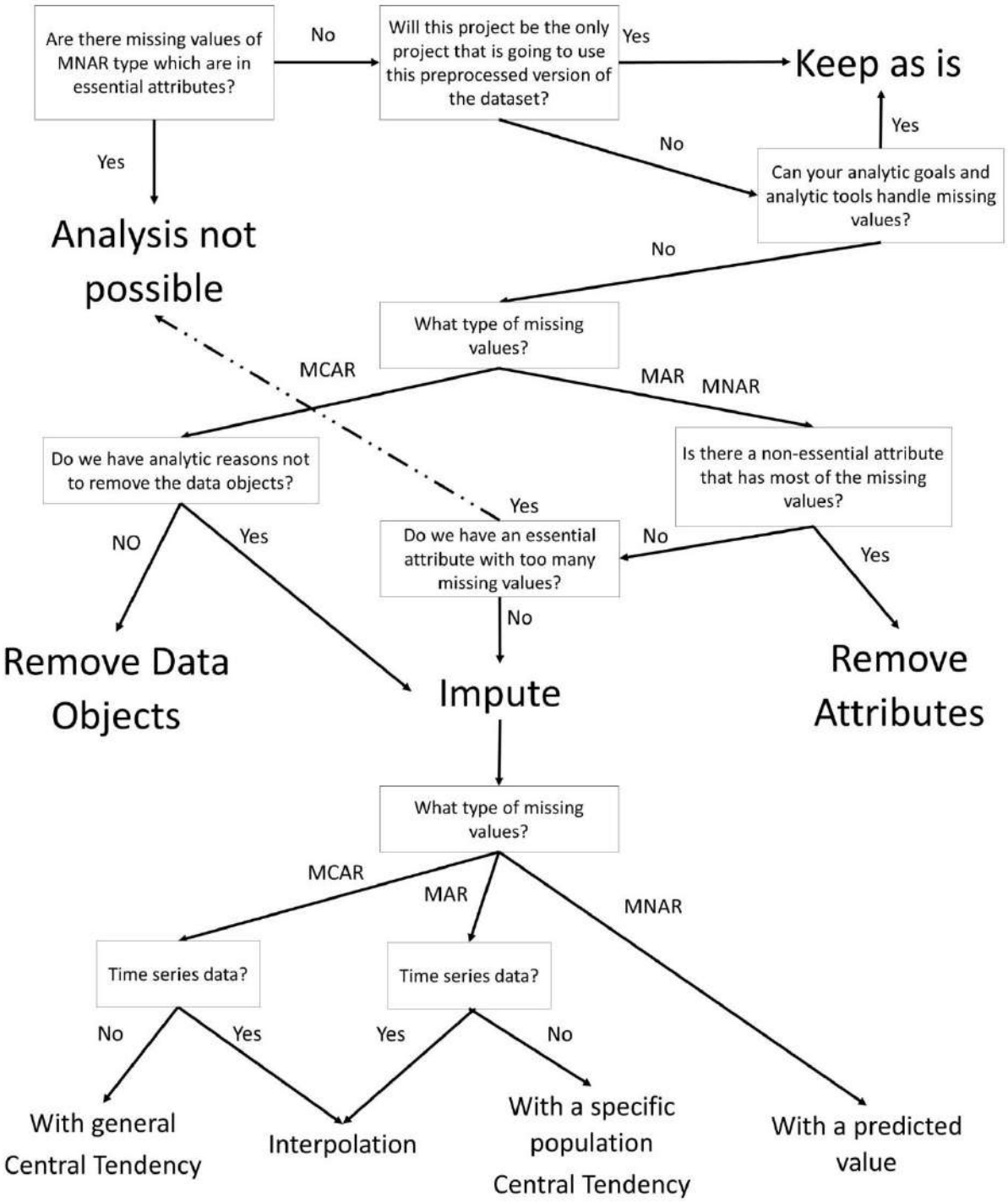


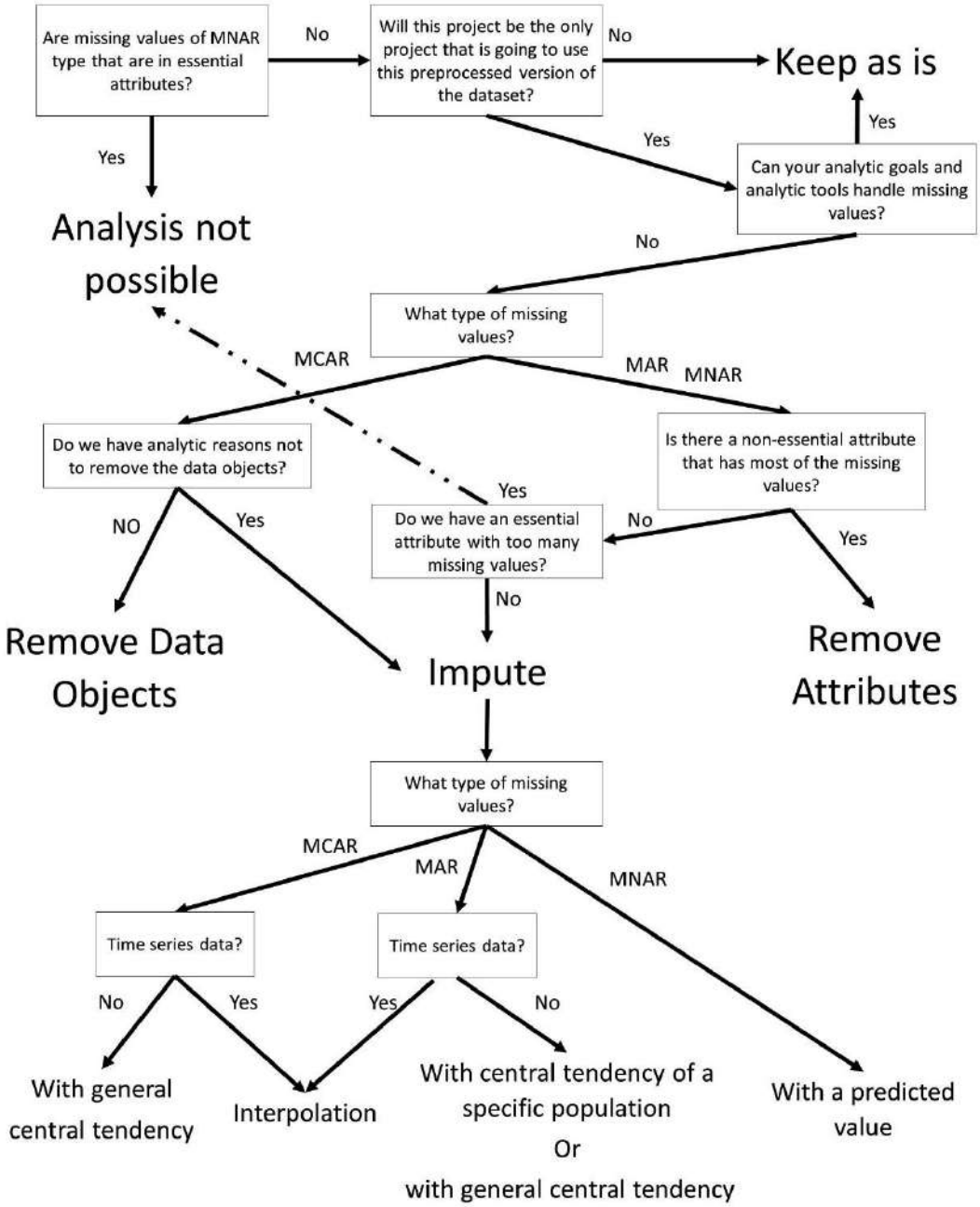
p-value of t-test: 1.3126894108159327e-85

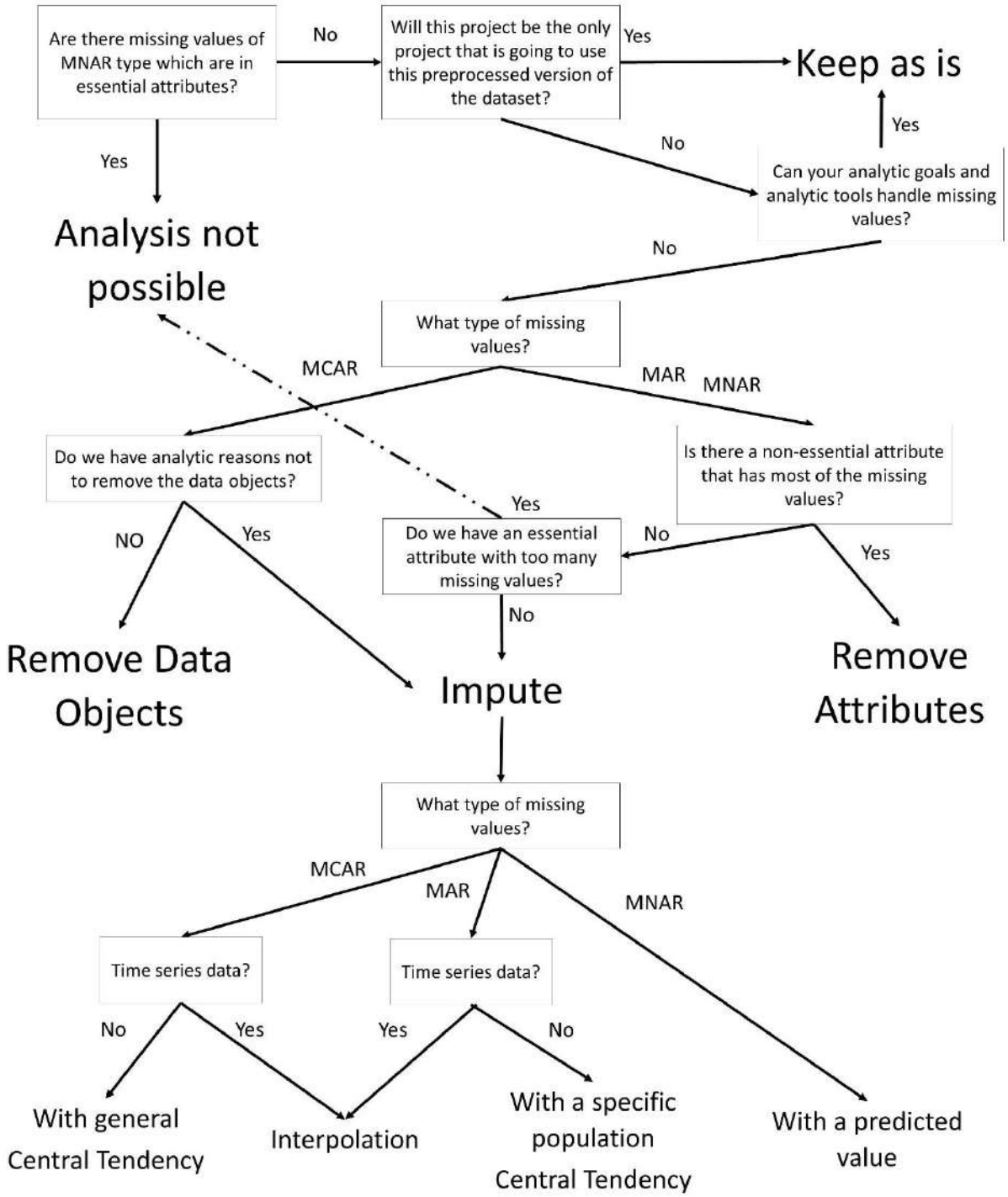
Diagnosis Analysis for weekday:

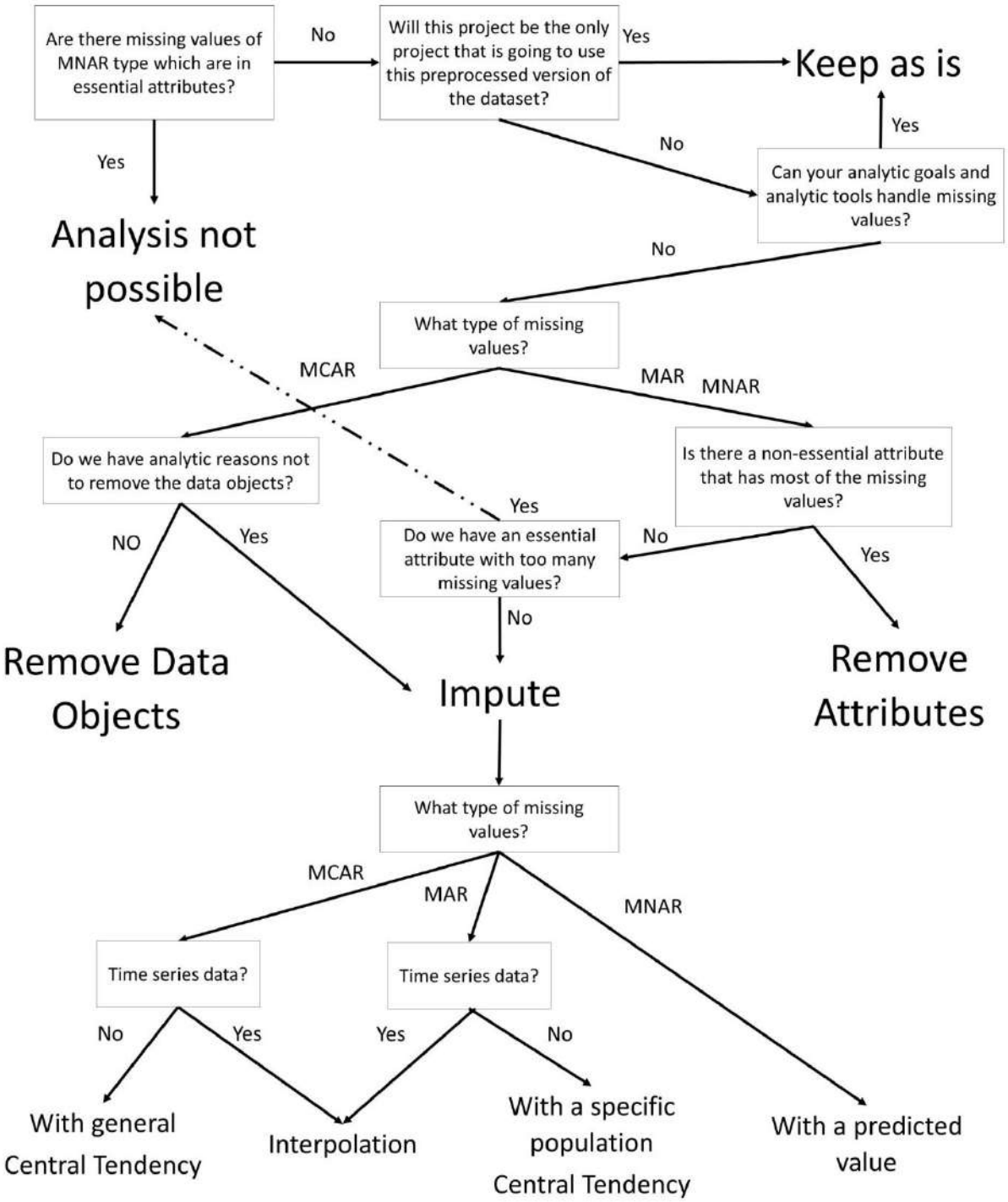


p-value of Chi_squared test: 1.554165460861991e-171


Are there missing values of MNAR type which are in essential attributes?
No
Will this project be the only project that is going to use this preprocessed version of the dataset?
Yes
Keep as is
Yes
Analysis not possible
No
Can your analytic goals and analytic tools handle missing values?
Yes
No
What type of missing values?
MCAR
MAR
MNAR
Do we have analytic reasons not to remove the data objects?
Is there a non-essential attribute that has most of the missing values?
Yes
Do we have an essential attribute with too many missing values?
NO
Yes
No
Yes
No
Remove Data Objects
Impute
Remove Attributes
What type of missing values?
MCAR
MAR
MNAR
Time series data?
Time series data?
No
Yes
Yes
No
With general Central Tendency
Interpolation
With a specific population Central Tendency
With a predicted value

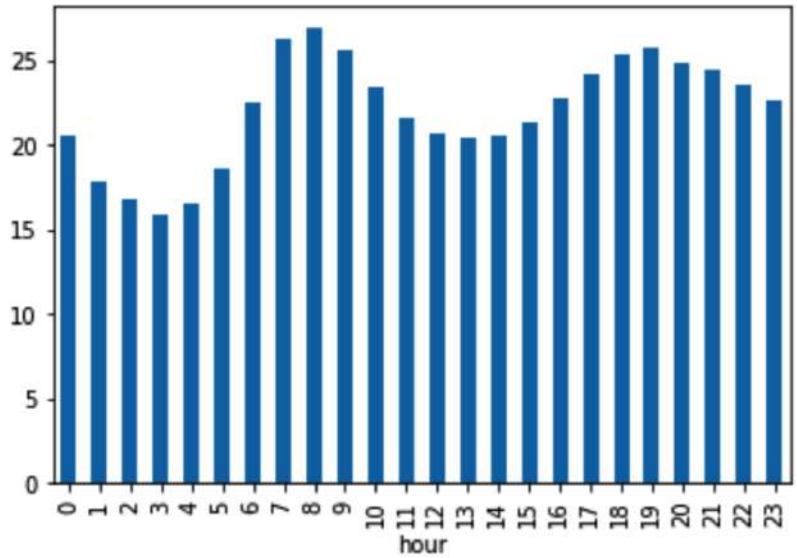

Are missing values of MNAR type that are in essential attributes?
No
Will this project be the only project that is going to use this preprocessed version of the dataset?
No
Keep as is
Yes
Yes
Can your analytic goals and analytic tools handle missing values?
Yes
Analysis not possible
No
What type of missing values?
MCAR
MAR
MNAR
Do we have analytic reasons not to remove the data objects?
Is there a non-essential attribute that has most of the missing values?
Yes
NO
Yes
Do we have an essential attribute with too many missing values?
No
Yes
No
Remove Data Objects
Impute
Remove Attributes
What type of missing values?
MCAR
MAR
MNAR
Time series data?
Time series data?
No
Yes
Yes
No
With general central tendency
Interpolation
With central tendency of a specific population
Or
with general central tendency
With a predicted value

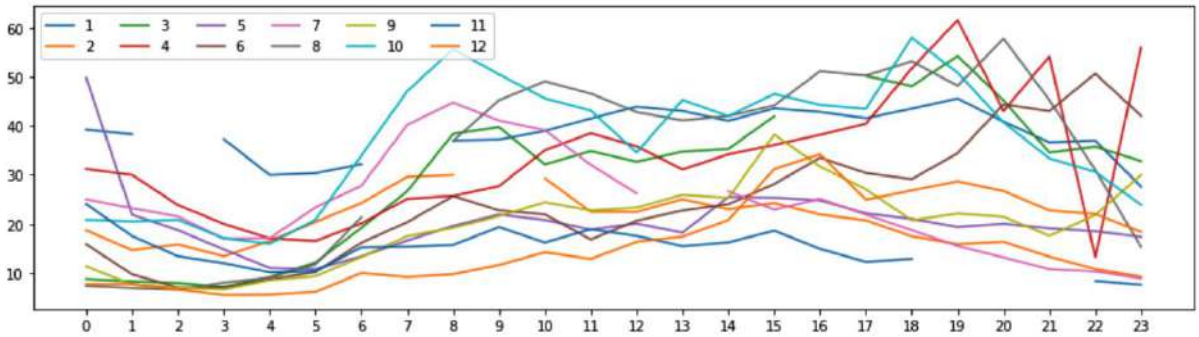

Are there missing values of MNAR type which are in essential attributes?
No
Will this project be the only project that is going to use this preprocessed version of the dataset?
Yes
Keep as is
Yes
Analysis not possible
No
Can your analytic goals and analytic tools handle missing values?
Yes
No
What type of missing values?
MCAR
MAR
MNAR
Do we have analytic reasons not to remove the data objects?
Is there a non-essential attribute that has most of the missing values?
Yes
Do we have an essential attribute with too many missing values?
NO
Yes
No
Yes
No
Remove Data Objects
Impute
Remove Attributes
What type of missing values?
MCAR
MAR
MNAR
Time series data?
Time series data?
No
Yes
Yes
No
With general Central Tendency
Interpolation
With a specific population Central Tendency
With a predicted value


Are there missing values of MNAR type which are in essential attributes?
No
Will this project be the only project that is going to use this preprocessed version of the dataset?
Yes
Keep as is
Yes
Analysis not possible
No
Can your analytic goals and analytic tools handle missing values?
Yes
No
What type of missing values?
MCAR
MAR
MNAR
Do we have analytic reasons not to remove the data objects?
Is there a non-essential attribute that has most of the missing values?
Yes
Do we have an essential attribute with too many missing values?
NO
Yes
No
Yes
No
Remove Data Objects
Impute
Remove Attributes
What type of missing values?
MCAR
MAR
MNAR
Time series data?
Time series data?
No
Yes
Yes
No
With general Central Tendency
Interpolation
With a specific population Central Tendency
With a predicted value

In [22]:

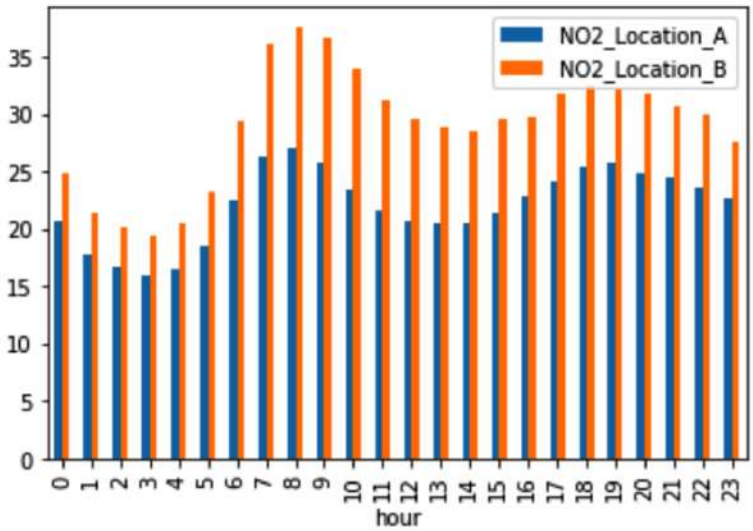
```
air_df.groupby('hour').NO2_Location_A.mean().plot.bar()
plt.show()
```





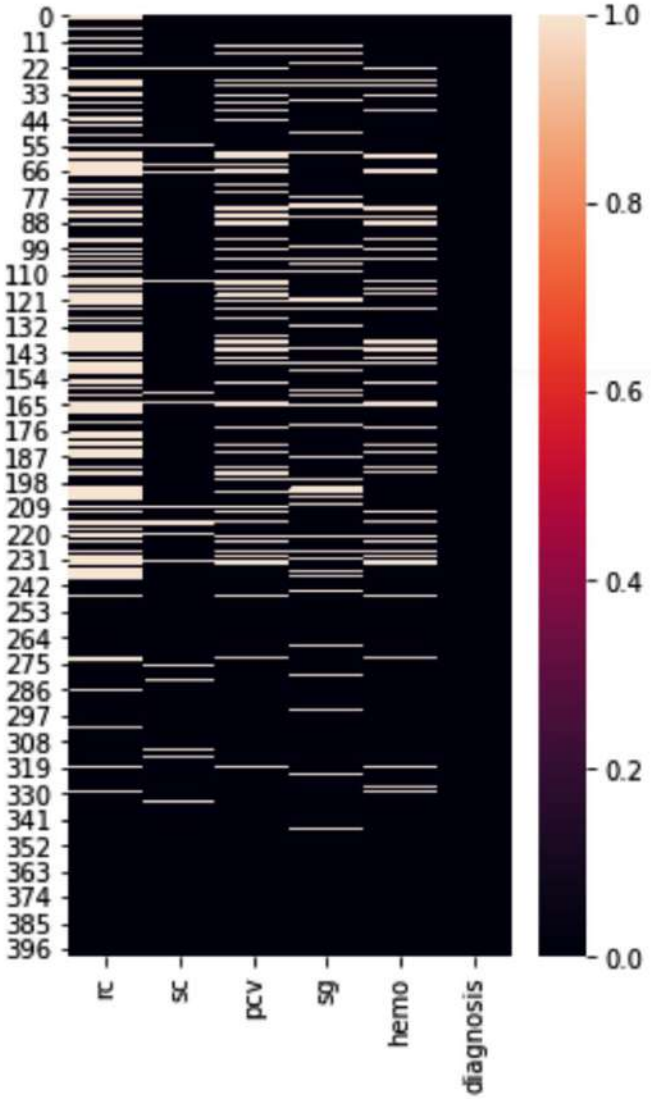
In [25]:

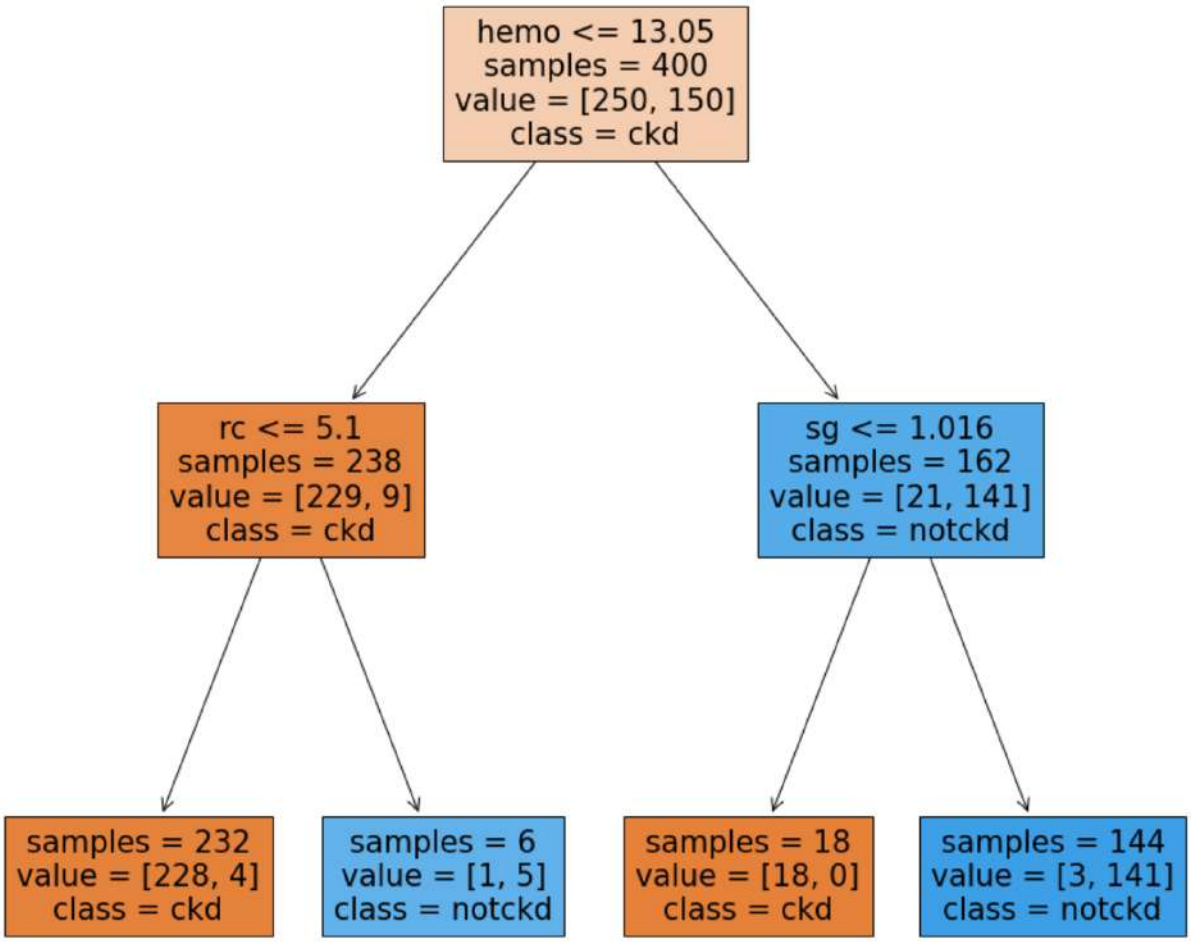
```
air_df.groupby('hour')[
    ['NO2_Location_A','NO2_Location_B']].mean().plot.bar()
plt.show()
```



In [29]:

```
patient_df = pd.read_csv('kidney_disease.csv')
plt.figure(figsize=(4,7))
sns.heatmap(patient_df.isna())
plt.show()
```





In [42]:
```
column_df = pd.read_csv('columns.csv')
column_df.head(2)
```

Out[42]:

	original	short
0	I enjoy listening to music.	Music
1	I prefer.	Slow songs or fast songs

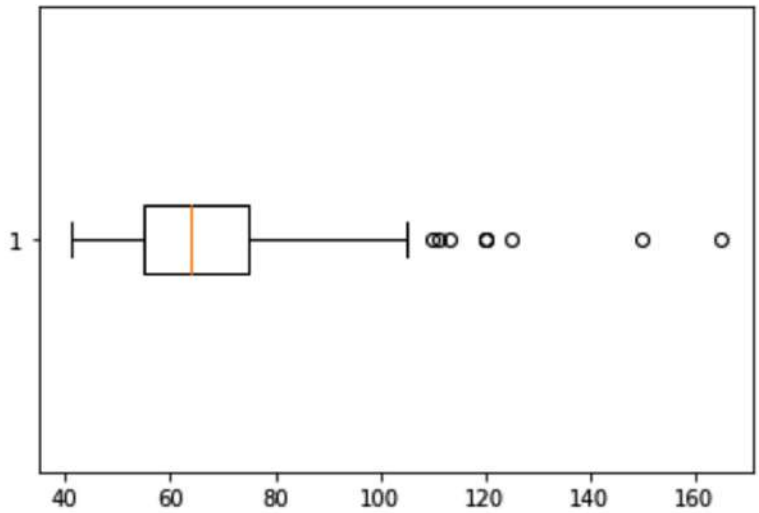
In [43]:
```
response_df = pd.read_csv('responses.csv')
response_df.head(2)
```

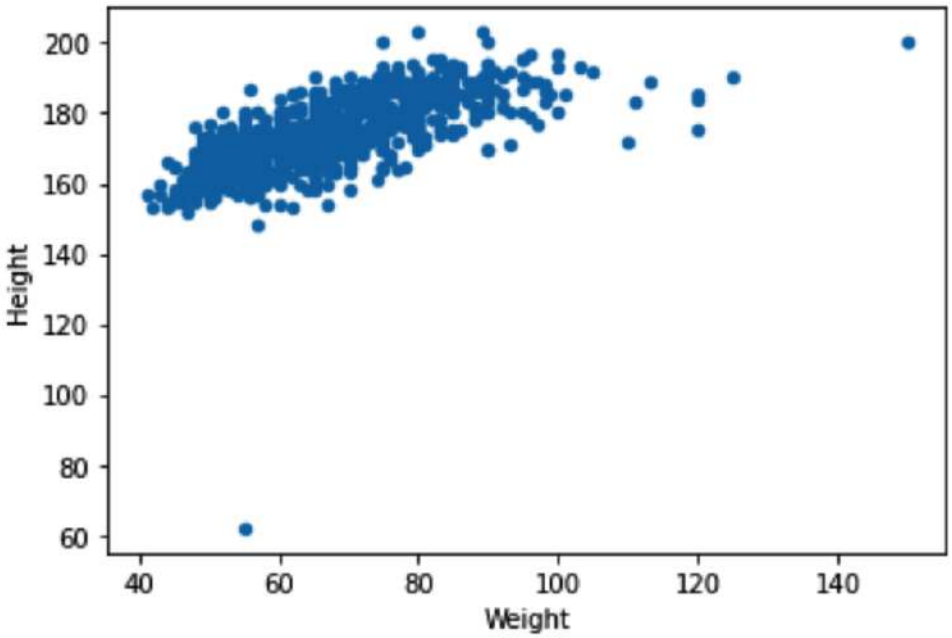
Out[43]:

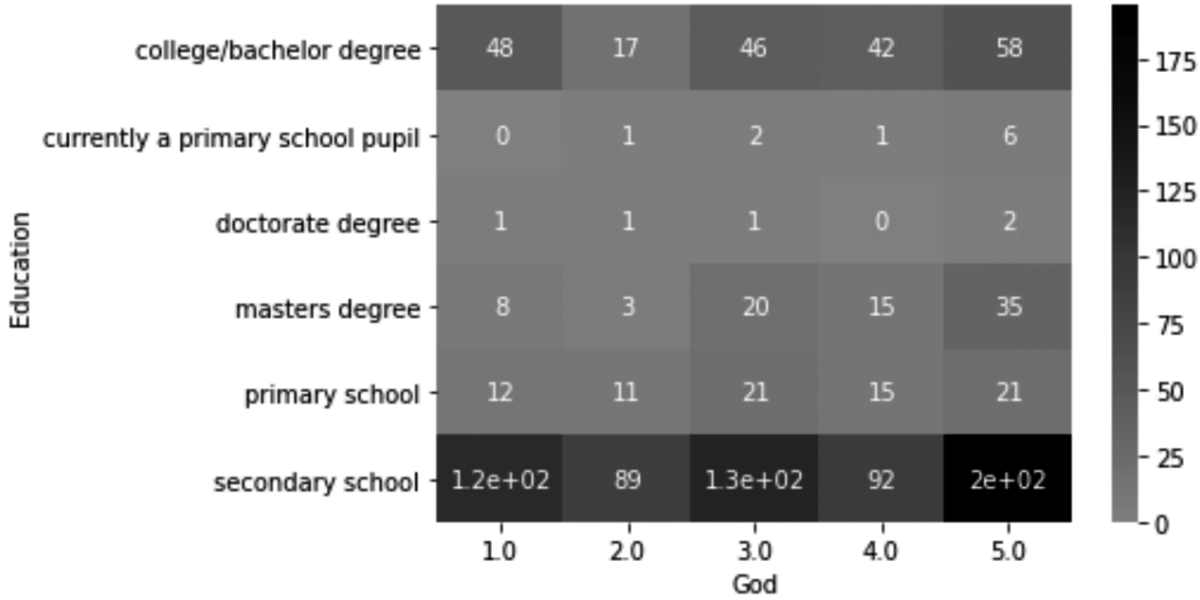
	Music	Slow songs or fast songs	Dance	Folk	Country	Classical music	Musical	Pop	Rock	Metal or Hardrock	...	Age	H
0	5.0	3.0	2.0	1.0	2.0	2.0	1.0	5.0	5.0	1.0	...	20.0	
1	4.0	4.0	2.0	1.0	1.0	1.0	2.0	3.0	5.0	4.0	...	19.0	

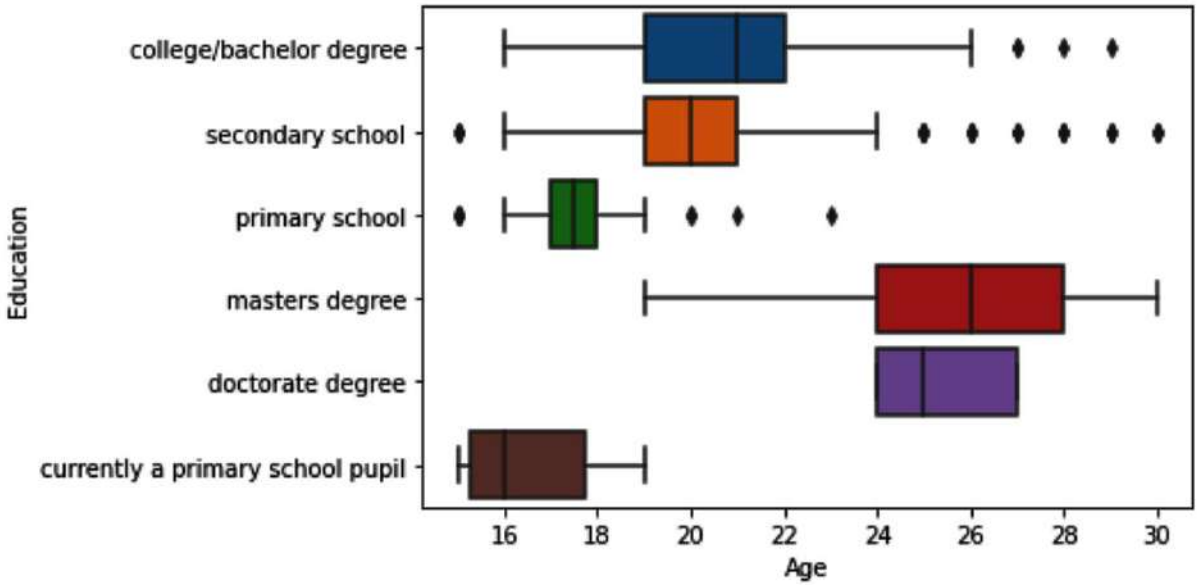
2 rows × 150 columns

In [44]:
```
fig = plt.boxplot(response_df.Weight.dropna(),vert=False)
```

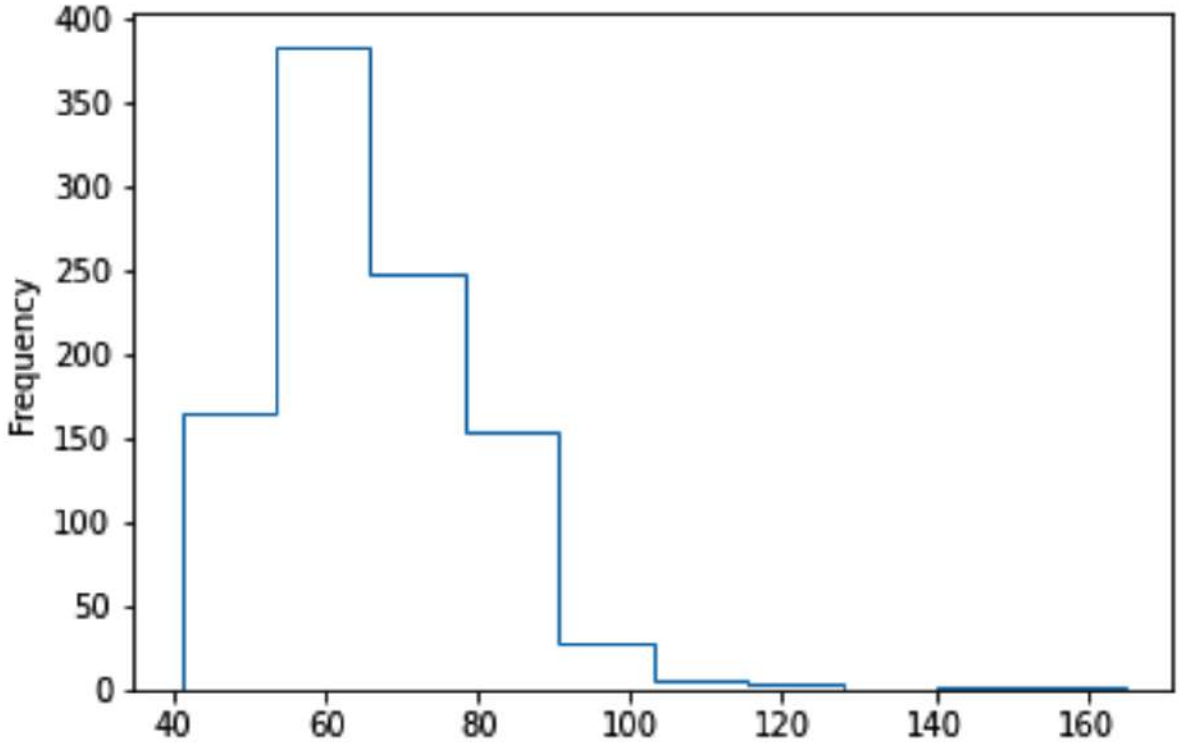


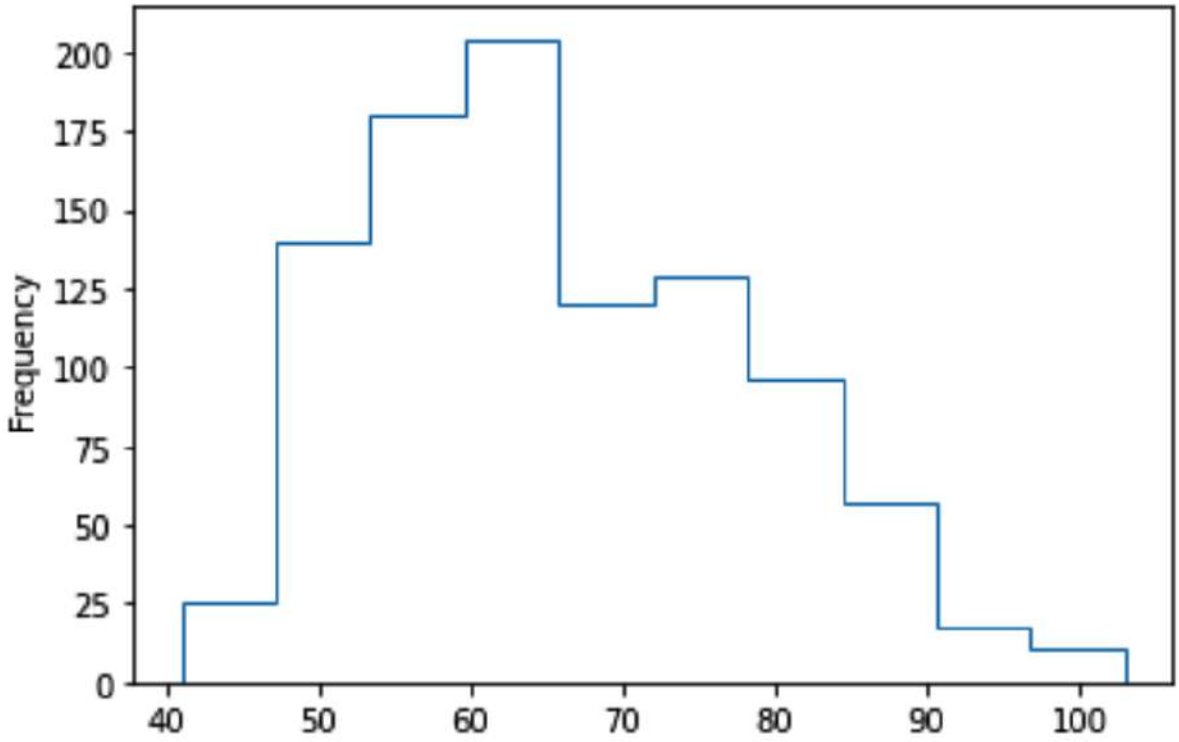






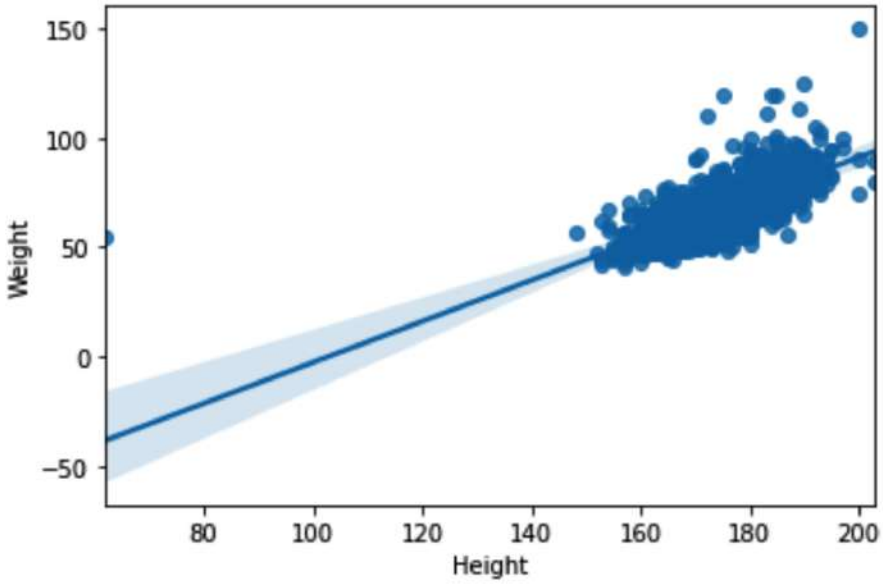
Analytic goals/tools	Prone to outliers	How best to deal with outliers?
Visualization: Summarizing a population/histogram	Yes	- Do nothing. - Remove data objects with outliers.
Visualization: Summarizing a population/boxplot	No	- Do nothing.
Visualization: Summarizing a population/bar chart	No	- Do nothing.
Visualization: Comparing populations	No	- Do nothing.
Visualization: The relationship between two attributes/scatterplot	Could be	- Do nothing. - Remove data objects with outliers. - Perform log transformation.
Visualization: The relationship between two attributes/contingency table	No	- Do nothing.
Visualization: Adding visual dimensions/adding size and color	Yes	- Replace with the upper cap or lower cap.
Visualization: Visualizing and comparing trends/line plots	No	- Do nothing.
Prediction: Regression	Yes	- Remove data objects with outliers. - Replace with the upper cap or lower cap.
Prediction: MLP	No	- Do nothing.
Classification: Decision Tree	No	- Do nothing.
Classification: KNN	Yes	- Replace with the upper cap or lower cap.
Clustering: K-Means	Could be	- Do nothing. - Replace with the upper cap or lower cap.


Frequency
400
350
300
250
200
150
100
50
0
40
60
80
100
120
140
160


Frequency
200
175
150
125
100
75
50
25
0
40
50
60
70
80
90
100

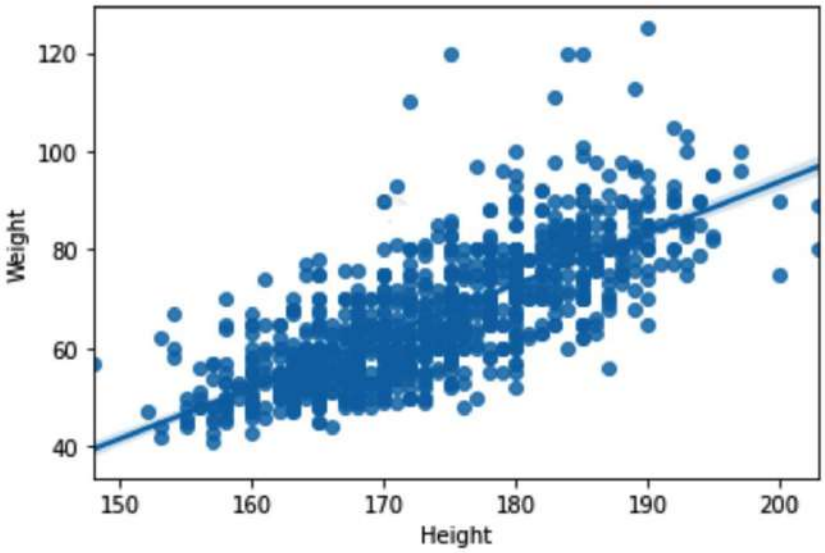
In [70]:

```
sns.regplot(x='Height',
            y='Weight',data=response_df)
plt.show()
```



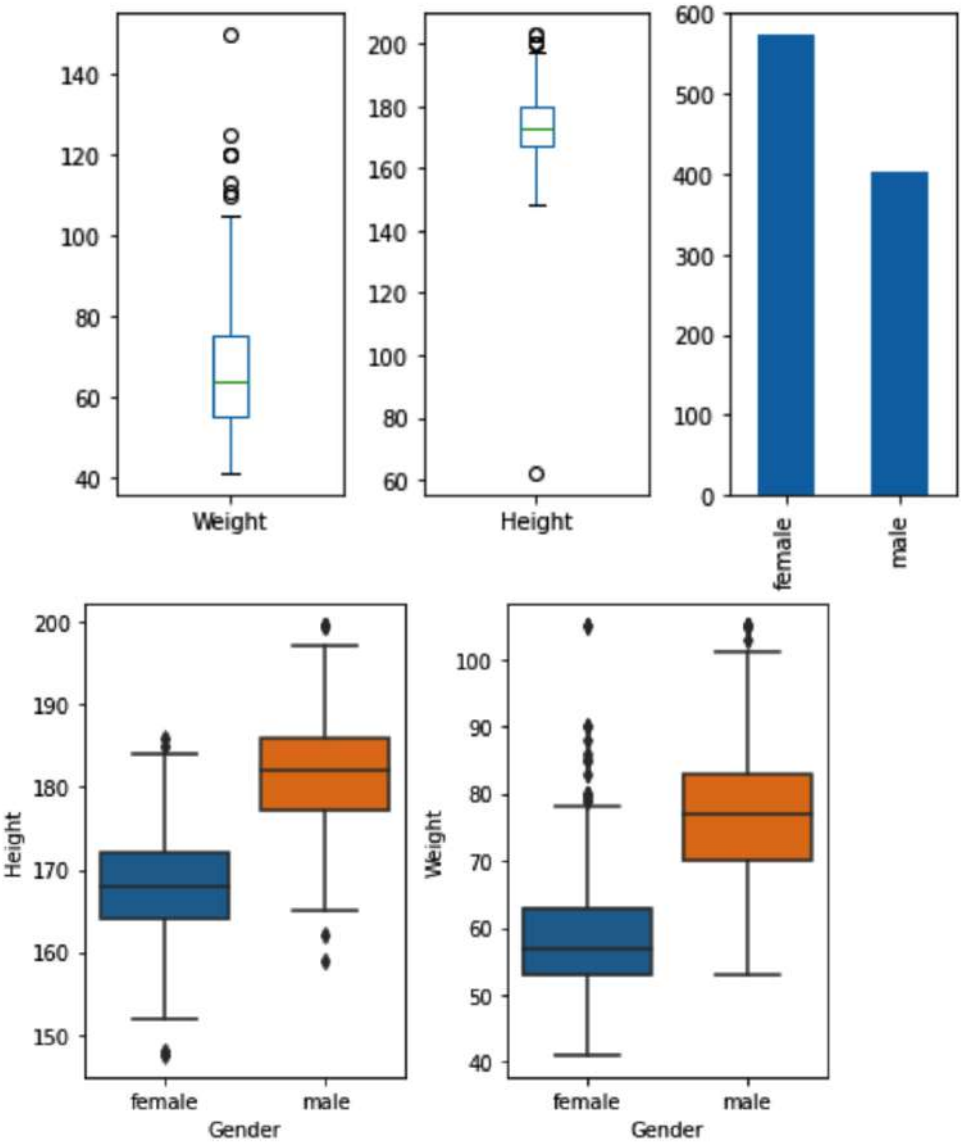
In [71]:

```
BM = (response_df.Weight>130) | (response_df.Height<70)
sns.regplot(x='Height',
            y='Weight',data=response_df[~BM])
plt.show()
```



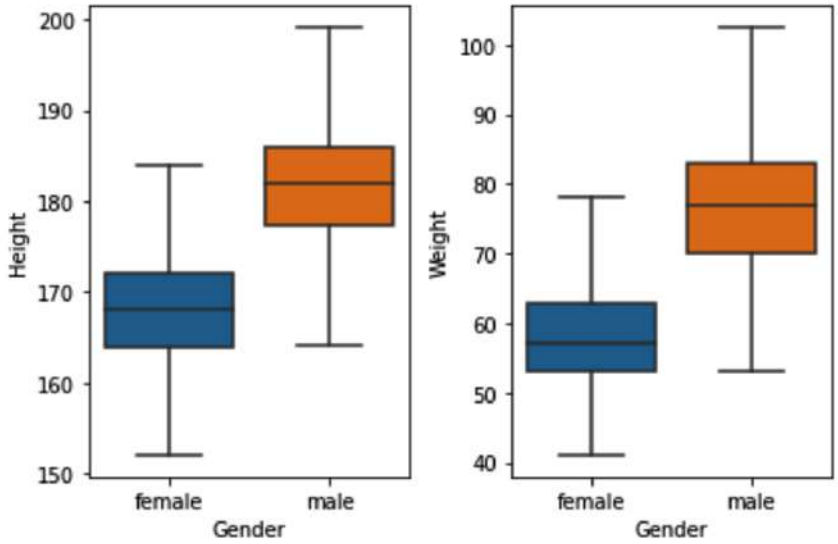
In [75]:

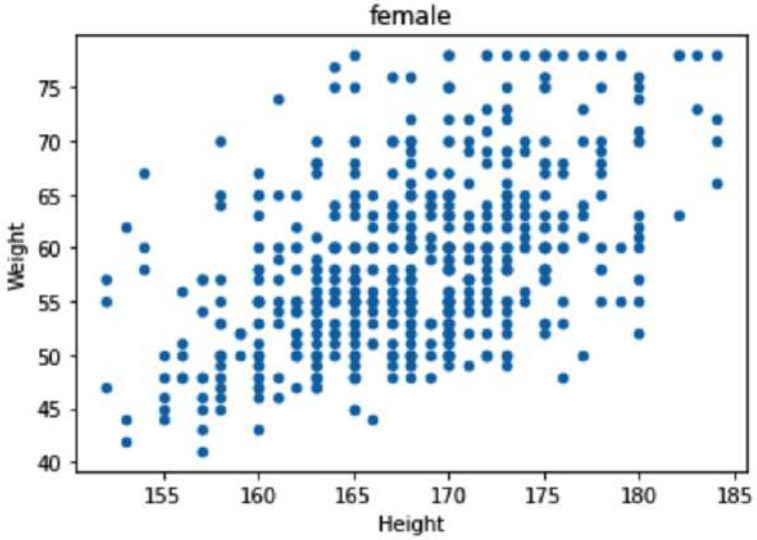
```
num_attributes = ['Weight','Height']
for i,att in enumerate(num_attributes):
    plt.subplot(1,3,i+1)
    pre_process_df[att].plot.box()

plt.subplot(1,3,3)
pre_process_df.Gender.value_counts().plot.bar()
plt.tight_layout()
plt.show()
```

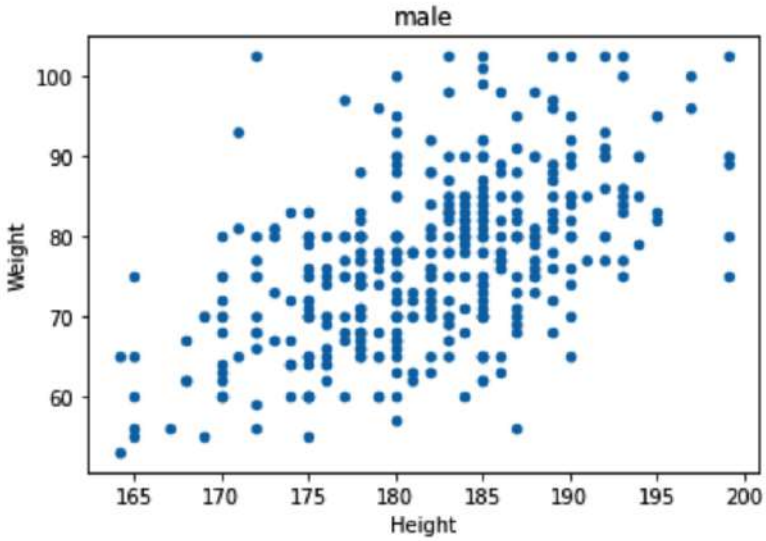


In [84]:

```
plt.subplot(1,2,1)
sns.boxplot(y=pre_process_df.Height,x=pre_process_df.Gender)
plt.subplot(1,2,2)
sns.boxplot(y=pre_process_df.Weight, x=pre_process_df.Gender)
plt.tight_layout()
```



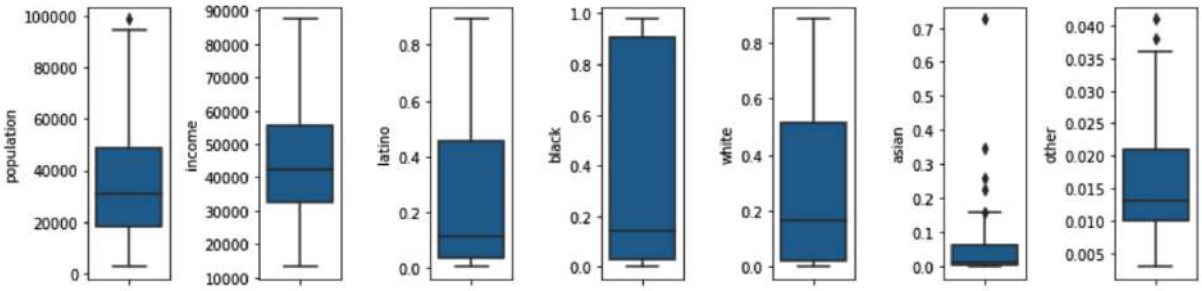




In [88]:

```
print('intercept (b0) ', lm.intercept_)
coef_names = ['b1','b2']
print(pd.DataFrame({'Predictor': data_X.columns,
                    'coefficient Name':coef_names,
                    'coefficient Value': lm.coef_}))
```

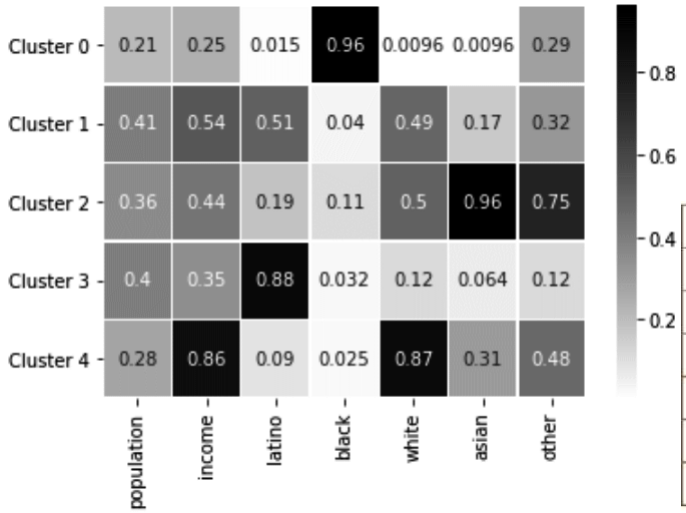
```
intercept (b0)  -51.10382582783839
  Predictor coefficient Name  coefficient Value
0    Height               b1           0.704025
1    Gender               b2          -8.602017
```



In [97]:

```
for i in range(5):
    BM = kmeans.labels_==i
    print('Cluster {}: {}'.format(i,pre_process_df[BM].index.values))
```

```
Cluster 0: ['Armour Square' 'Douglas' 'McKinley Park' 'Bridgeport']
Cluster 1: ['Montclare' 'Belmont Cragin' 'Hermosa' 'Avondale' 'Logan Squar
e'
 'Humboldt Park' 'South Lawndale' 'Lower West Side' 'East Side'
 'Hegewisch' 'Archer Heights' 'Brighton Park' 'New City' 'West Elsdon'
 'Gage Park' 'Clearing' 'West Lawn' 'Chicago Lawn' 'Ashburn']
Cluster 2: ['Rogers Park' 'West Ridge' 'Uptown' 'Lincoln Square' 'North Par
k'
 'Albany Park' 'Irving Park' 'Near West Side' 'Loop' 'Near South Side'
 'Hyde Park' 'Edgewater']
Cluster 3: ['Austin' 'West Garfield Park' 'East Garfield Park' 'North Lawnd
ale'
 'Oakland' 'Fuller Park' 'Grand Boulevard' 'Kenwood' 'Washington Park'
 'Woodlawn' 'South Shore' 'Chatham' 'Avalon Park' 'South Chicago'
 'Burnside' 'Calumet Heights' 'Roseland' 'Pullman' 'South Deering'
 'West Pullman' 'Riverdale' 'West Englewood' 'Englewood'
 'Greater Grand Crossing' 'Auburn Gresham' 'Washington Heights'
 'Morgan Park']
Cluster 4: ['North Center' 'Lake View' 'Lincoln Park' 'Near North Side' 'Ed
ison Park'
 'Norwood Park' 'Jefferson Park' 'Forest Glen' 'Portage Park' 'Dunning'
 'West Town' 'Garfield Ridge' 'Beverly' 'Mount Greenwood' "O'Hare"]
```



Thermometer 1	70.16
Thermometer 2	69.94
Thermometer 3	70.35
Thermometer 4	69.83
Thermometer 5	70.01
Thermometer 6	70.38
Thermometer 7	70.12

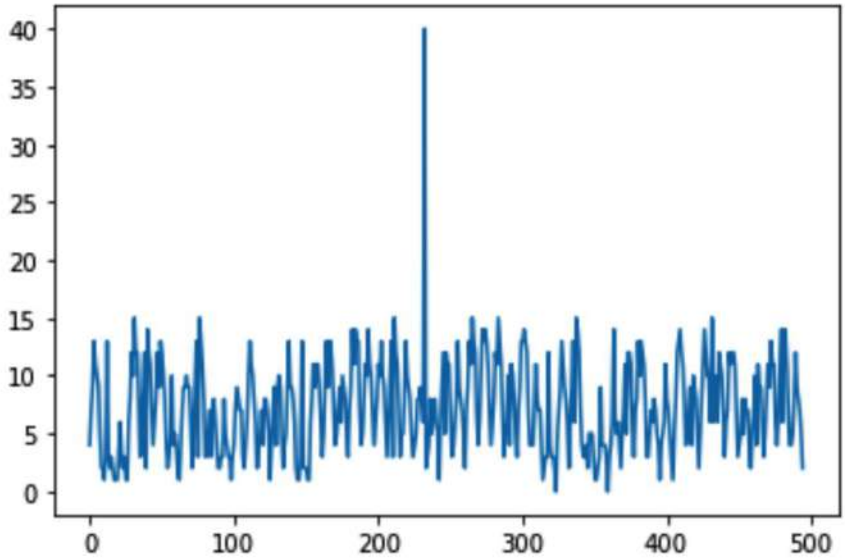
In [99]:

```
hour_df = pd.read_excel('CustomerEnteries.xlsx')
hour_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 495 entries, 0 to 494
Data columns (total 3 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   Date         495 non-null    datetime64[ns]
 1   Time         495 non-null    int64
 2   N_Cusotmers  495 non-null    int64
dtypes: datetime64[ns](1), int64(2)
memory usage: 11.7 KB
```

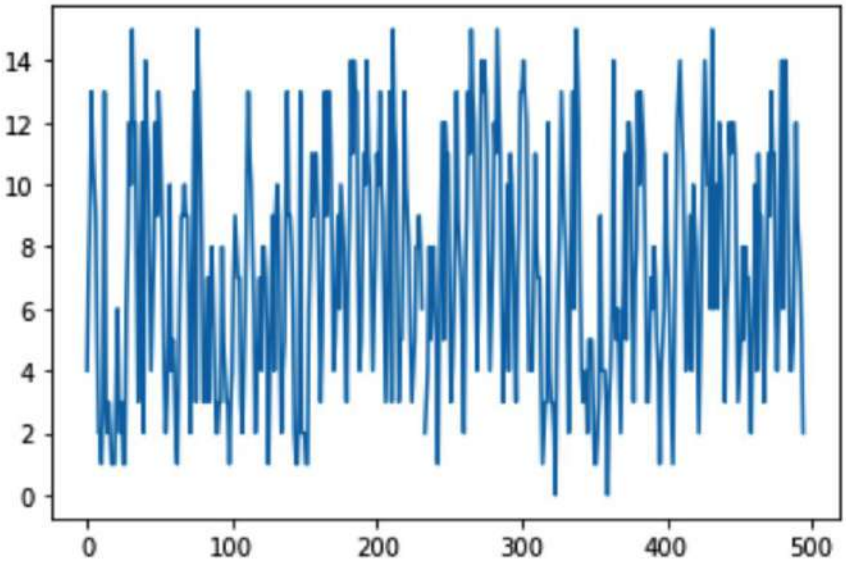
In [100]:

```
hour_df.N_Customers.plot()
plt.show()
```



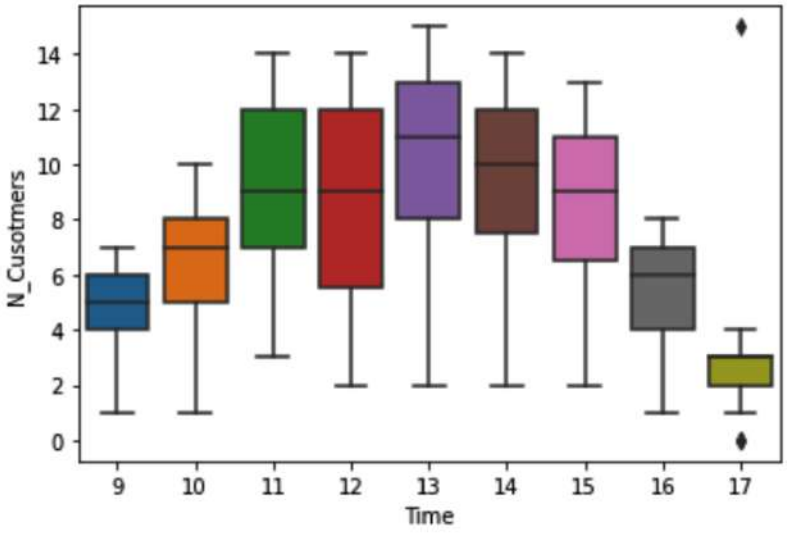
In [103]:

```
hour_df.N_Customers.plot()
plt.show()
```

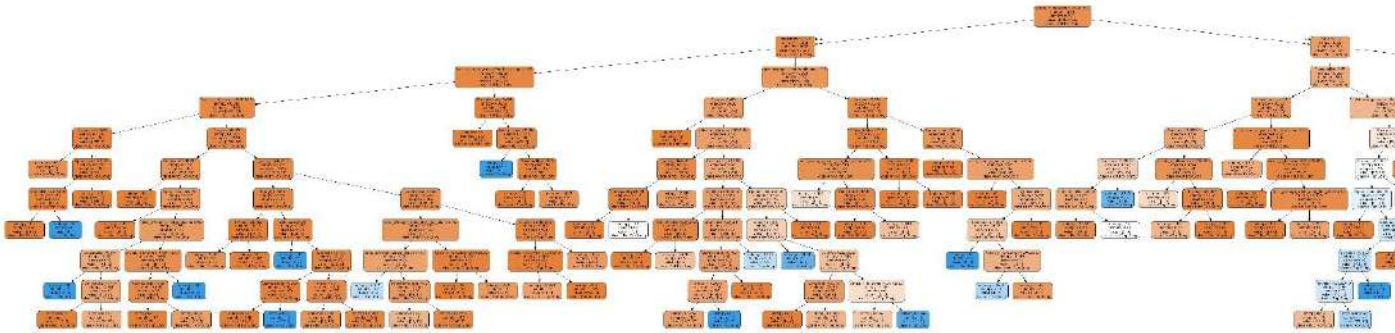


In [104]:

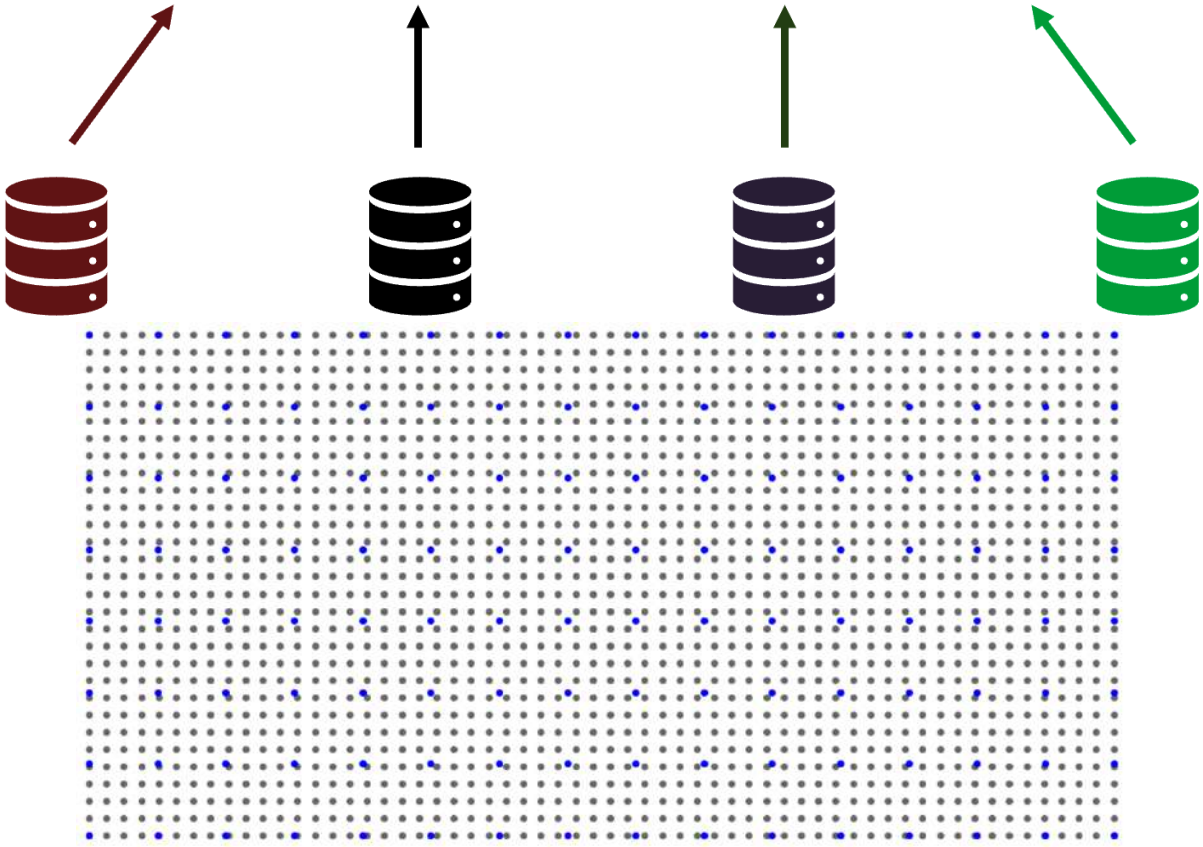
```
sns.boxplot(y=hour_df.N_Customers,x=hour_df.Time)
plt.show()
```



Chapter 12: Data Fusion and Data Integration



	Column 1	Column 2	.	.	.	Column m
Row 1	xxx	xxx	.	.	.	xxx
Row 2	xxx	xxx	.	.	.	xxx
.	.	.	.	.	.	.
.	.	.	.	.	.	.
.	.	.	.	.	.	.
Row n	xxx	xxx	.	.	.	xxx



temp_df

Date	Time	temp
2016-01-01	00:00:00	79.0
	01:00:00	79.0
	02:00:00	77.5
	03:00:00	79.0
	04:00:00	77.0
...	...	...
2016-12-31	19:00:00	79.0
	20:00:00	77.0
	21:00:00	77.0
	22:00:00	77.0
	23:00:00	77.0

+

electric_df

Date	Time	Consumption
1/1/2016	0:00	119052.0
	10:00	101915.0
	11:00	105796.0
	12:00	109960.0
	13:00	112189.0
...	...	...
9/9/2016	5:00	77492.0
	6:00	84960.0
	7:00	94969.0
	8:00	99795.0
	9:00	104091.0

→ ✗ Error
No overlap

Date	Time	temp
2016-01-01	00:00	79.0
	01:00	79.0
	02:00	77.5
	03:00	79.0
	04:00	77.0
...	...	...
2016-12-31	19:00	79.0
	20:00	77.0
	21:00	77.0
	22:00	77.0
	23:00	77.0

+

Date	Time	Consumption
2016-01-01	0:00	119052.0
	10:00	101915.0
	11:00	105796.0
	12:00	109960.0
	13:00	112189.0
...	...	...
2016-12-31	5:00	128275.0
	6:00	130920.0
	7:00	134707.0
	8:00	139168.0
	9:00	143965.0

→ ✗

Date	Time	temp	Consumption
2016-01-01	00:00	79.0	NaN
	01:00	79.0	NaN
	02:00	77.5	NaN
	03:00	79.0	NaN
	04:00	77.0	NaN
...	...	...	...
2016-12-31	19:00	79.0	154958.0
	20:00	77.0	149484.0
	21:00	77.0	143693.0
	22:00	77.0	142717.0
	23:00	77.0	150928.0

Date	Time	temp
2016-01-01	00:00	79.0
	01:00	79.0
	02:00	77.5
	03:00	79.0
	04:00	77.0
...	...	...
2016-12-31	19:00	79.0
	20:00	77.0
	21:00	77.0
	22:00	77.0
	23:00	77.0

+

Date	Time	Consumption
2016-01-01	00:00	119052.0
	01:00	113138.0
	02:00	111013.0
	03:00	104808.0
	04:00	99552.0
...	...	...
2016-12-31	19:00	154958.0
	20:00	149484.0
	21:00	143693.0
	22:00	142717.0
	23:00	150928.0

→ ✓

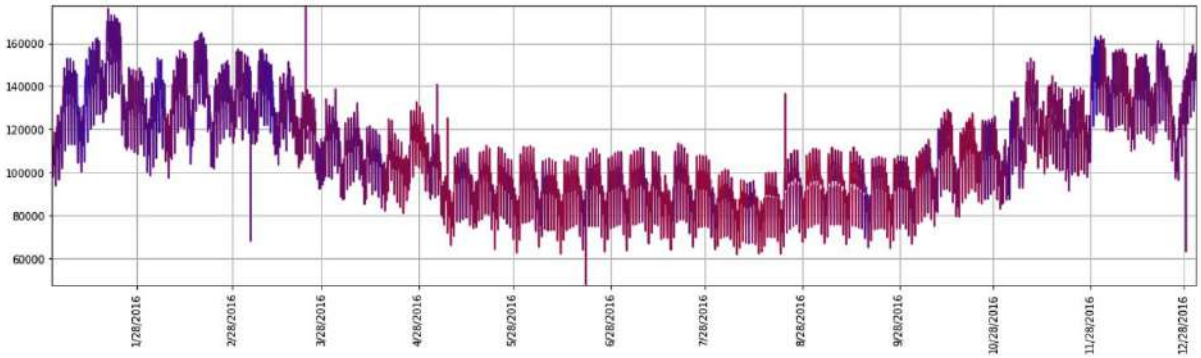
Date	Time	temp	Consumption
2016-01-01	00:00	79.0	119052.0
	01:00	79.0	113138.0
	02:00	77.5	111013.0
	03:00	79.0	104808.0
	04:00	77.0	99552.0
...	...	...	...
2016-12-31	19:00	79.0	154958.0
	20:00	77.0	149484.0
	21:00	77.0	143693.0
	22:00	77.0	142717.0
	23:00	77.0	150928.0

temp_df

	Timestamp	temp
0	2016-01-01T00:00:00	79.0
1	2016-01-01T01:00:00	79.0
2	2016-01-01T02:00:00	77.5
3	2016-01-01T03:00:00	79.0
4	2016-01-01T04:00:00	77.0
...	...	...
8726	2016-12-31T19:00:00	79.0
8727	2016-12-31T20:00:00	77.0
8728	2016-12-31T21:00:00	77.0
8729	2016-12-31T22:00:00	77.0
8730	2016-12-31T23:00:00	77.0

electric_df

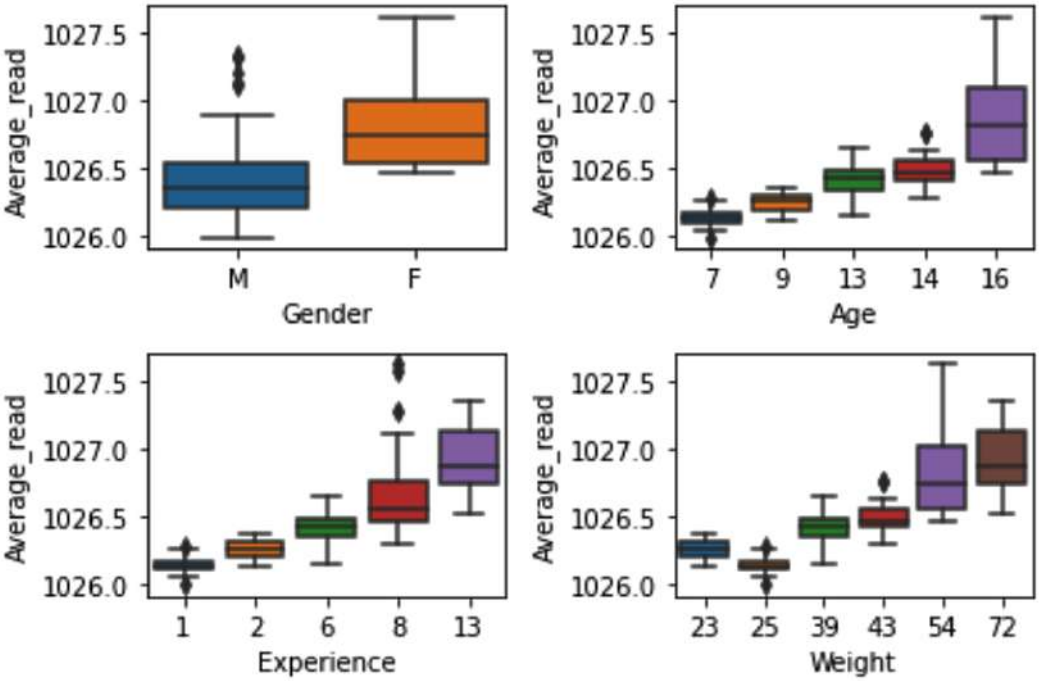
	Date	Time	Consumption
0	12/1/2017	0:00:00	72650.0
1	12/1/2017	0:30:00	70553.0
2	12/1/2017	1:00:00	68277.0
3	12/1/2017	1:30:00	67611.0
4	12/1/2017	2:00:00	67388.0
...	...	...	...
19051	1/1/2016	21:30	56059.0
19052	1/1/2016	22:00	55107.0
19053	1/1/2016	22:30	55609.0
19054	1/1/2016	23:00	58199.0
19055	1/1/2016	23:30	57539.0

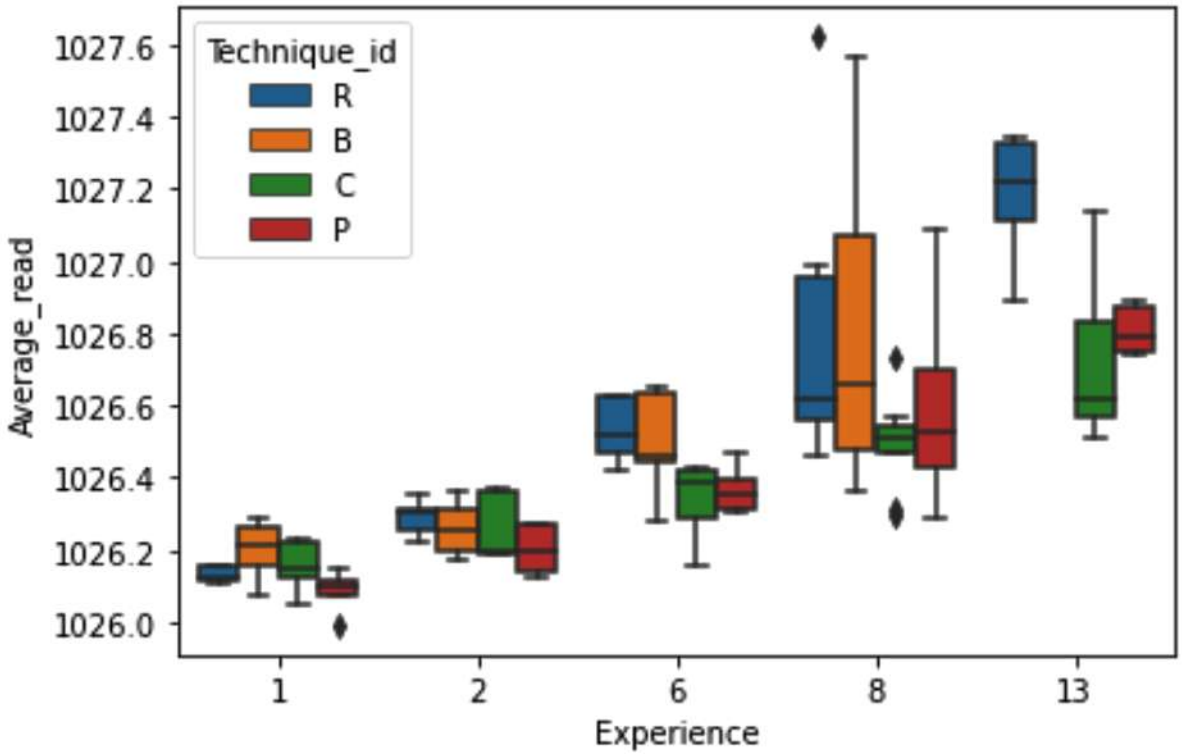


	Participant_id	Gender	Age	Weight	Experience	Technique_id	Trial_number	Average_read
0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...
115	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
116	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
117	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
118	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
119	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

In [23]:

```
techniques = ['R','B','C','P']
index = 0
for col in unknown_df.columns:
    if(col[0] in techniques):
        performance_df.loc[index,'Technique_id'] = col[0]
        performance_df.loc[index,'Trial_number'] = unknown_df[col][1]

        P_id = unknown_df[col][0]
        performance_df.loc[index,'Participant_id'] = P_id
        performance_df.loc[index,'Gender'] = athlete_df.loc[P_id].Sex
        performance_df.loc[index,'Age'] = athlete_df.loc[P_id].Age
        performance_df.loc[index,'Weight'] = athlete_df.loc[P_id].Weight
        performance_df.loc[
            index,'Experience'] = athlete_df.loc[P_id].Experience
        BM = unknown_df[col][2:].isna()
        performance_df.loc[
            index,'Average_read'] = unknown_df[
            col][2:][~BM].astype(int).mean()
        index +=1
```





In [27]:
```
billboard_df.query("Artists == '50 Cent' and Name=='Outta Control' \
                   and Week== '2005-09-14'")
```

Out[27]:

	Unnamed: 0	Artists	Name	Weekly.rank	Peak.position	Weeks.on.chart	Week	Date
67588	67589	50 Cent	Outta Control	25	25.0	9.0	2005-09-14	August 6, 2005
67647	67648	50 Cent	Outta Control	92	NaN	NaN	2005-09-14	August 6, 2005

In [34]:

```
songAttribute_df = pd.read_csv('songAttributes_1999-2019.csv')
wsr = songAttribute_df.apply(lambda r: '{}---{}'
                                .format(r.Name,r.Artist),axis=1)
doFrequencies = wsr.value_counts()

BM = doFrequencies>1
n_totalSongs = sum(BM)
print('Total processings: ' + str(n_totalSongs))

t = time.time()
i_progress = 0
for i,v in doFrequencies[BM].iteritems():
    [name,artist] = i.split('---')
    BM = ((songAttribute_df.Name == name) &
          (songAttribute_df.Artist == artist))

    wdf = songAttribute_df[BM]
    dropping_index = wdf.index[1:]
    songAttribute_df.drop(index = dropping_index, inplace=True)

    i_progress +=1
    if(i_progress%500==0):
        print('Processed: ' + str(i_progress))
        process_time = time.time() - t
        print('Elapsed: ' + str(round(process_time,1)) + ' s')
        estimate_finish = round((n_totalSongs-i_progress) *
                                (process_time/500)/60,1)

        print('To finish: ' + str(estimate_finish)+ 'mins')
        t = time.time()
        print('-----------------------------------')
```

In [37]:

```
artist_df.query("Artist == 'Reba McEntire'")
```

Out[37]:

	X	Artist	Followers	Genres	NumAlbums	YearFirstAlbum	Gender	Gr
398	398	Reba McEntire	974392	contemporary country,country,country dawn	40	1977	F	
716	716	Reba McEntire	974392	contemporary country,country,country dawn	40	1977	F	

In [39]:

```
songIntegrate_df = pd.DataFrame(
    columns = ['Name', 'Artists', 'Top_song', 'First_date_on_Billboard',
               'Acousticness', 'Danceability', 'Duration', 'Energy',
               'Explicit', 'Instrumentalness', 'Liveness', 'Loudness',
               'Mode', 'Speechiness', 'Tempo', 'TimeSignature', 'Valence',
               'Artists_n_followers', 'n_male_artists', 'n_female_artists',
               'n_bands', 'artist_average_years_after_first_album',
               'artist_average_number_albums'])
songIntegrate_df
```

Out[39]:

Name	Artists	Top_song	First_date_on_Billboard	Acousticness	Danceability	Duration	Ene

0 rows × 23 columns

Situations	Description	Example	
Situation 1	- Songs with only one artist - Songs with unique song names	songIntegrate_df	Artists Name 16 Taylor Swift You Need To Calm Down
		songAttribute_df	Artist Name 154047 Taylor Swift You Need To Calm Down
Situation 2	- Songs with only one artist - Songs with non-unique song names To see the difference between situations 1 and 2, run and compare the following code: - songAttribute_df.query("Name == 'Sucker'") - songAttribute_df.query("Name == 'You Need To Calm Down'")	songIntegrate_df	Artists Name 9 Jonas Brothers Sucker
		songAttribute_df	Artist Name 21644 New Found Glory Sucker 154557 Jonas Brothers Sucker
Situation 3	- Songs with more than one artist - Both artists are recognized in both sources but in different ways	songIntegrate_df	Artists Name 6 Ed Sheeran, Justin Bieber I Don't Care
		songAttribute_df	Artist Name 154921 Ed Sheeran I Don't Care (with Justin Bieber)
Situation 4	- Songs with more than one artist but only songAttribute_df recognizes the second artist	songIntegrate_df	Artists Name 12 Chris Brown No Guidance
		songAttribute_df	Artist Name 154214 Chris Brown No Guidance (feat. Drake)
Situation 5	Songs with more than one artist but only songIntegrate_df recognizes the second artist	songIntegrate_df	Artists Name 137 DJ Sammy, Yanou Heaven
		songAttribute_df	Artist Name 22487 DJ Sammy Heaven

In [42]:

```
adding_columns = ['Acousticness','Danceability','Duration','Energy','Explicit','Instrumentalness',
                  'Liveness','Loudness','Mode','Speechiness','Tempo','TimeSignature', 'Valence']
template = 'Index= {} - The song {} by {} was integrated using sitution {}.'
for i, row in songIntegrate_df.iterrows():
    filled = False
    Artists = row.Artists.split(',')
    Artists = list(map(str.strip,Artists))
    # Situation 1
    BM = songAttribute_df.Name == row.Name
    if(sum(BM) == 1):
        for col in adding_columns:
            songIntegrate_df.loc[i,col]= songAttribute_df[BM][col].values[0]
        filled = True
        print(template.format(i,row.Name,row.Artists,1))
    # Situation 2
    elif(sum(BM) > 1):
        wdf = songAttribute_df[BM]
        if(len(Artists)==1):
            BM2 = wdf.Artist.str.contains(Artists[0])
            if(sum(BM2)==1):
                for col in adding_columns:
                    songIntegrate_df.loc[i,col]= wdf[BM2][col].values[0]
                filled = True
                print(template.format(i,row.Name,row.Artists,2))
    # Situation 3
    if((not filled) and len(Artists)>1):
        BM2= (songAttribute_df.Name.str.contains(row.Name)&songAttribute_df.Artist.isin(Artists))
        if(sum(BM2)==1):
            for col in adding_columns:
                songIntegrate_df.loc[i,col]= songAttribute_df[BM2][col].values[0]
            filled = True
            print(template.format(i,row.Name,row.Artists,3))
    if(not filled):
        # Situation 4
        BM2 = songAttribute_df.Name.str.contains(row.Name)
        if(sum(BM2)==1):
            for artist in Artists:
                if(artist == songAttribute_df[BM2].Artist.iloc[0]):
                    for col in adding_columns:
                        songIntegrate_df.loc[i,col]= songAttribute_df[BM2][col].values[0]
                    filled = True
                    print(template.format(i,row.Name,row.Artists,4))
        # Situation 5
        if(sum(BM2)>1):
            wdf2 = songAttribute_df[BM2]
            BM3 = wdf2.Artist.isin(Artists)
            if(sum(BM3)>0):
                wdf3 = wdf2[BM3]
                for i3, row3 in wdf3.iterrows():
                    if(row3.Name == row.Name):
                        for col in adding_columns:
                            songIntegrate_df.loc[i,col]= row3[col]
                        filled = True
                        print(template.format(i,row.Name,row.Artists,5))
```

In [45]:

```
B_MV = songIntegrate_df.Acousticness.isna()
B_MV.rename('Missing Values',inplace=True)
contigency_table = pd.crosstab(songIntegrate_df.Top_song,B_MV)
contigency_table
```

Out[45]:

Missing Values	False	True
Top_song		
False	3618	2874
True	427	294

In [46]:

```
from scipy.stats import chi2_contingency
p_value = chi2_contingency(contigency_table)[1]
p_value
```

Out[46]: 0.07952275342130063

In [52]:

```
for i,row in songIntegrate_df.iterrows():
    Artists = row.Artists.split(',')
    Artists = list(map(str.strip,Artists))
    ArtistsIn_artist_df = True
    for artist in Artists:
        if(artist not in artist_df.index.values):
            ArtistsIn_artist_df= False
            break
    if(not ArtistsIn_artist_df):
        continue

    songIntegrate_df.loc[i,'Artists_n_followers'] = 0
    songIntegrate_df.loc[i,'n_male_artists'] = 0
    songIntegrate_df.loc[i,'n_female_artists'] = 0
    songIntegrate_df.loc[i, 'artist_average_years_after_first_album'] = 0
    songIntegrate_df.loc[i, 'artist_average_number_albums'] = 0
    songIntegrate_df.loc[i,'n_bands'] = 0

    for artist in Artists:
        songIntegrate_df.loc[i,'Artists_n_followers'] += artist_df.loc[artist].Followers
        if(artist_df.loc[artist]['Group.Solo']=='Solo'):
            if(artist_df.loc[artist].Gender == 'M'):
                songIntegrate_df.loc[i,'n_male_artists'] += 1
            if(artist_df.loc[artist].Gender == 'F'):
                songIntegrate_df.loc[i,'n_female_artists'] += 1

        if(artist_df.loc[artist]['Group.Solo']=='Group'):
            if(artist_df.loc[artist].Gender == 'M'):
                songIntegrate_df.loc[i,'n_male_artists'] += 2
            if(artist_df.loc[artist].Gender == 'F'):
                songIntegrate_df.loc[i,'n_female_artists'] += 2
            songIntegrate_df.loc[i,'n_bands'] += 1
        First_date_on_Billboard = int(row.First_date_on_Billboard[:4])
        songIntegrate_df.loc[i, 'artist_average_years_after_first_album'] += \
            (First_date_on_Billboard - int(artist_df.loc[artist].YearFirstAlbum))

        songIntegrate_df.loc[i,
            'artist_average_number_albums'] += int(artist_df.loc[artist].NumAlbums)

    songIntegrate_df.loc[i,'artist_average_years_after_first_album'] /= len(Artists)
    songIntegrate_df.loc[i, 'artist_average_number_albums'] /= len(Artists)
```

In [54]:

```
B_MV = songIntegrate_df.Artists_n_followers.isna()
B_MV.rename('Missing Values',inplace=True)
contigency_table = pd.crosstab(songIntegrate_df.Top_song,B_MV)
contigency_table
```

Out[54]:

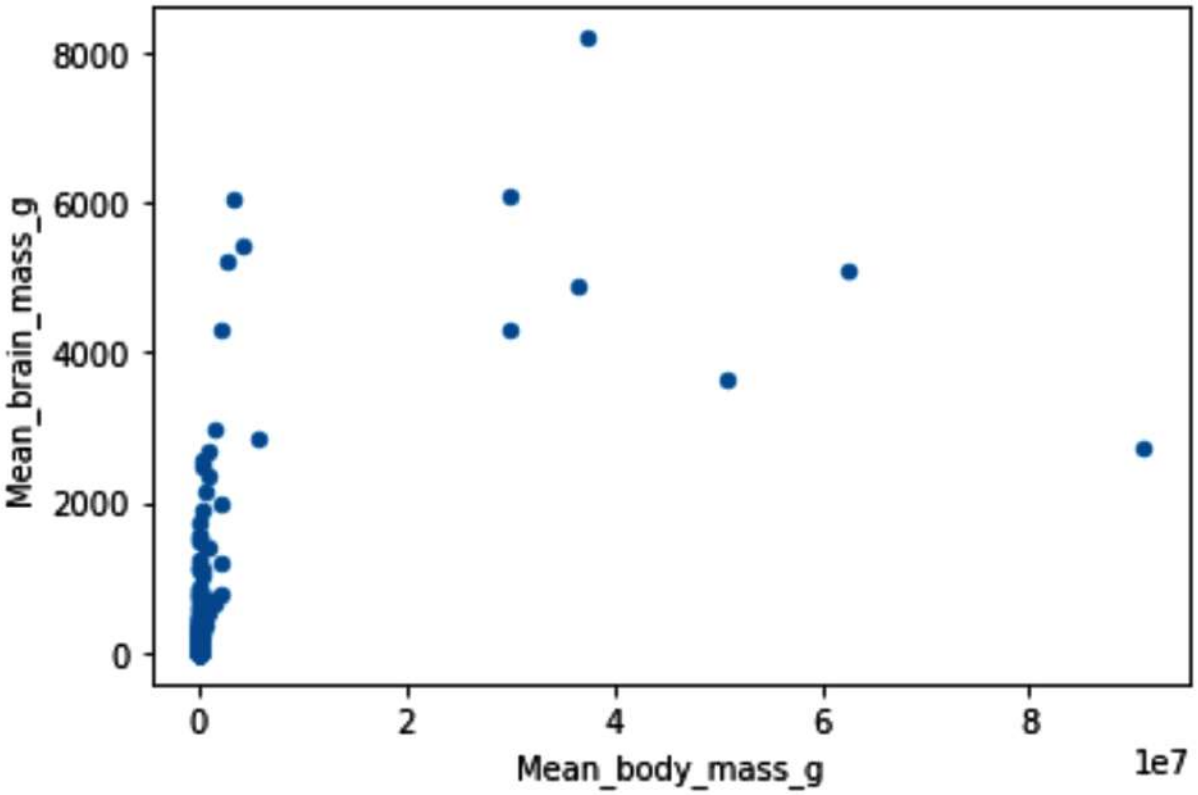
Missing Values	False	True
Top_song		
False	3280	338
True	392	35

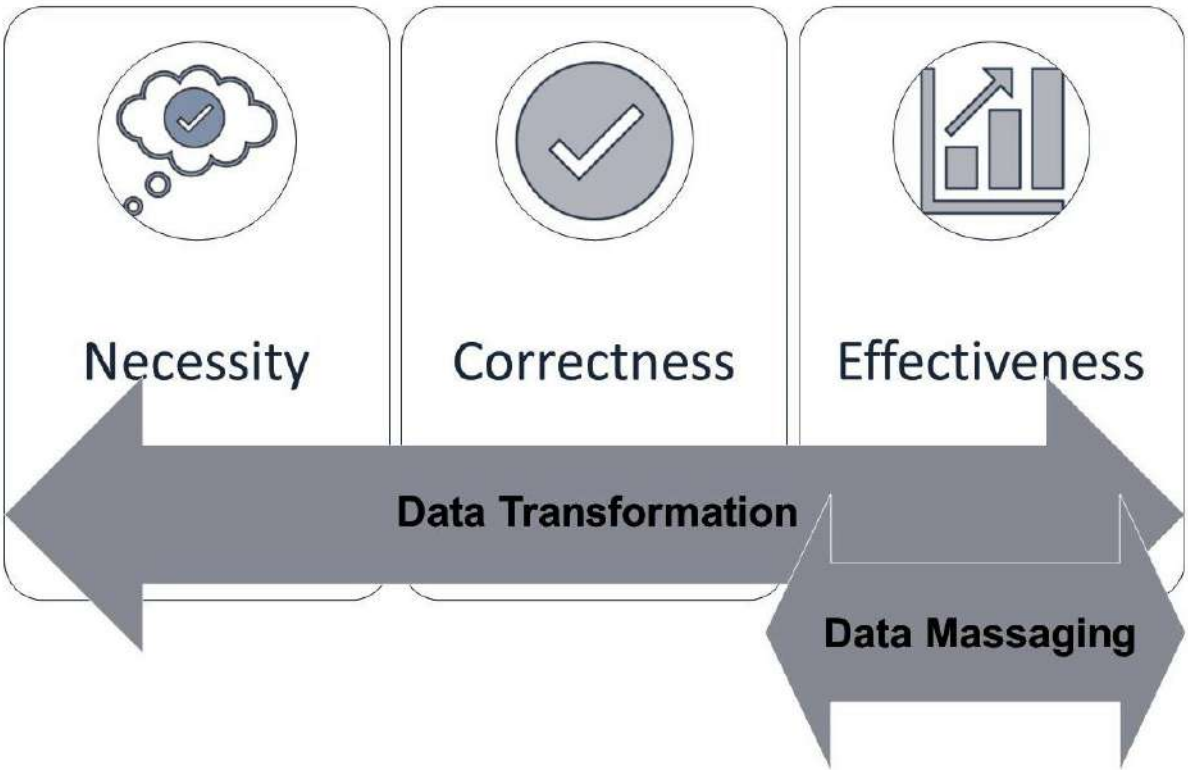
In [55]:

```
from scipy.stats import chi2_contingency
p_value = chi2_contingency(contigency_table)[1]
p_value
```

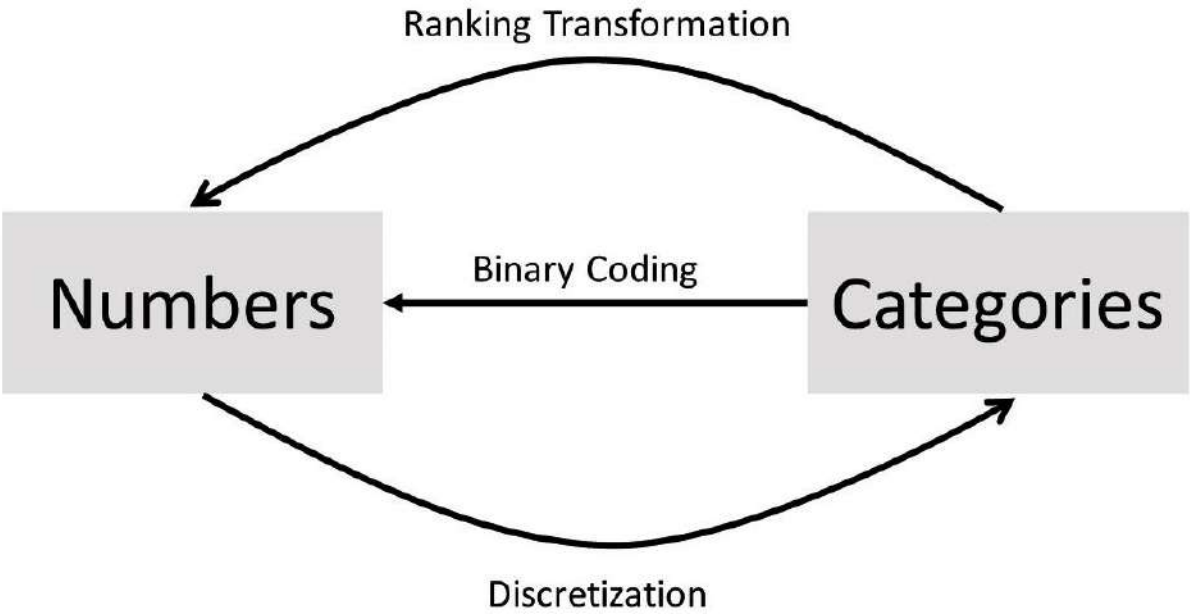
Out[55]: 0.4931640410927335

Chapter 13: Data Reduction





	Salary	GPA	N_Salary	N_GPA	S_Salary	S_GPA
	__A__	__B__	__NA__	__NB__	__SA__	__SB__
1	92000	3.25	0.75	0.339806	0.817616	-0.57477
2	83000	3.36	0.5	0.446602	-0.00919	-0.15882
3	83000	3.16	0.5	0.252427	-0.00919	-0.91509
4	72000	3.45	0.194444	0.533981	-1.01972	0.181506
5	101000	3.32	1	0.407767	1.644418	-0.31007
6	85000	3.57	0.555556	0.650485	0.174547	0.635271
7	74000	3.93	0.25	1	-0.83599	1.996565
8	65000	3.61	0	0.68932	-1.66279	0.786526
9	98000	3.47	0.916667	0.553398	1.368817	0.257133
10	78000	2.9	0.361111	0	-0.46852	-1.89825
Max=	101000	3.93	1	1	1.644418	1.996565
Min=	65000	2.9	0	0	-1.66279	-1.89825
Mean=	83100	3.402	0.502778	0.487379	0	0
STD=	10885.31	0.264454	0.30237	0.256752	1	1

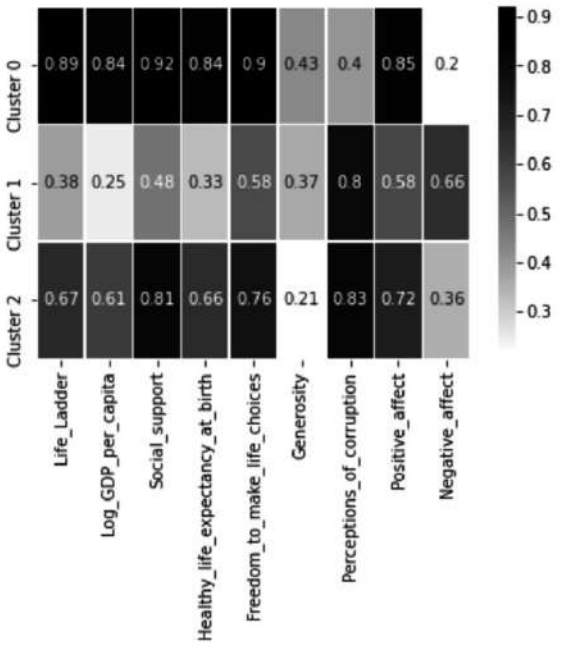


In [3]:

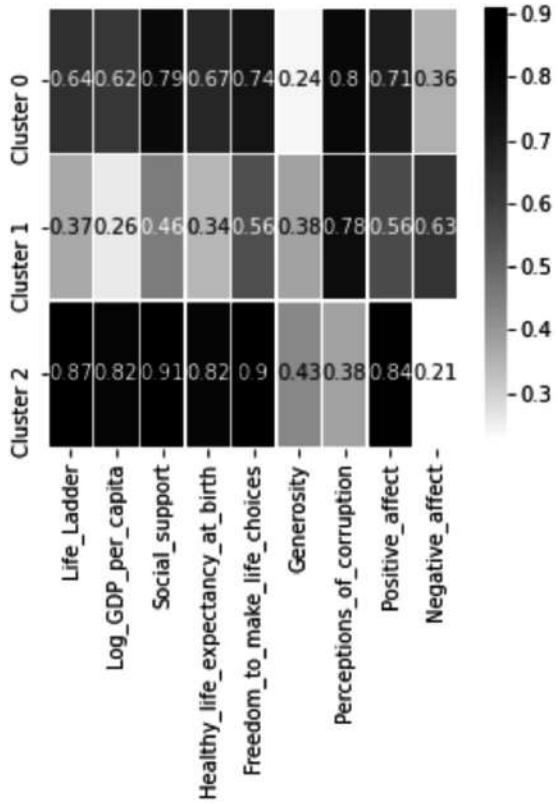
```
bc_Continent = pd.get_dummies(report2019_df.Continent)
bc_Continent.head(5)
```

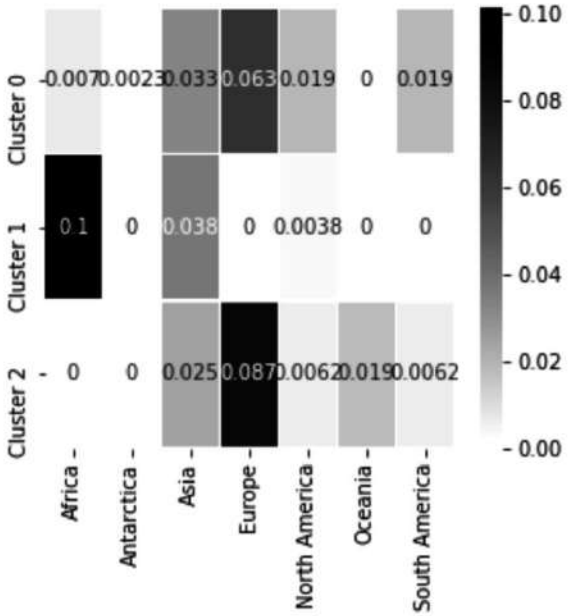
Out[3]:

Name	Africa	Antarctica	Asia	Europe	North America	Oceania	South America
Afghanistan	0	0	1	0	0	0	0
Albania	0	0	0	1	0	0	0
Algeria	1	0	0	0	0	0	0
Argentina	0	0	0	0	0	0	1
Armenia	0	0	0	1	0	0	0



a) Clustering Analysis without `Continent`





b) Clustering Analysis with `bc_Continent`

Education level
High School
Bachelor
High School
Masters
Doctorate
Bachelor
Masters
High School
High School
Bachelor

Binary Coding

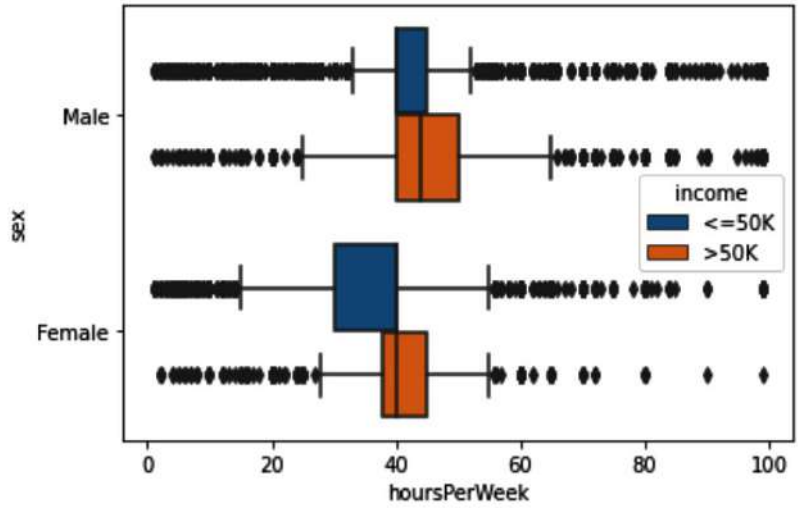
High School	Bachelor	Masters	Doctorate
1	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
0	0	1	0
1	0	0	0
1	0	0	0
0	1	0	0

Ranking Transformation

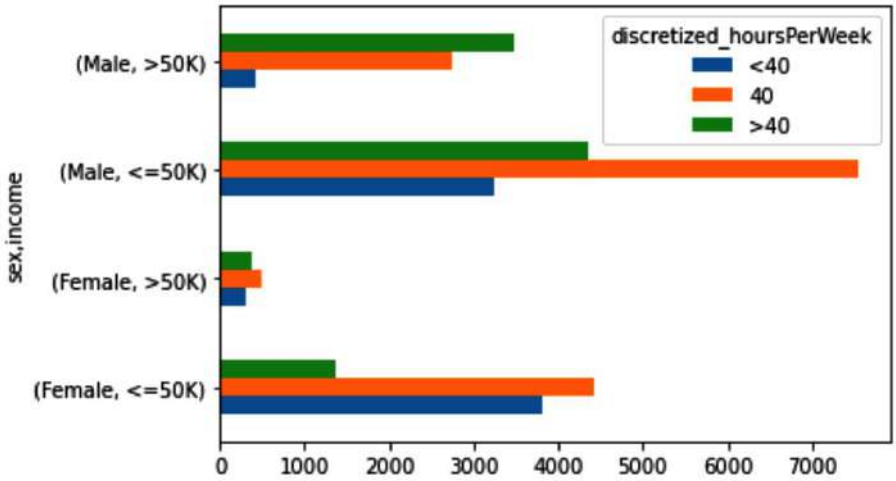
Education Rank
1
2
1
3
4
2
3
1
1
2

Attribute Construction

Education Years
12
16
12
18
21
16
18
12
12
16

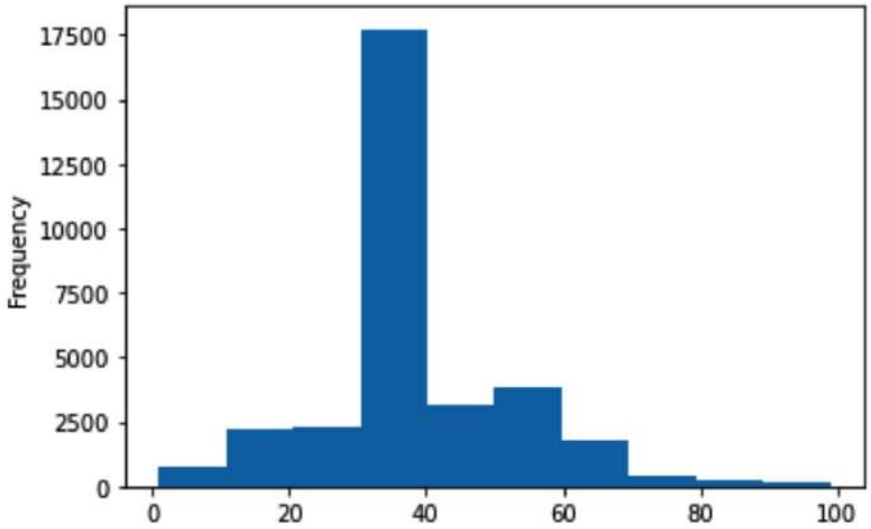


Discretize hoursPerWeek

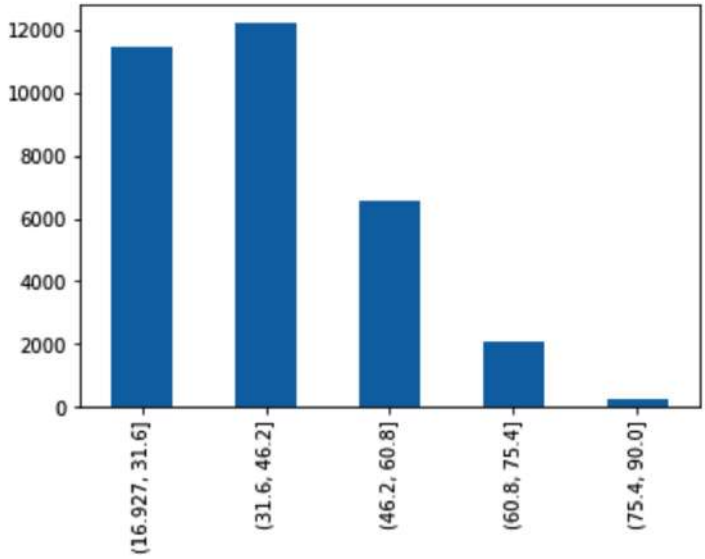


In [10]:

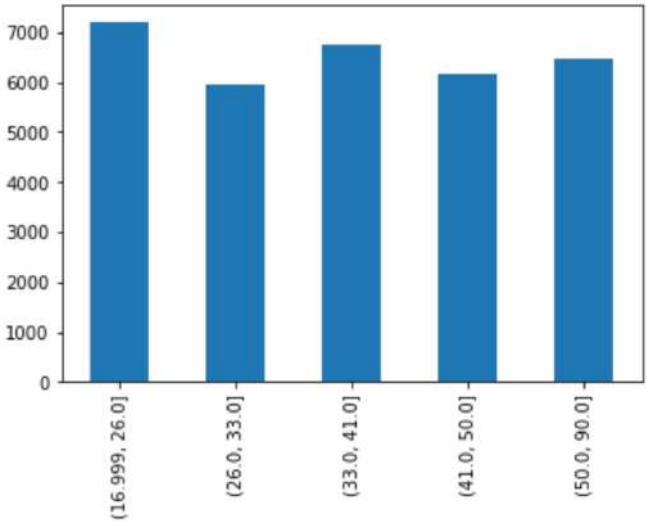
```
adult_df.hoursPerWeek.plot.hist()
plt.show()
```



In [13]:
```
pd.cut(adult_df.age, bins = 5).value_counts().sort_index().plot.bar()
plt.show()
```

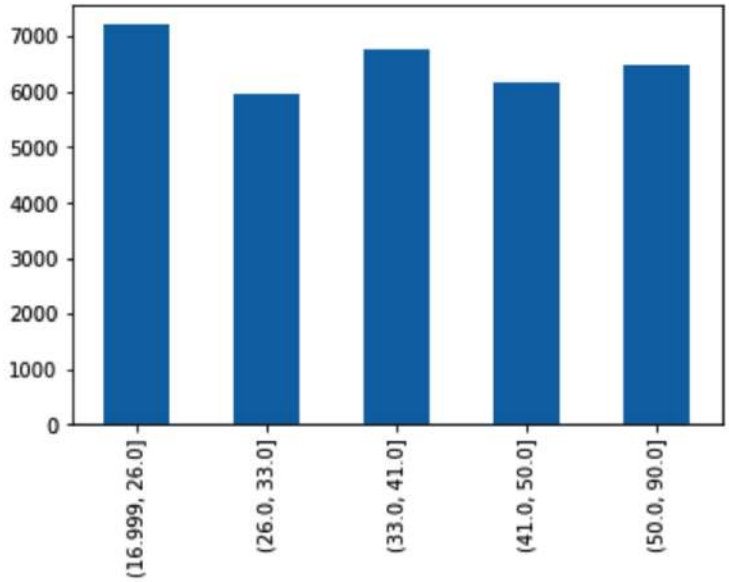


In [14]:
```
pd.qcut(adult_df.age,q=5).value_counts().sort_index().plot.bar()
plt.show()
```



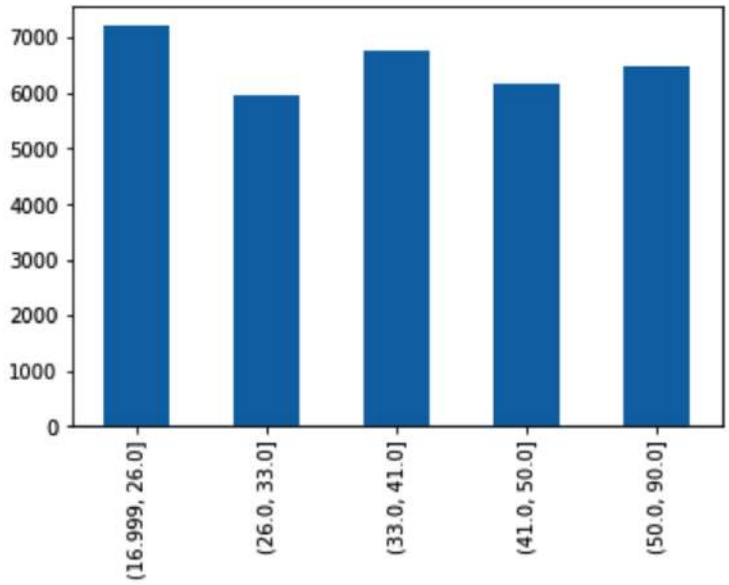
In [14]:

```
pd.qcut(adult_df.age,q=5,
        duplicates='drop').value_counts().sort_index().plot.bar()
plt.show()
```



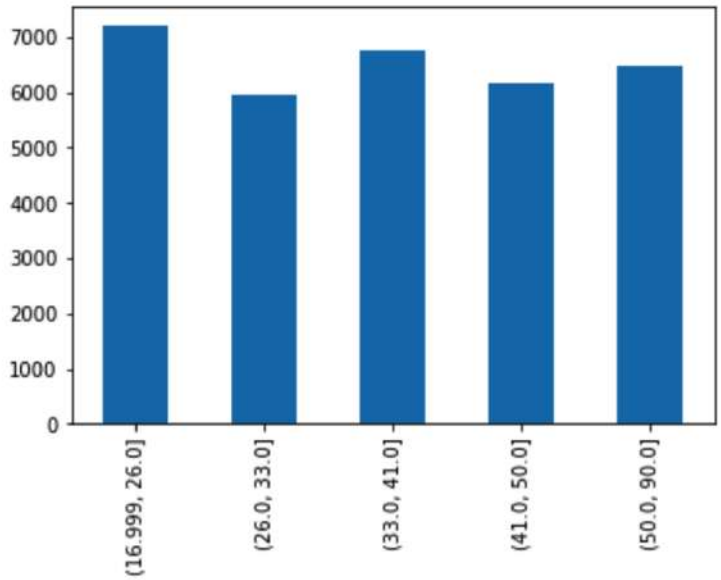
In [14]:

```
pd.qcut(adult_df.age,q=5,
        duplicates='drop').value_counts().sort_index().plot.bar()
plt.show()
```



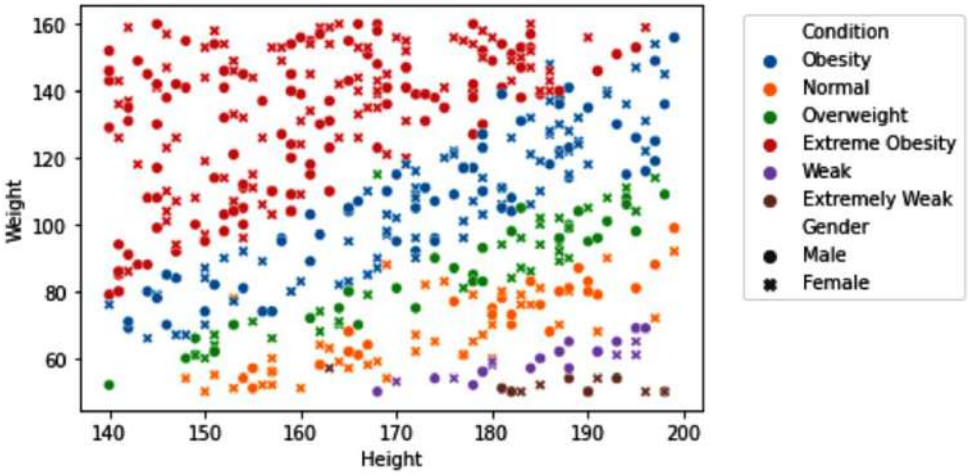
In [14]:

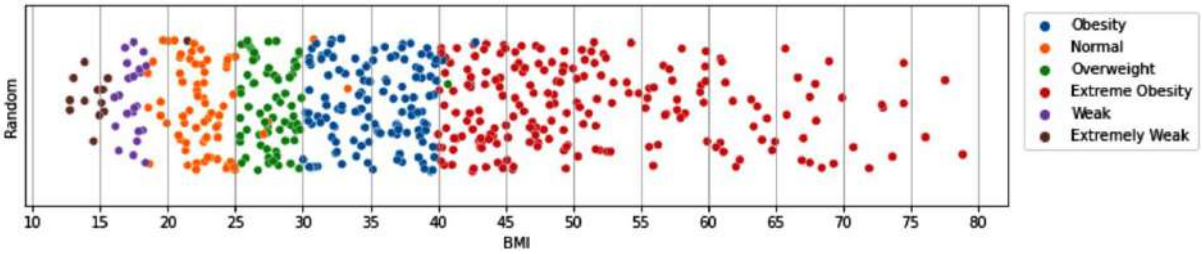
```
pd.qcut(adult_df.age,q=5).value_counts().sort_index().plot.bar()
plt.show()
```



In [17]:

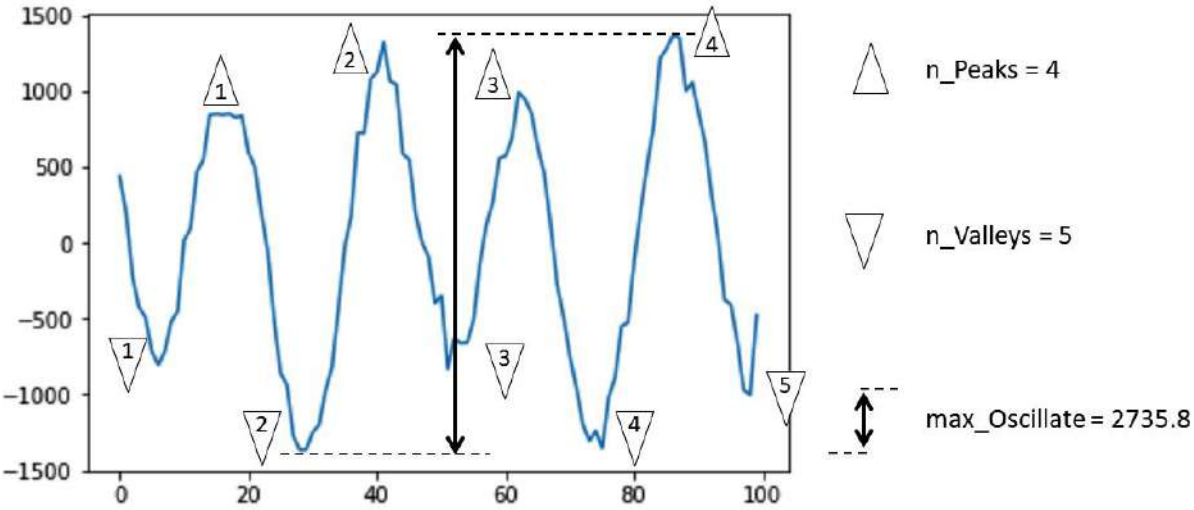
```
sns.scatterplot(data = person_df, x='Height',y='Weight',
                hue='Condition',style='Gender')
plt.legend(bbox_to_anchor=(1.05, 1))
plt.show()
```

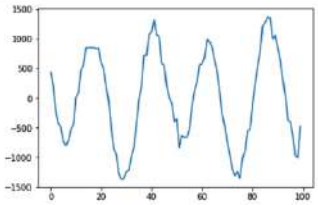
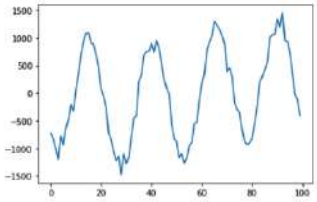
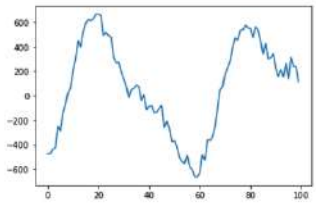
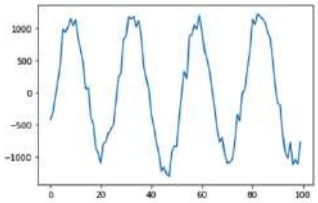
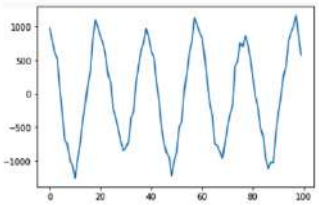




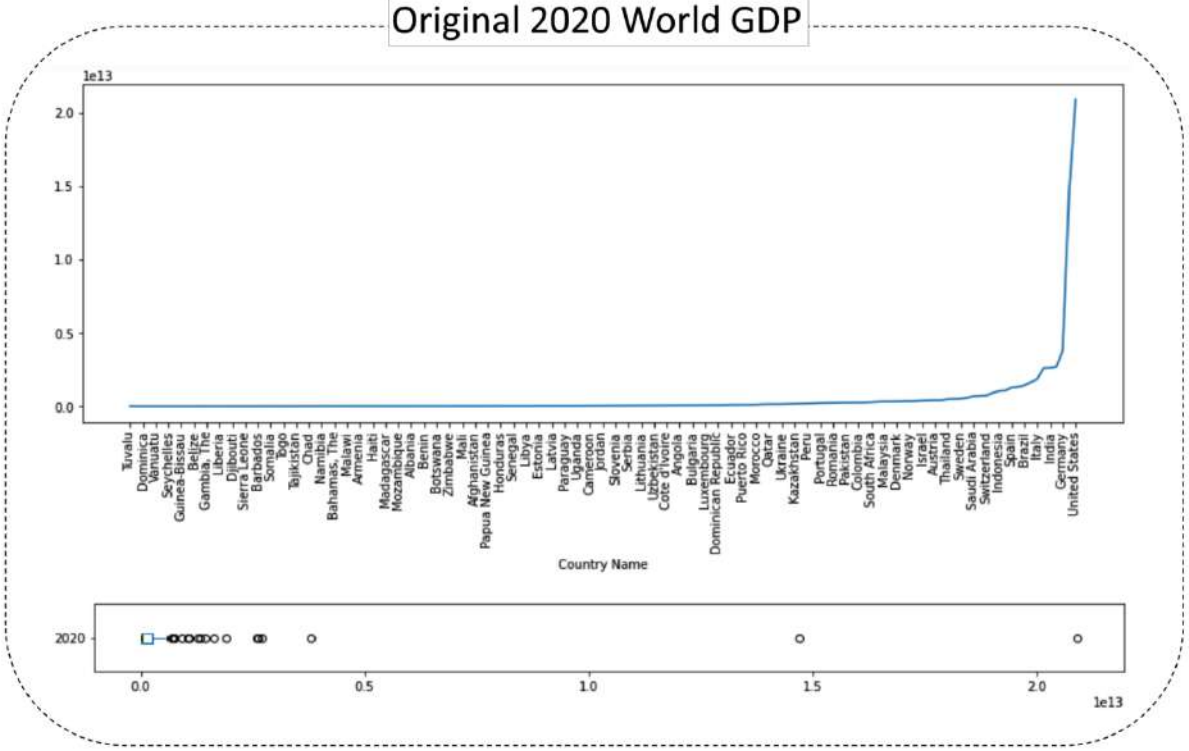
Email
Lkjds.fds@gmail.com
om21sdfds@gmail.com
89u43q@yahoo.com
lkdsjfa@redlands.edu
84utfd@gmail.com
iowjlk@msstate.edu
5431sldojk@yahoo.com
39dfoiuy@outlook.com
kljed@att.org
Lks321ld@calpoly.edu
jdsfl@gmail.com

Popular Free Platform	.edu	Others
1	0	0
1	0	0
1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
0	0	1
0	0	1
0	1	0
1	0	0

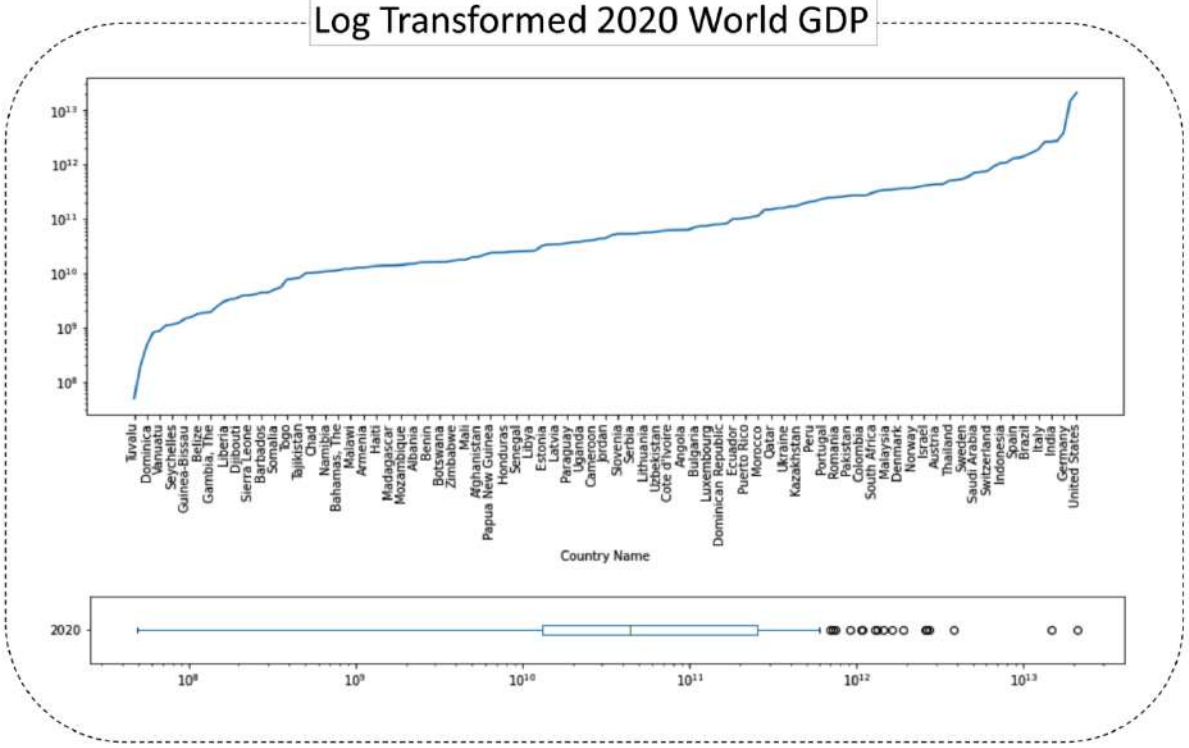


		n_Peaks	n_Valleys	max_Oscillate
Healthy		4	5	2735.8
Fault 1		4	4	2931.0
Fault 2		2	1	1331.5
Fault 3		4	4	2530.8
Fault 4		5	5	2422.0

Original 2020 World GDP

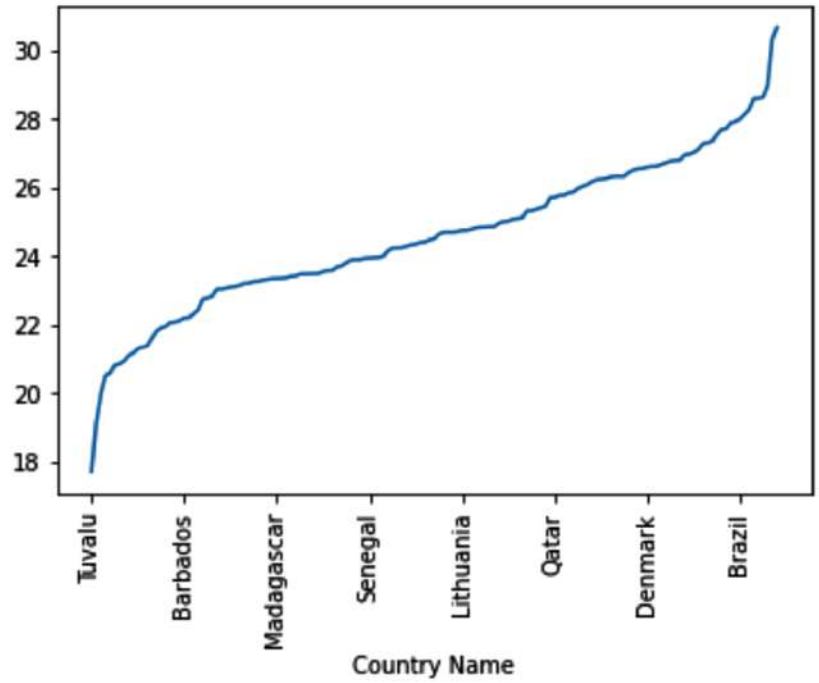


Log Transformed 2020 World GDP



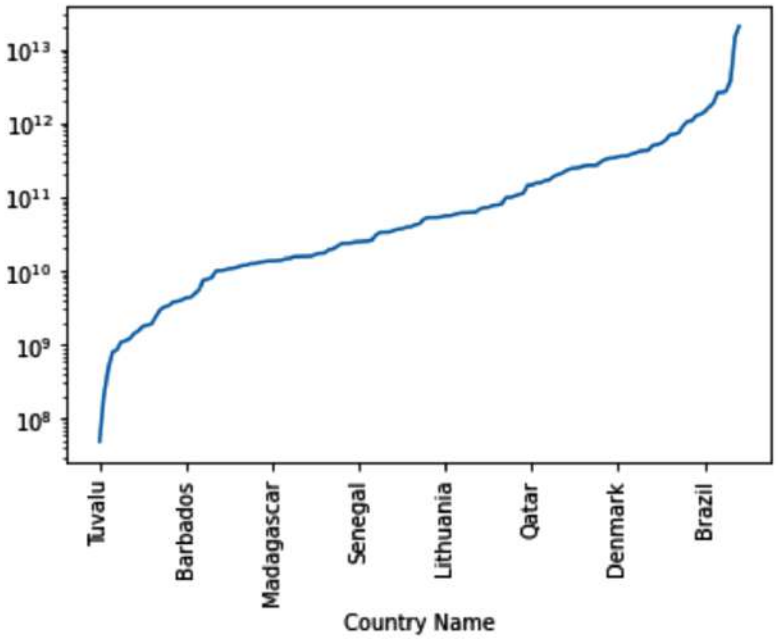
In [26]:

```
country_df['log_2020'] = np.log(country_df['2020'])
country_df.log_2020.sort_values().plot()
plt.xticks(rotation=90)
plt.show()
```



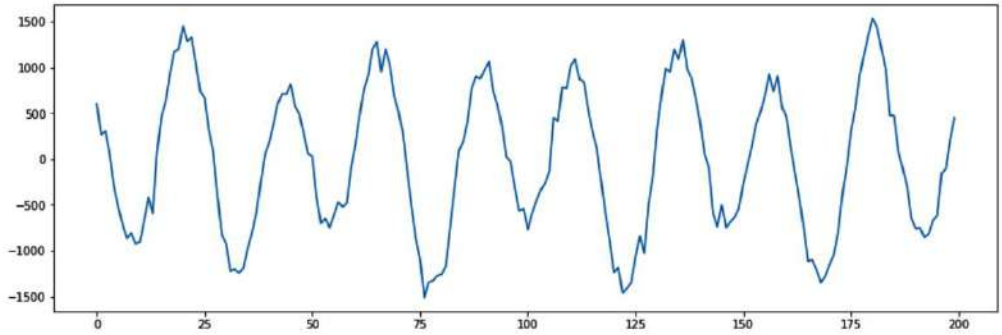
In [27]:

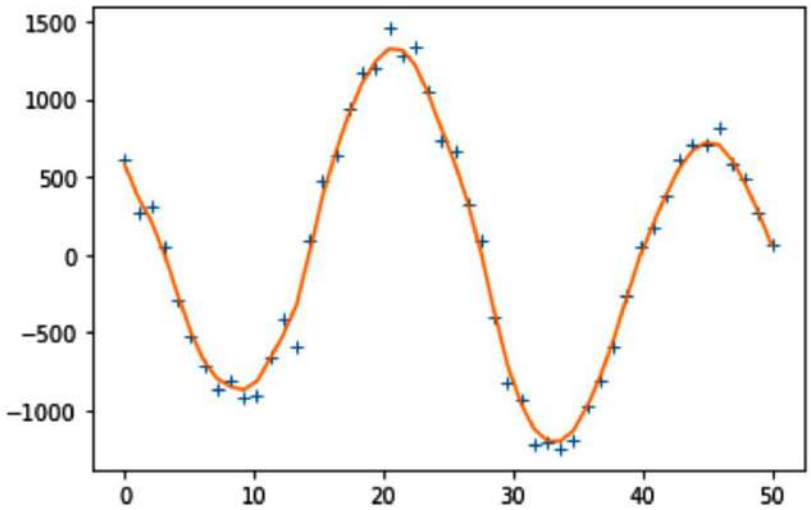
```
country_df['2020'].sort_values().plot(logy=True)
plt.xticks(rotation=90)
plt.show()
```

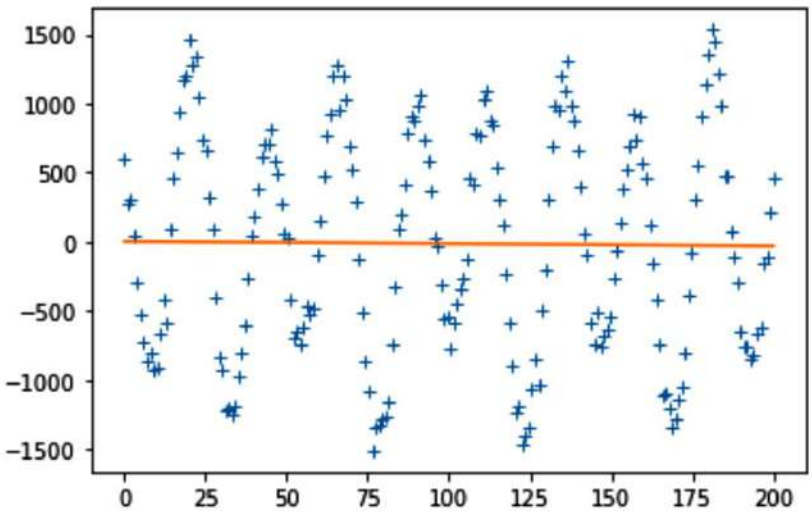


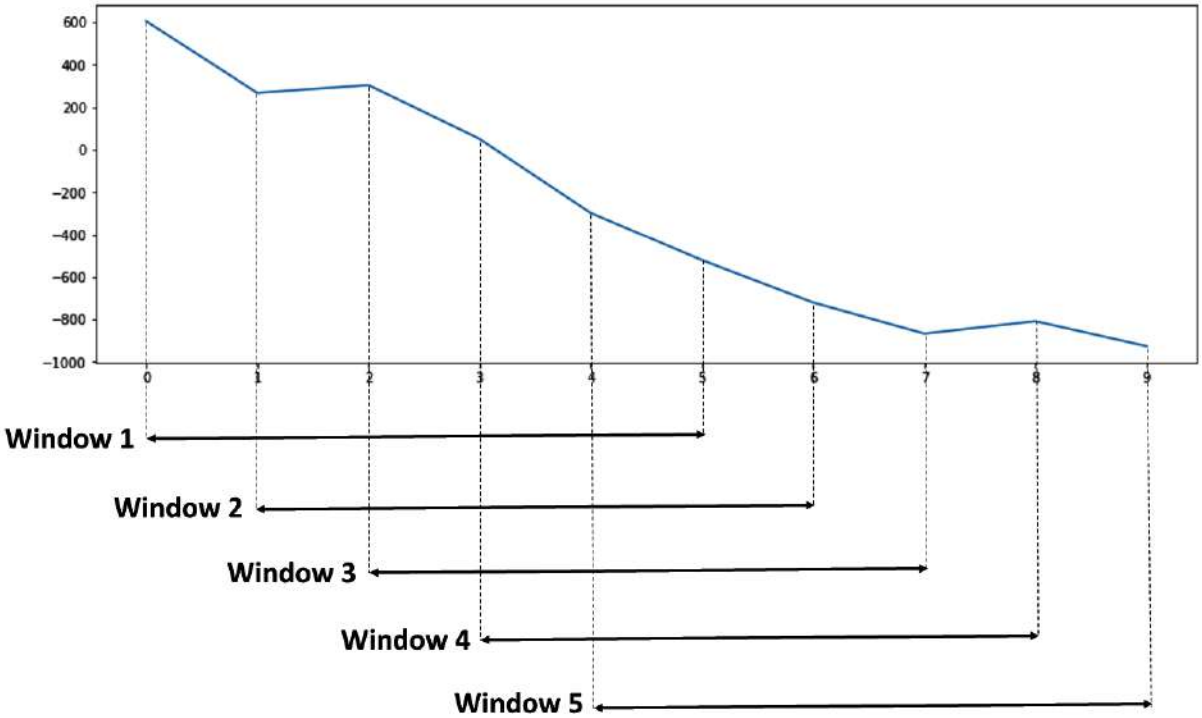
In [28]:

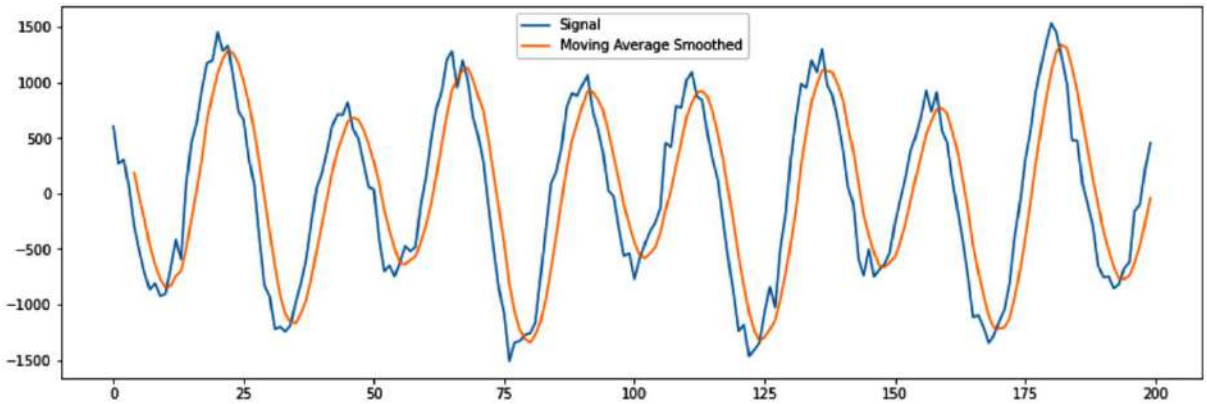
```
signal_df = pd.read_csv('Noise_data.csv')
signal_df.drop(columns='t',inplace=True)
signal_df.Signal.plot(figsize=(15,5))
plt.show()
```



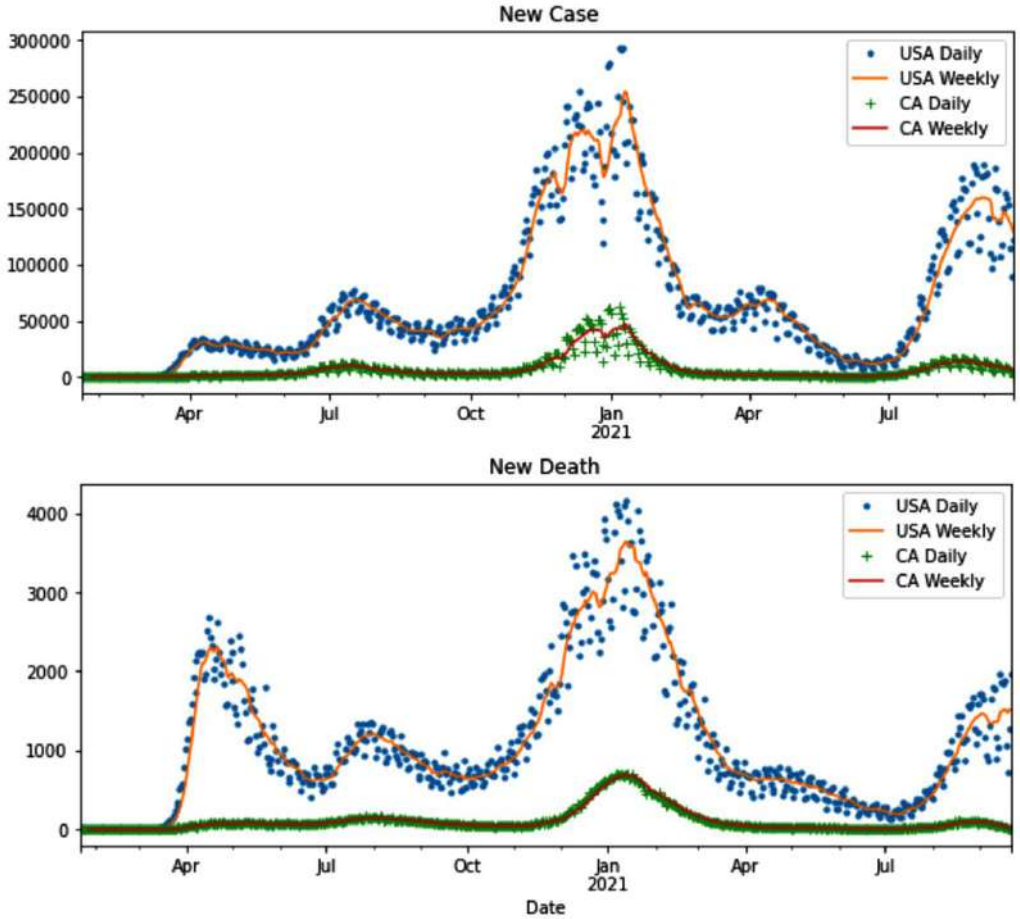





Window 1
Window 2
Window 3
Window 4
Window 5

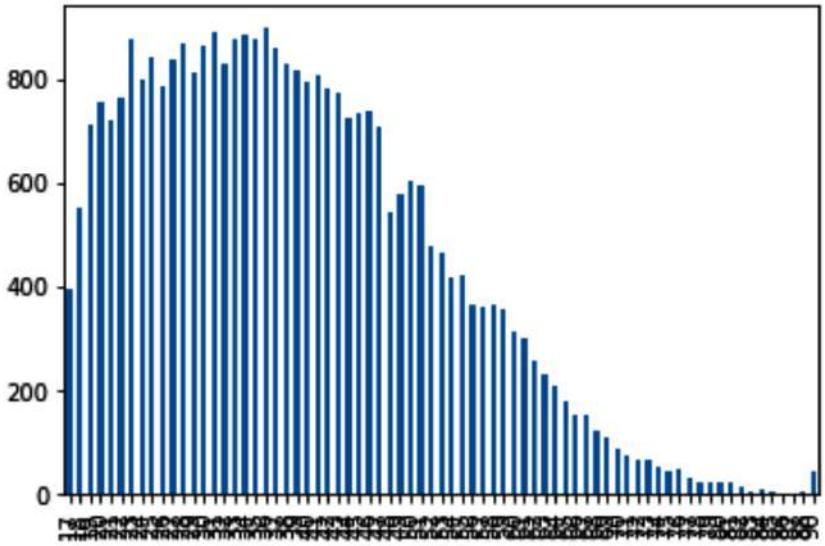

Signal
Moving Average Smoothed

	Signal	Moving Average Smoothed
0	605.340308	NaN
1	267.958658	NaN
2	304.652019	NaN
3	51.297364	NaN
4	-297.546288	186.340412
5	-520.492600	-38.826169
6	-719.919832	-236.401867
7	-866.546219	-470.641515
8	-807.907263	-642.482441
9	-925.817440	-768.136671



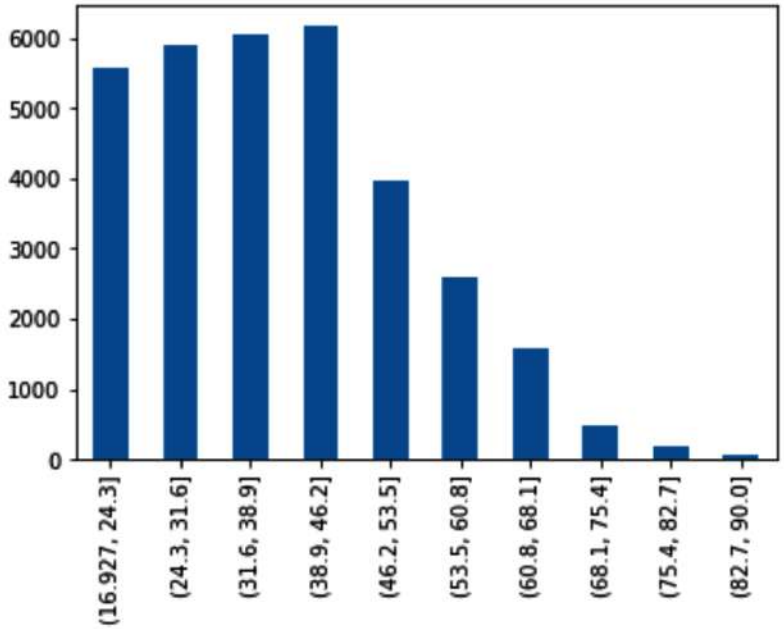
In [35]:

```
adult_df.age.value_counts().sort_index().plot.bar()
plt.show()
```



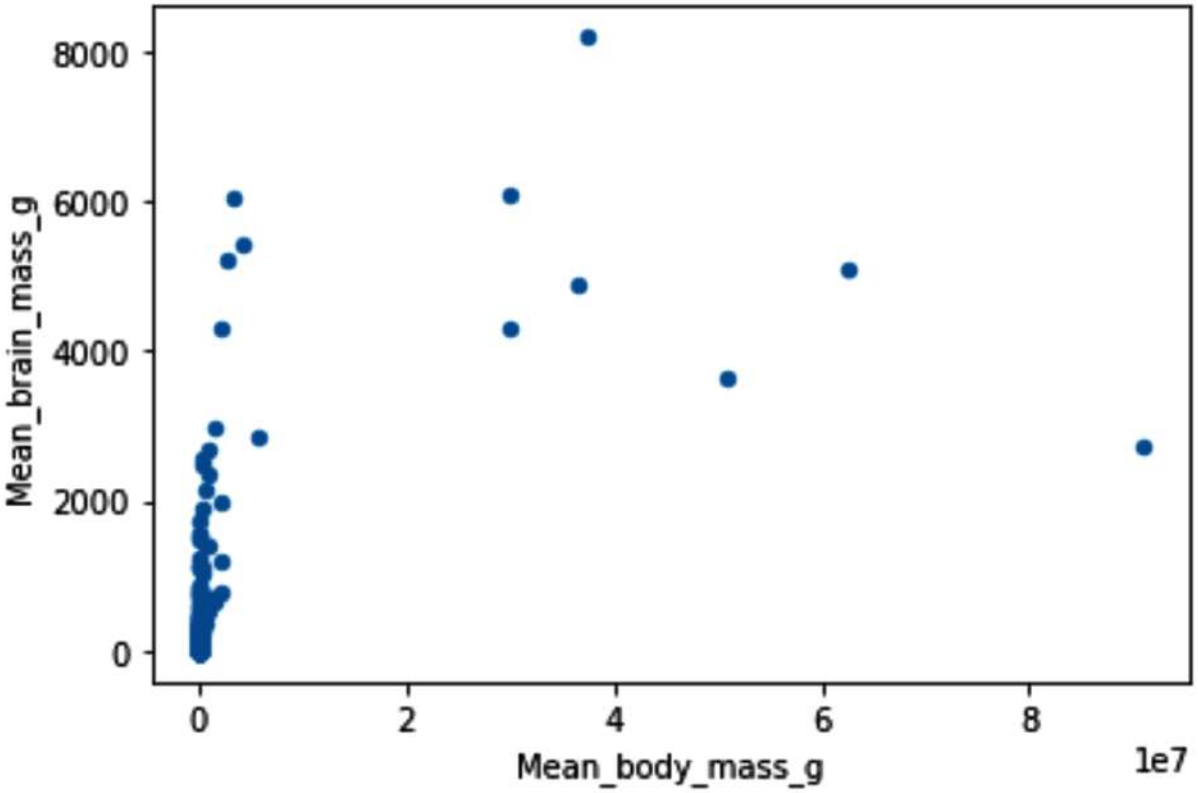
In [36]:
```
adult_df['age_binned']=pd.cut(adult_df.age,10)
adult_df.age_binned.value_counts().sort_index().plot.bar()
```

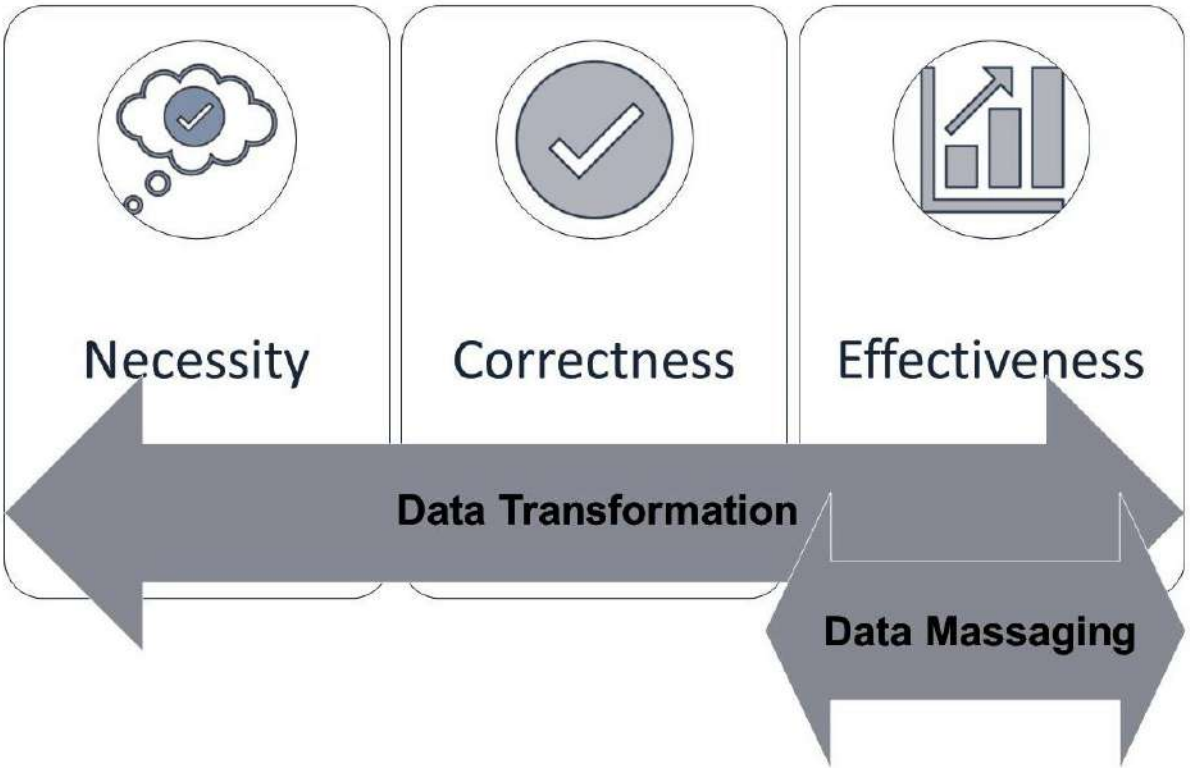
Out[36]: <AxesSubplot:>



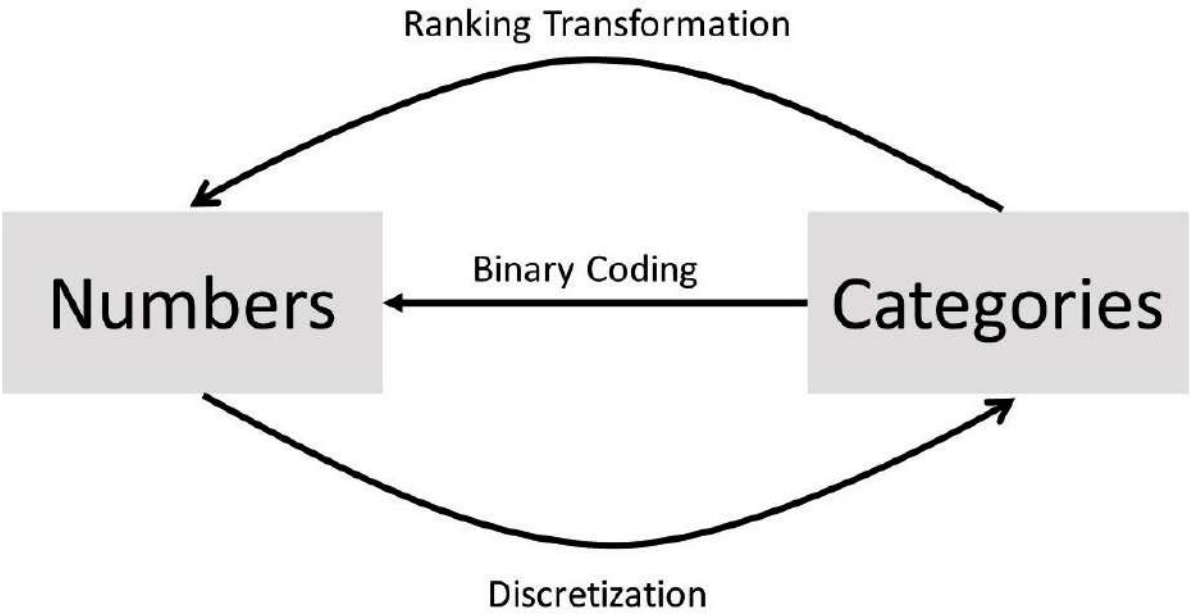
Index	Color	index	Color	Index	Color
1	Blue	11	White	21	Orange
2	Blue	12	Orange	22	Black
3	Black	13	White	23	Yellow
4	White	14	Black	24	Black
5	Green	15	Yellow	25	Orange
6	Orange	16	Yellow	26	White
7	White	17	Blue	27	Blue
8	Blue	18	Green	28	Orange
9	Brown	19	Orange	29	Orange
10	Yellow	20	Green	30	Yellow

Chapter 14: Data Transformation and Massaging





	Salary	GPA	N_Salary	N_GPA	S_Salary	S_GPA
	__A__	__B__	__NA__	__NB__	__SA__	__SB__
1	92000	3.25	0.75	0.339806	0.817616	-0.57477
2	83000	3.36	0.5	0.446602	-0.00919	-0.15882
3	83000	3.16	0.5	0.252427	-0.00919	-0.91509
4	72000	3.45	0.194444	0.533981	-1.01972	0.181506
5	101000	3.32	1	0.407767	1.644418	-0.31007
6	85000	3.57	0.555556	0.650485	0.174547	0.635271
7	74000	3.93	0.25	1	-0.83599	1.996565
8	65000	3.61	0	0.68932	-1.66279	0.786526
9	98000	3.47	0.916667	0.553398	1.368817	0.257133
10	78000	2.9	0.361111	0	-0.46852	-1.89825
Max=	101000	3.93	1	1	1.644418	1.996565
Min=	65000	2.9	0	0	-1.66279	-1.89825
Mean=	83100	3.402	0.502778	0.487379	0	0
STD=	10885.31	0.264454	0.30237	0.256752	1	1

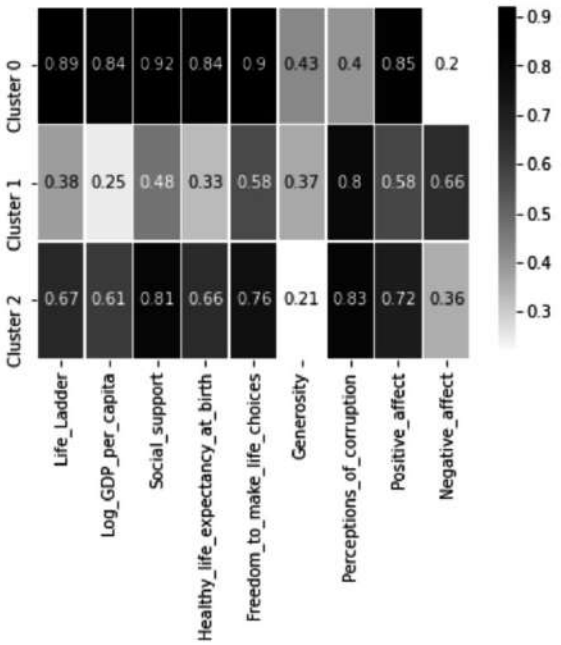


In [3]:

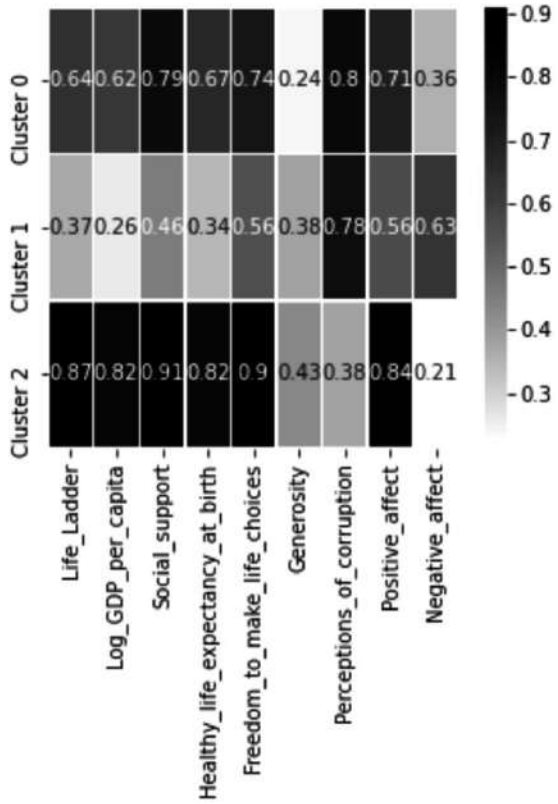
```
bc_Continent = pd.get_dummies(report2019_df.Continent)
bc_Continent.head(5)
```

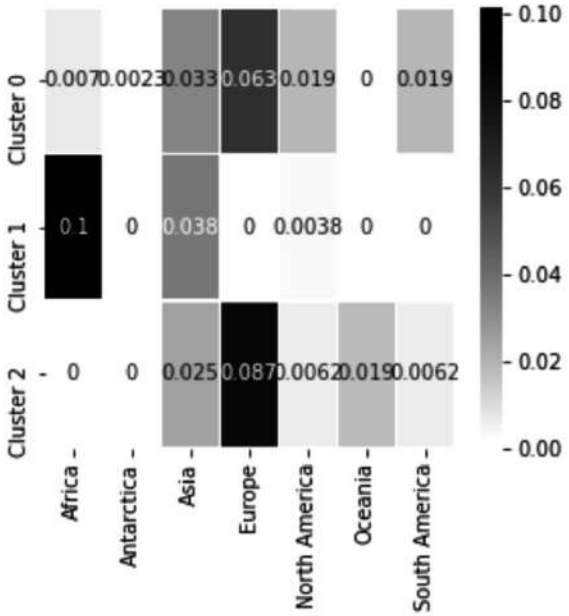
Out[3]:

Name	Africa	Antarctica	Asia	Europe	North America	Oceania	South America
Afghanistan	0	0	1	0	0	0	0
Albania	0	0	0	1	0	0	0
Algeria	1	0	0	0	0	0	0
Argentina	0	0	0	0	0	0	1
Armenia	0	0	0	1	0	0	0



a) Clustering Analysis without `Continent`





b) Clustering Analysis with `bc_Continent`

Education level
High School
Bachelor
High School
Masters
Doctorate
Bachelor
Masters
High School
High School
Bachelor

Binary Coding

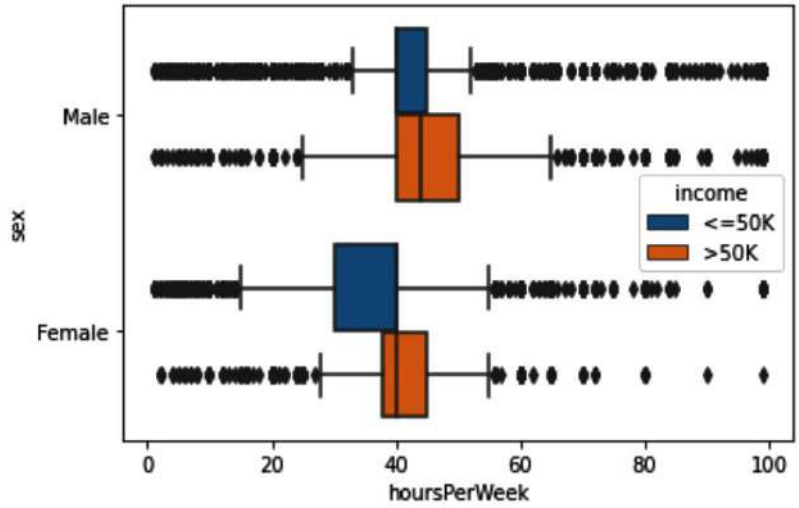
High School	Bachelor	Masters	Doctorate
1	0	0	0
0	1	0	0
1	0	0	0
0	0	1	0
0	0	0	1
0	1	0	0
0	0	1	0
1	0	0	0
1	0	0	0
0	1	0	0

Ranking Transformation

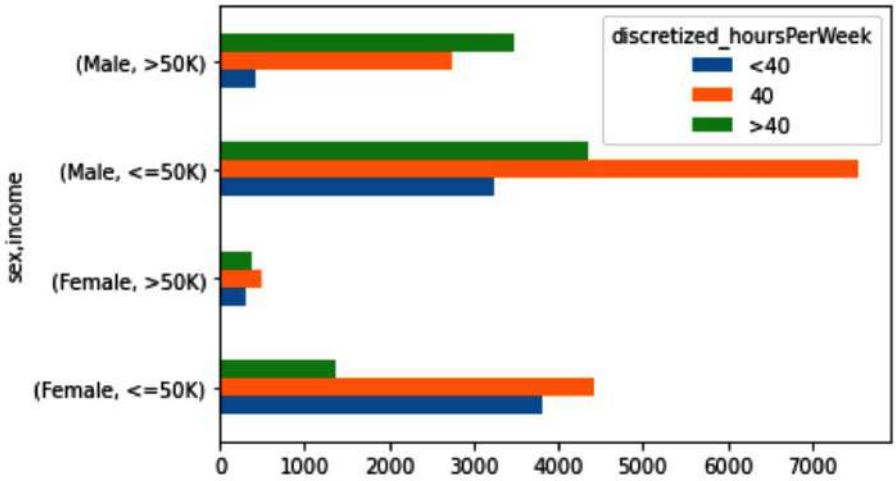
Education Rank
1
2
1
3
4
2
3
1
1
2

Attribute Construction

Education Years
12
16
12
18
21
16
18
12
12
16

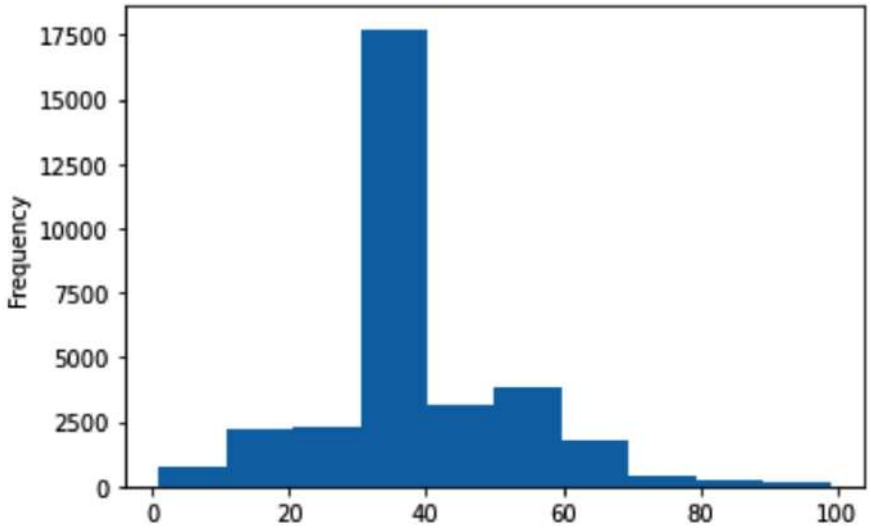


Discretize hoursPerWeek

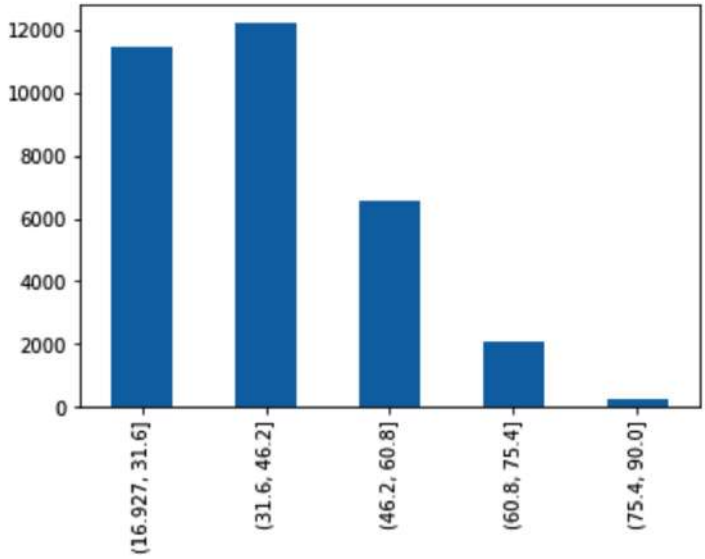


In [10]:

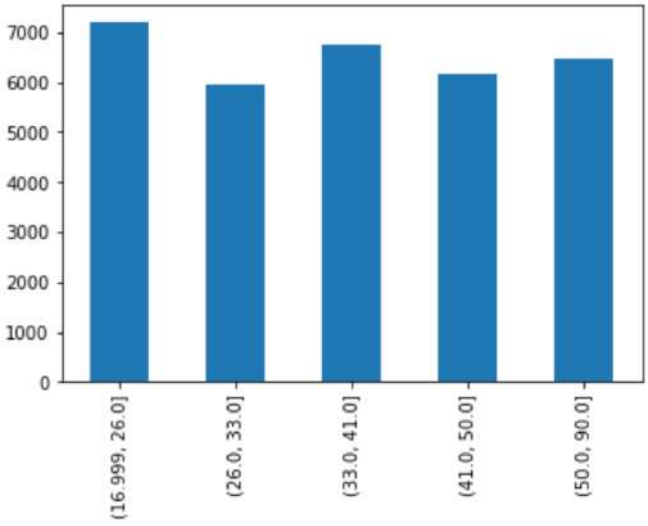
```
adult_df.hoursPerWeek.plot.hist()
plt.show()
```



In [13]:
```
pd.cut(adult_df.age, bins = 5).value_counts().sort_index().plot.bar()
plt.show()
```

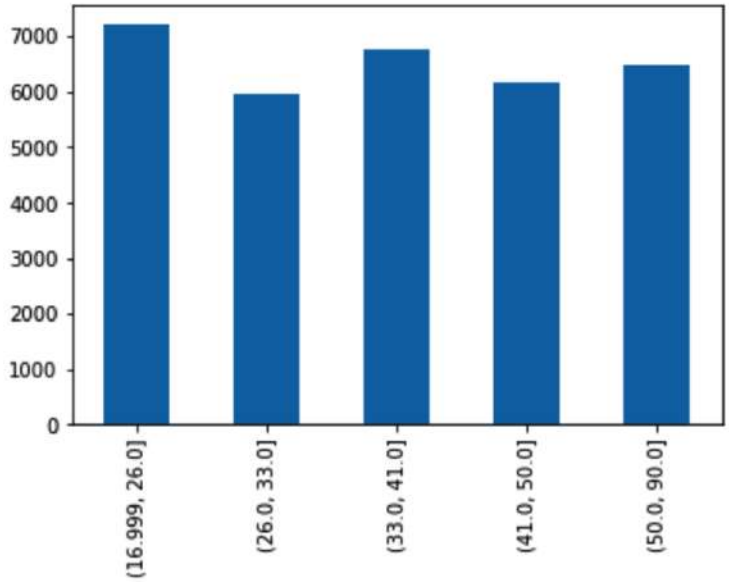


In [14]:
```
pd.qcut(adult_df.age,q=5).value_counts().sort_index().plot.bar()
plt.show()
```



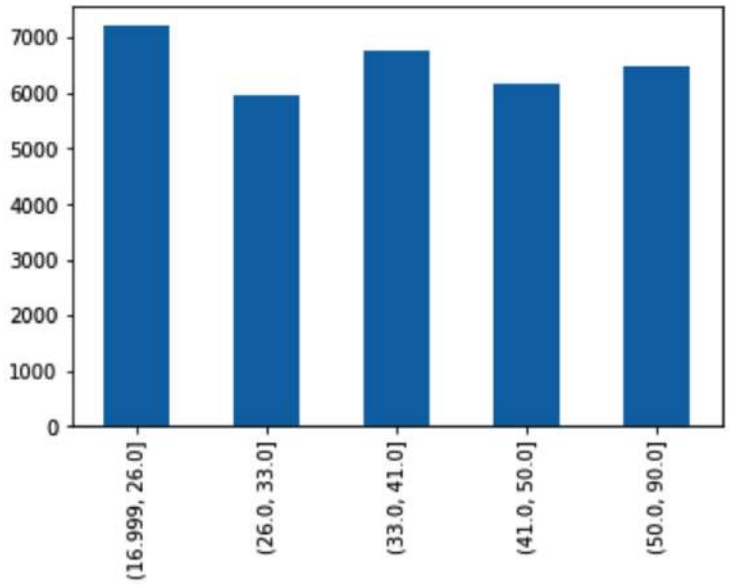
In [14]:

```
pd.qcut(adult_df.age,q=5,
        duplicates='drop').value_counts().sort_index().plot.bar()
plt.show()
```



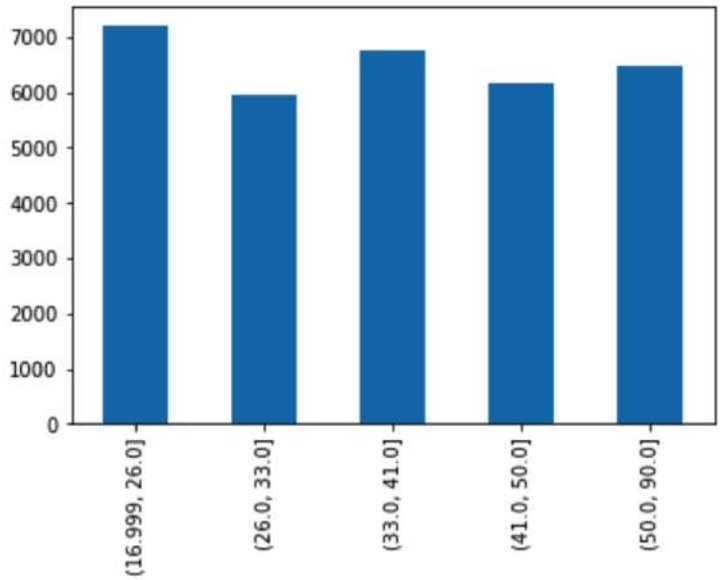
In [14]:

```
pd.qcut(adult_df.age,q=5,
        duplicates='drop').value_counts().sort_index().plot.bar()
plt.show()
```



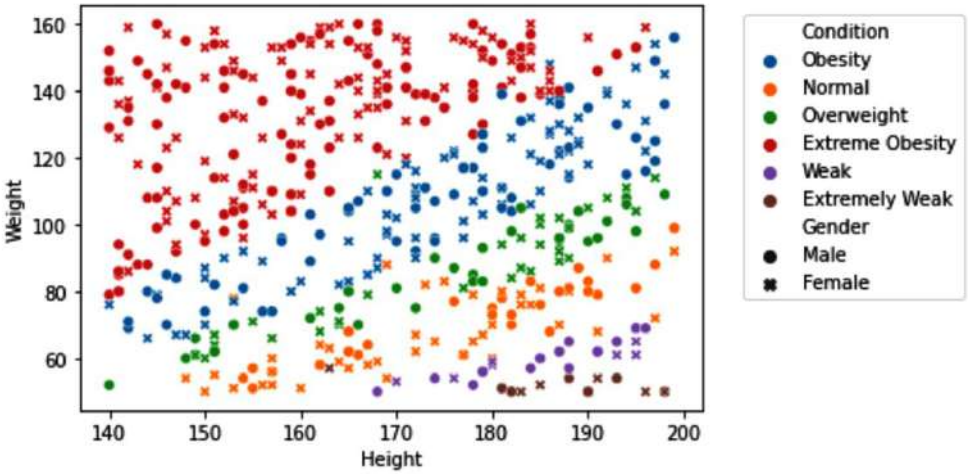
In [14]:

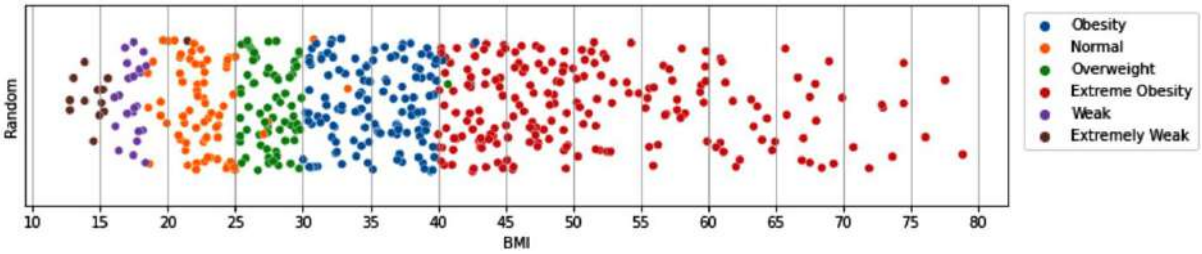
```
pd.qcut(adult_df.age,q=5).value_counts().sort_index().plot.bar()
plt.show()
```



In [17]:

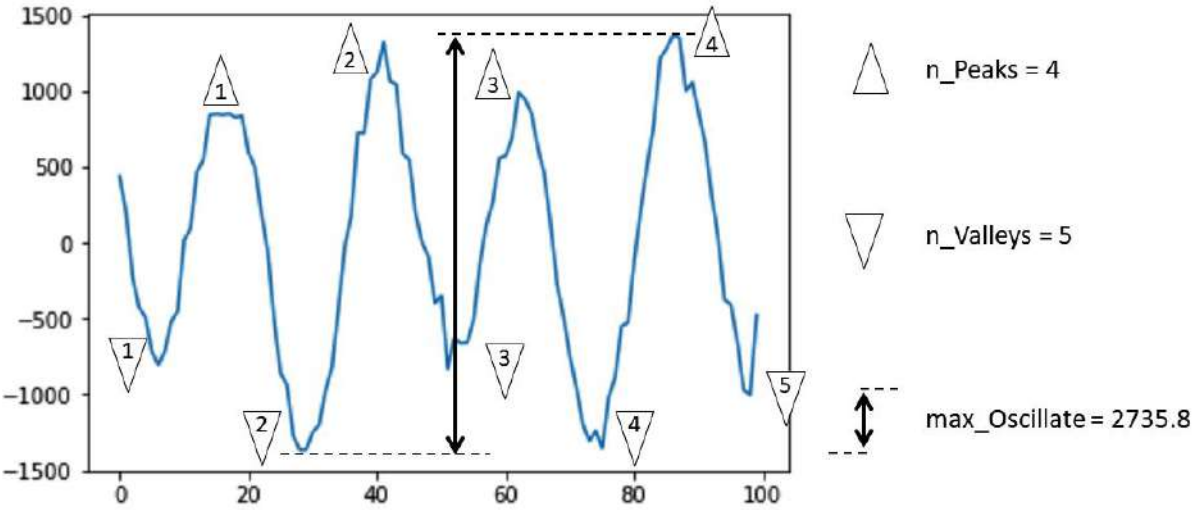
```
sns.scatterplot(data = person_df, x='Height',y='Weight',
                hue='Condition',style='Gender')
plt.legend(bbox_to_anchor=(1.05, 1))
plt.show()
```

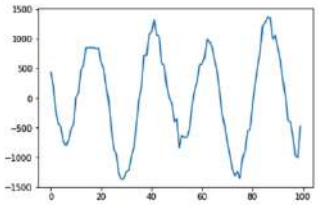
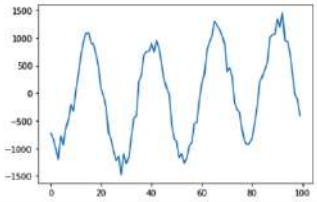
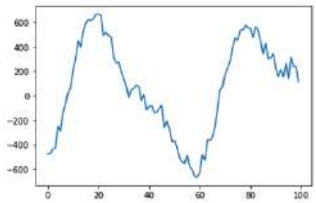
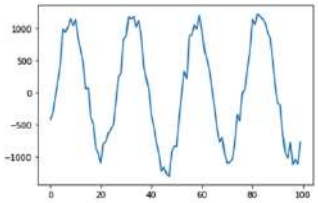
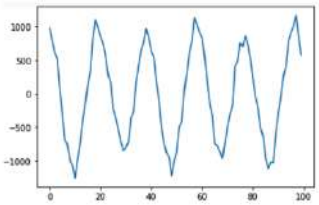




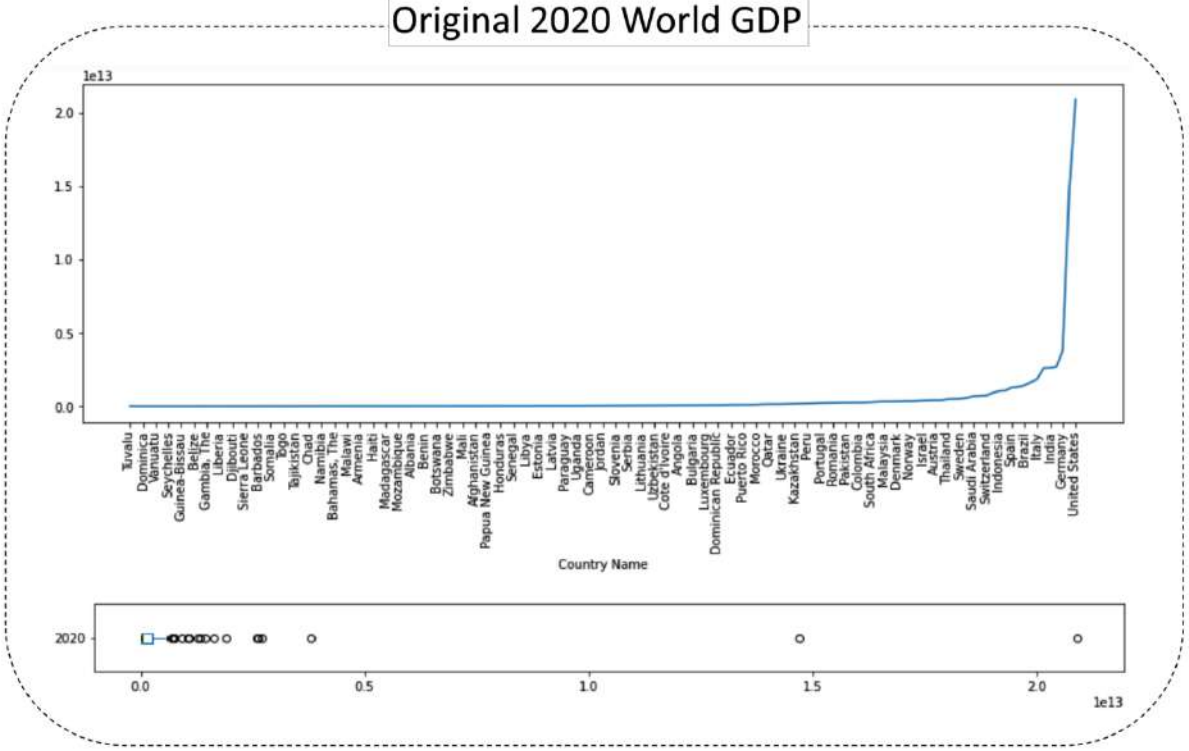
Email
Lkjds.fds@gmail.com
om21sdfds@gmail.com
89u43q@yahoo.com
lkdsjfa@redlands.edu
84utfd@gmail.com
iowjlk@msstate.edu
5431sldojk@yahoo.com
39dfoiuy@outlook.com
kljed@att.org
Lks321ld@calpoly.edu
jdsfl@gmail.com

Popular Free Platform	.edu	Others
1	0	0
1	0	0
1	0	0
0	1	0
1	0	0
0	1	0
1	0	0
0	0	1
0	0	1
0	1	0
1	0	0

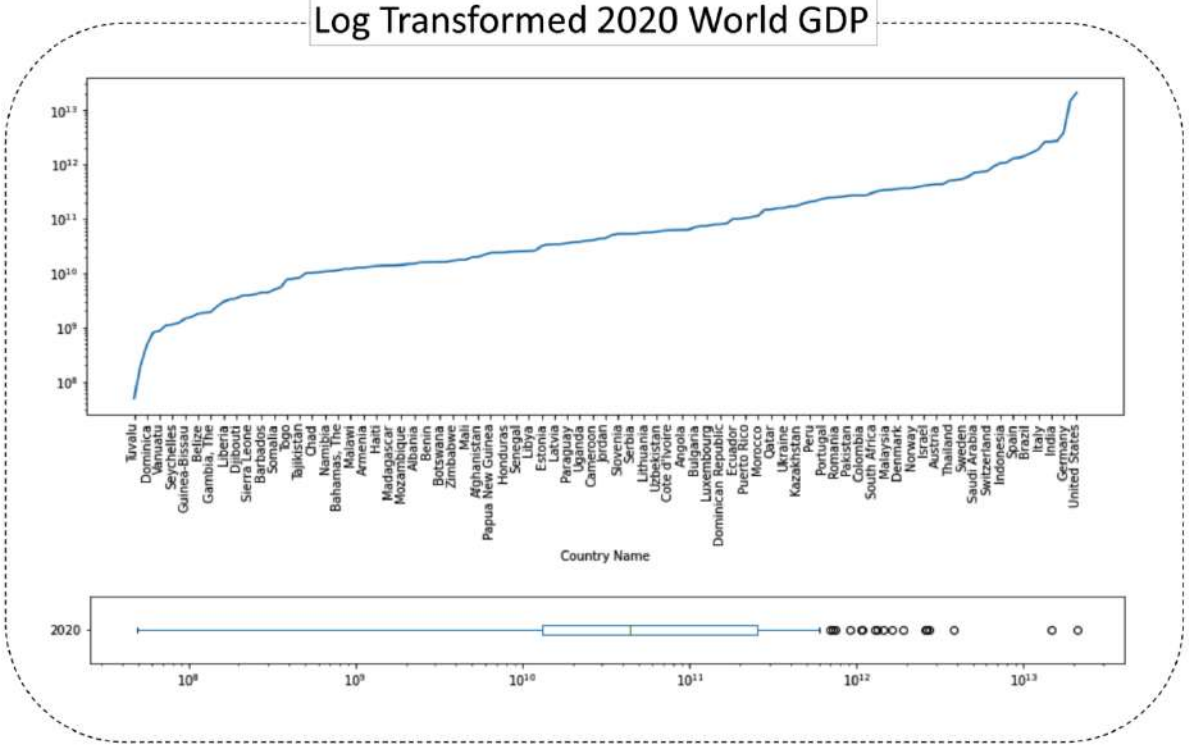


		n_Peaks	n_Valleys	max_Oscillate
Healthy		4	5	2735.8
Fault 1		4	4	2931.0
Fault 2		2	1	1331.5
Fault 3		4	4	2530.8
Fault 4		5	5	2422.0

Original 2020 World GDP

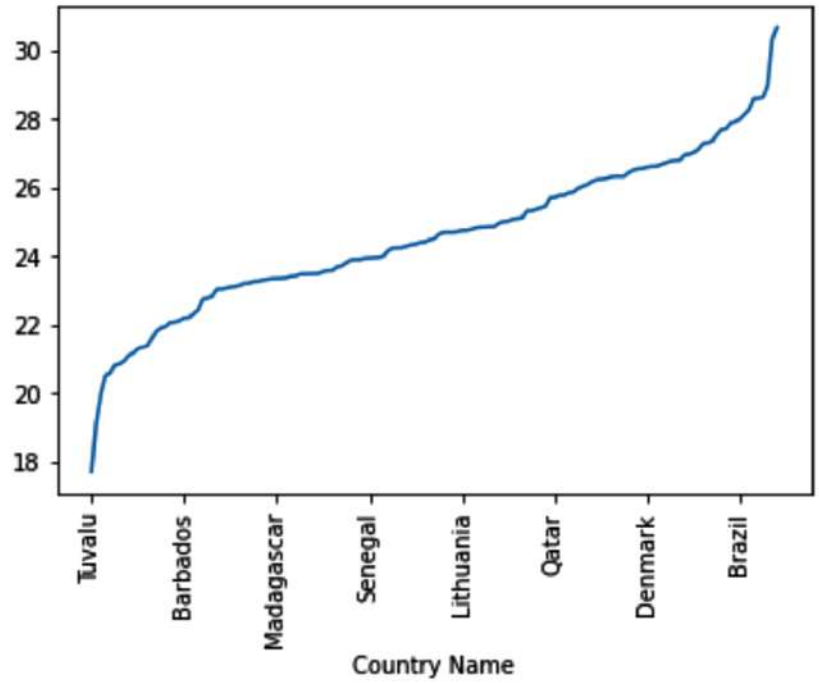


Log Transformed 2020 World GDP



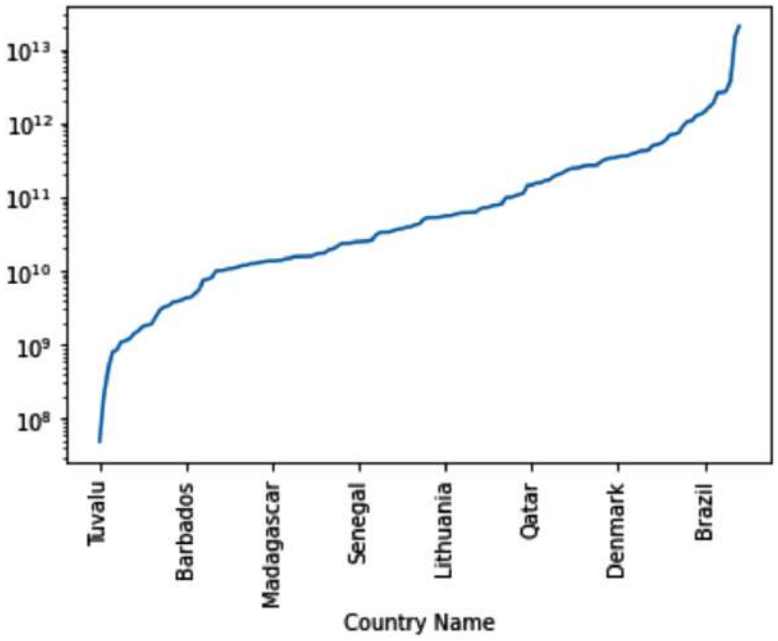
In [26]:

```
country_df['log_2020'] = np.log(country_df['2020'])
country_df.log_2020.sort_values().plot()
plt.xticks(rotation=90)
plt.show()
```



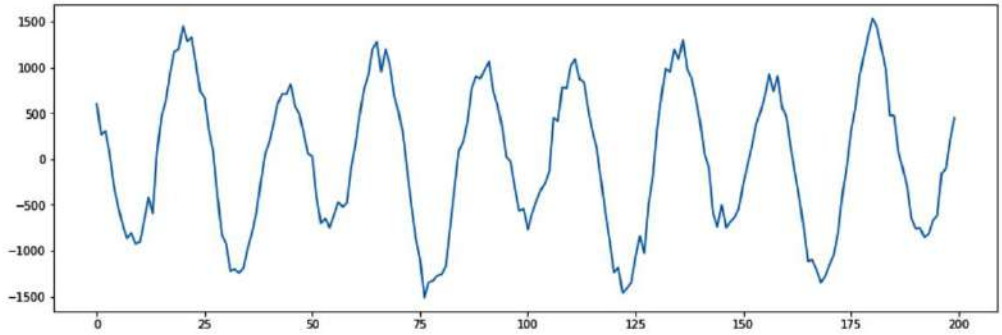
In [27]:

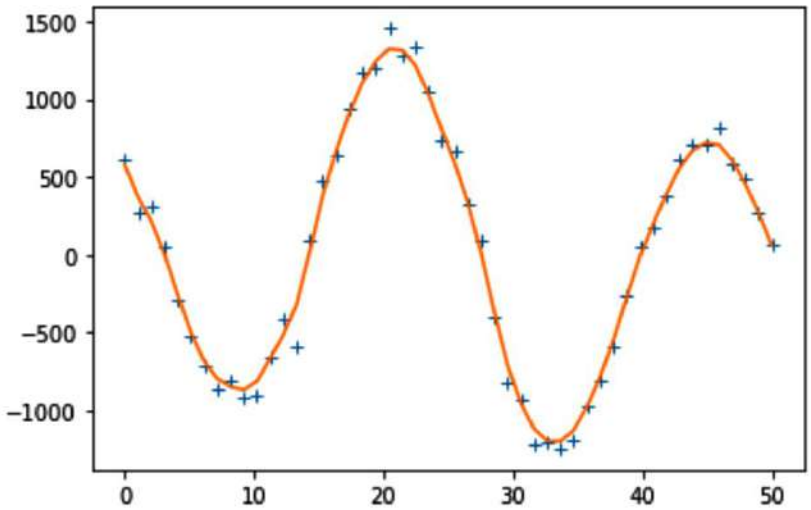
```
country_df['2020'].sort_values().plot(logy=True)
plt.xticks(rotation=90)
plt.show()
```

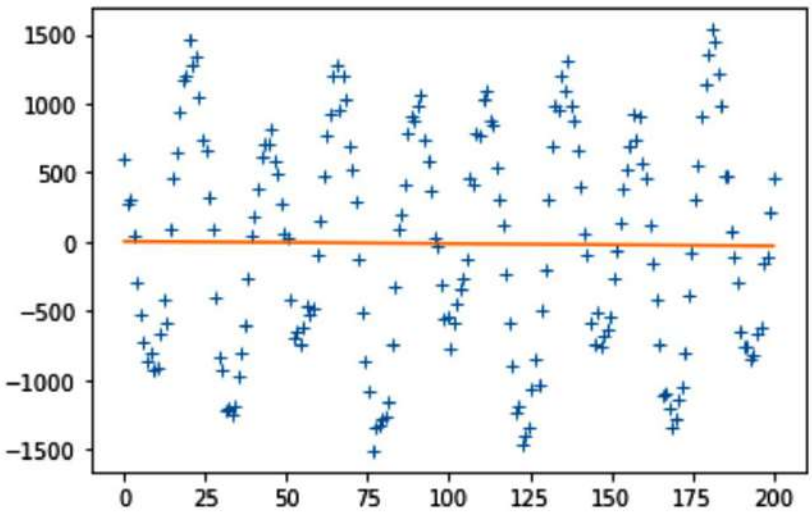


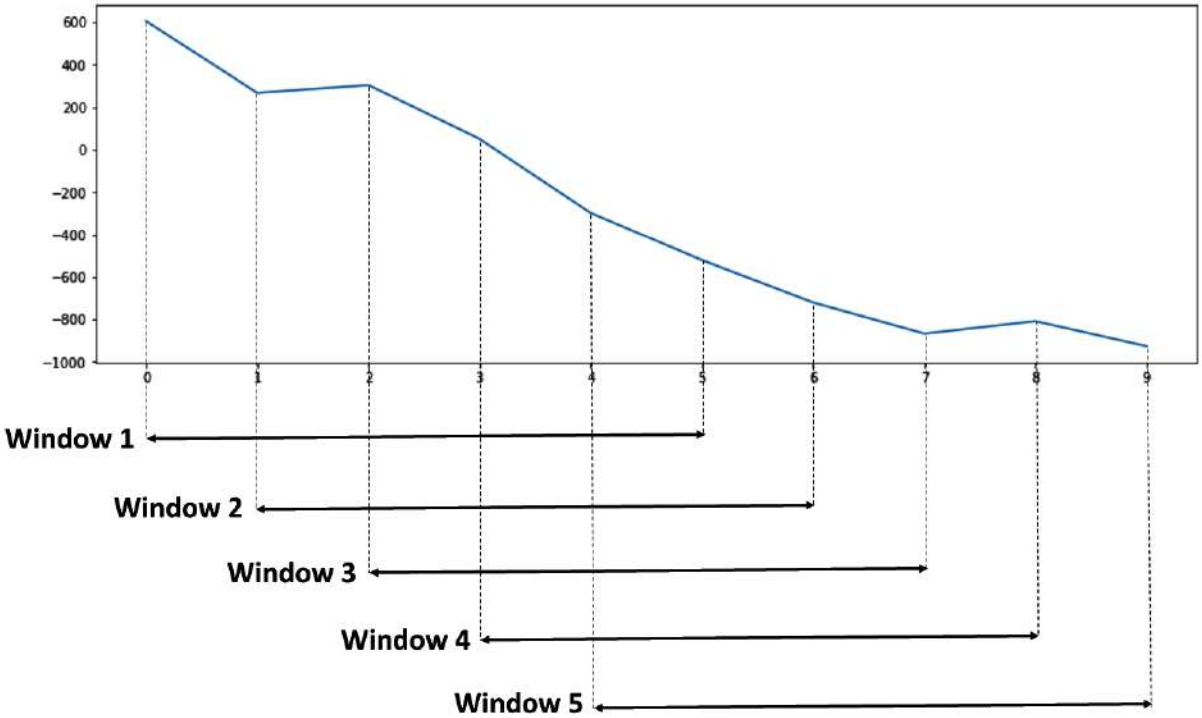
In [28]:

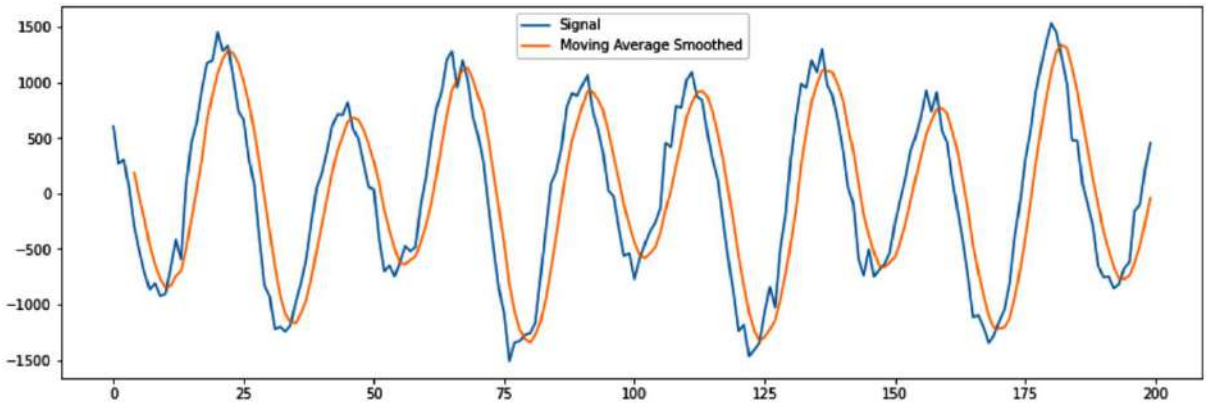
```
signal_df = pd.read_csv('Noise_data.csv')
signal_df.drop(columns='t',inplace=True)
signal_df.Signal.plot(figsize=(15,5))
plt.show()
```



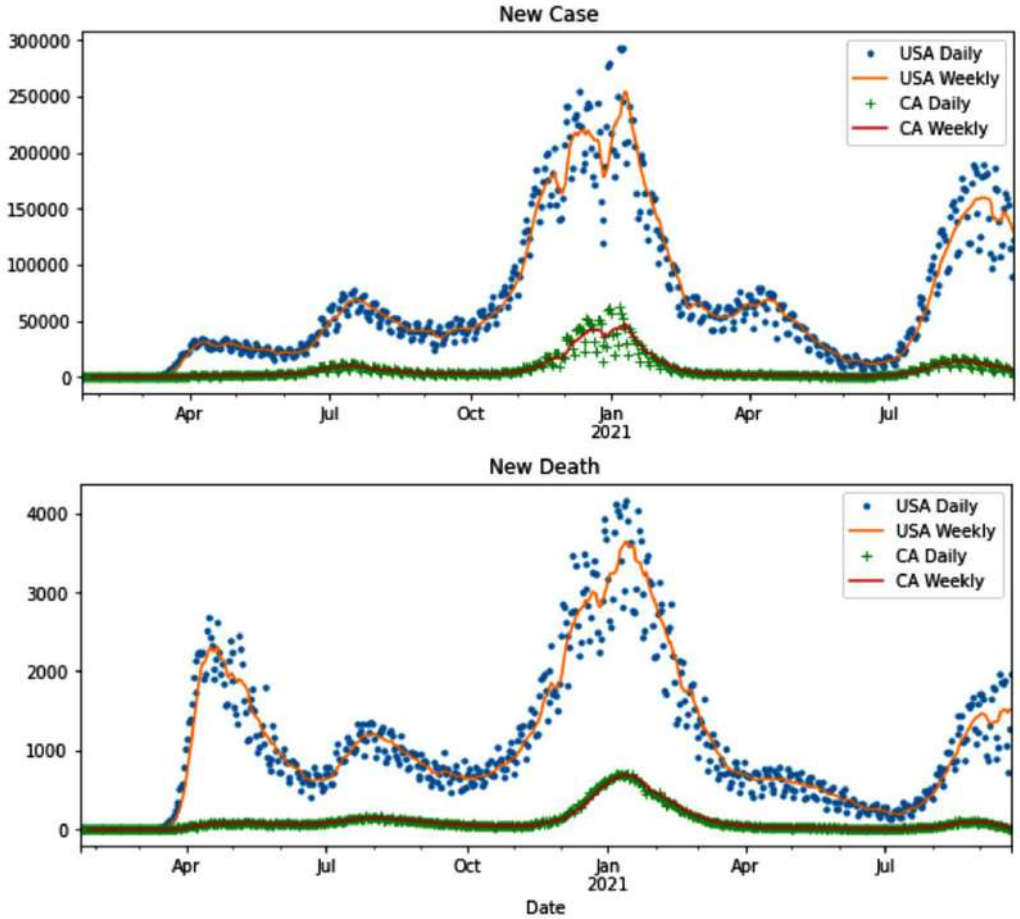





Window 1
Window 2
Window 3
Window 4
Window 5

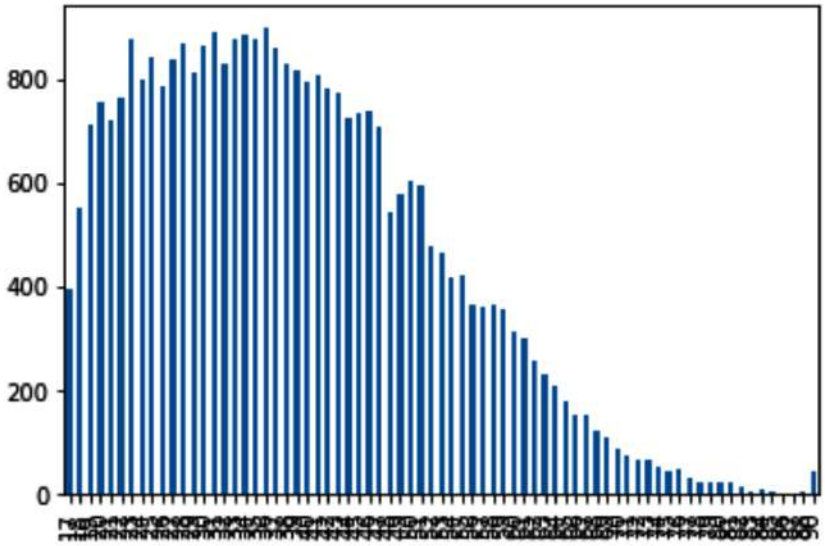

Signal
Moving Average Smoothed

	Signal	Moving Average Smoothed
0	605.340308	NaN
1	267.958658	NaN
2	304.652019	NaN
3	51.297364	NaN
4	-297.546288	186.340412
5	-520.492600	-38.826169
6	-719.919832	-236.401867
7	-866.546219	-470.641515
8	-807.907263	-642.482441
9	-925.817440	-768.136671



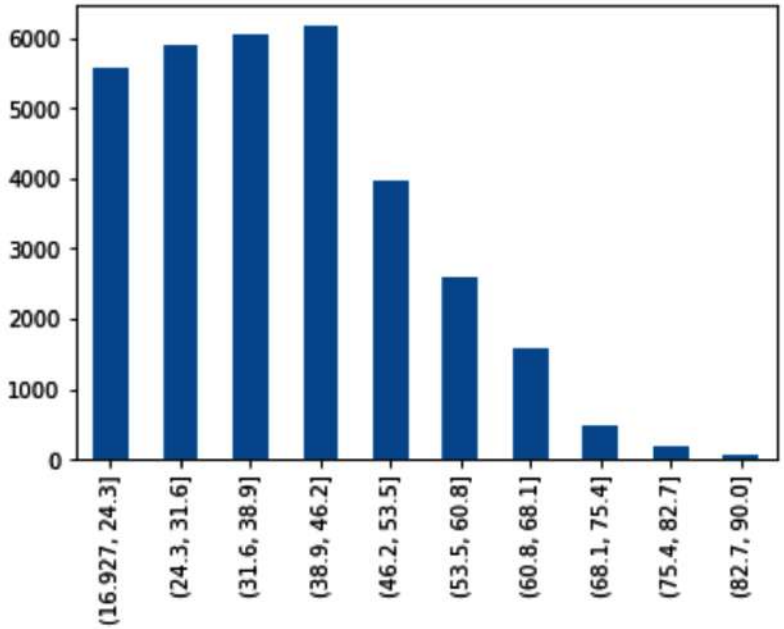
In [35]:

```
adult_df.age.value_counts().sort_index().plot.bar()
plt.show()
```



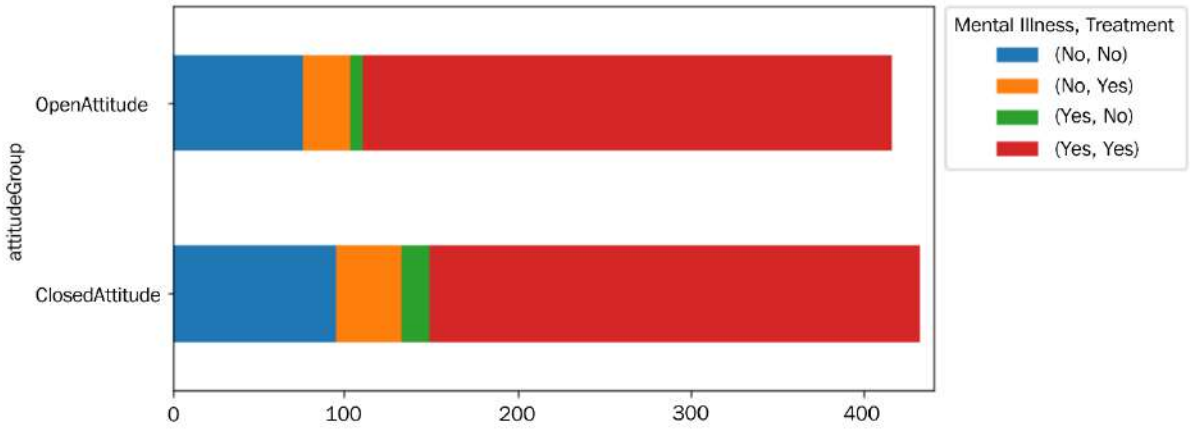
In [36]:
```
adult_df['age_binned']=pd.cut(adult_df.age,10)
adult_df.age_binned.value_counts().sort_index().plot.bar()
```

Out[36]: <AxesSubplot:>



Index	Color	index	Color	Index	Color
1	Blue	11	White	21	Orange
2	Blue	12	Orange	22	Black
3	Black	13	White	23	Yellow
4	White	14	Black	24	Black
5	Green	15	Yellow	25	Orange
6	Orange	16	Yellow	26	White
7	White	17	Blue	27	Blue
8	Blue	18	Green	28	Orange
9	Brown	19	Orange	29	Orange
10	Yellow	20	Green	30	Yellow

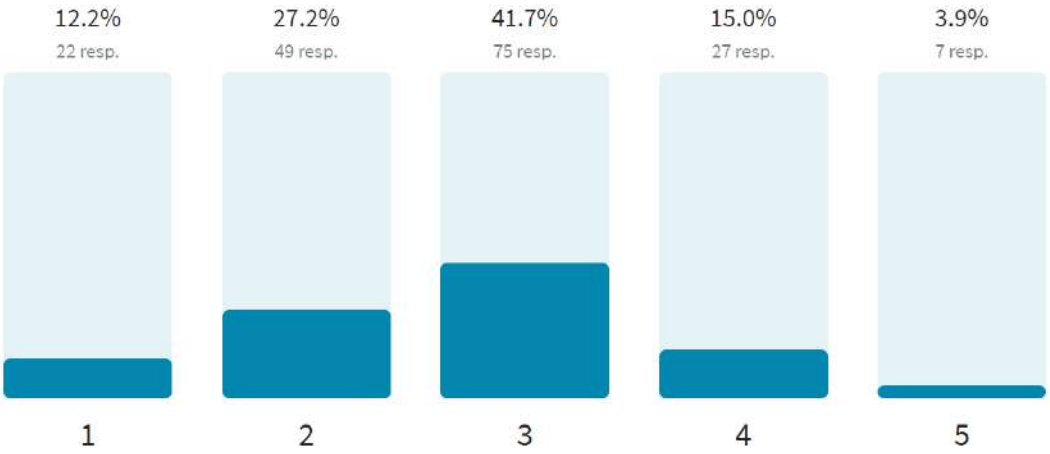
Chapter 15: Case Study 1 - Mental Health in Tech

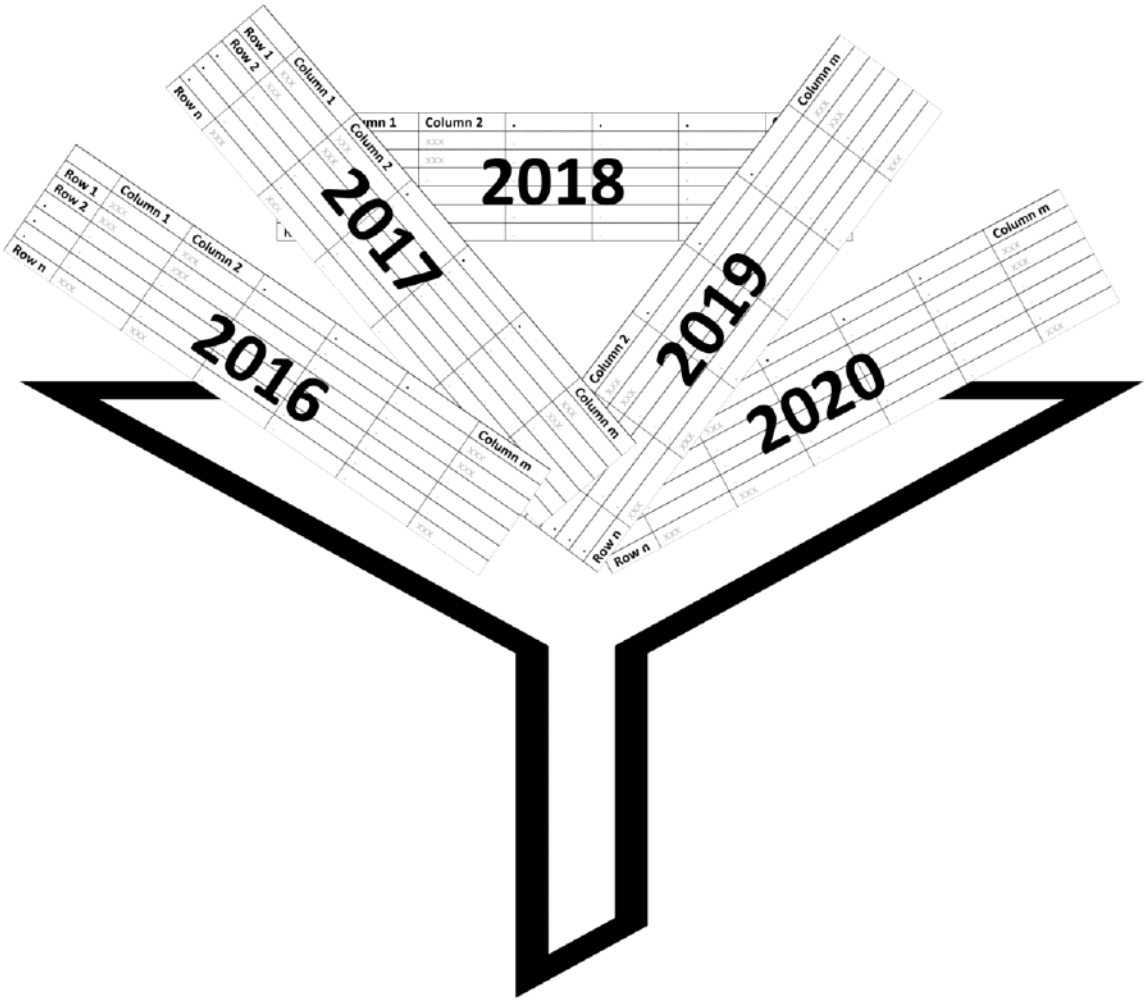


Overall, how well do you think the tech industry supports employees with mental health issues?

180 out of 180 answered

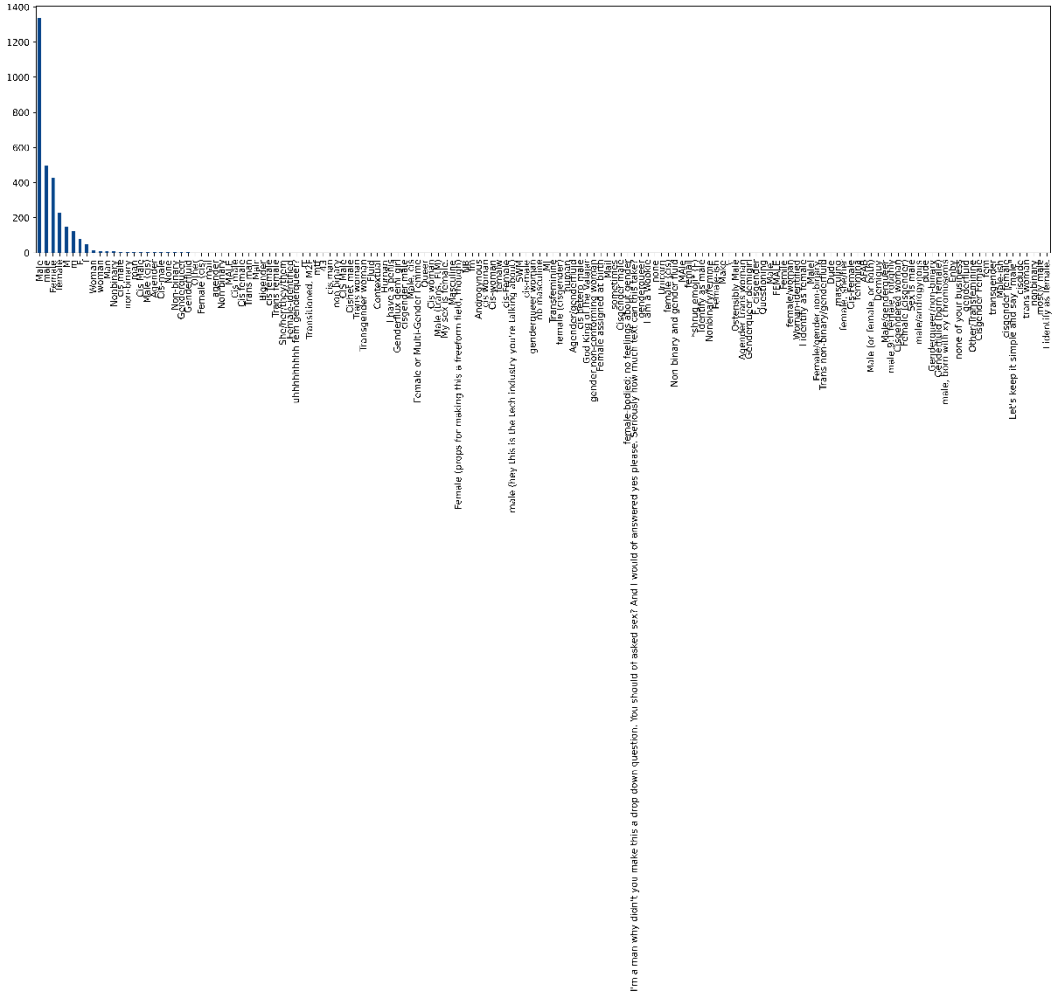
★ 2.7 Average rating

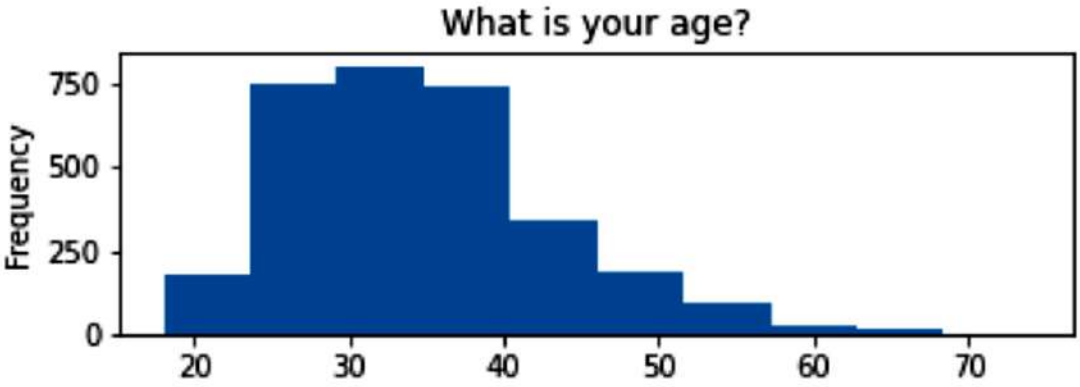


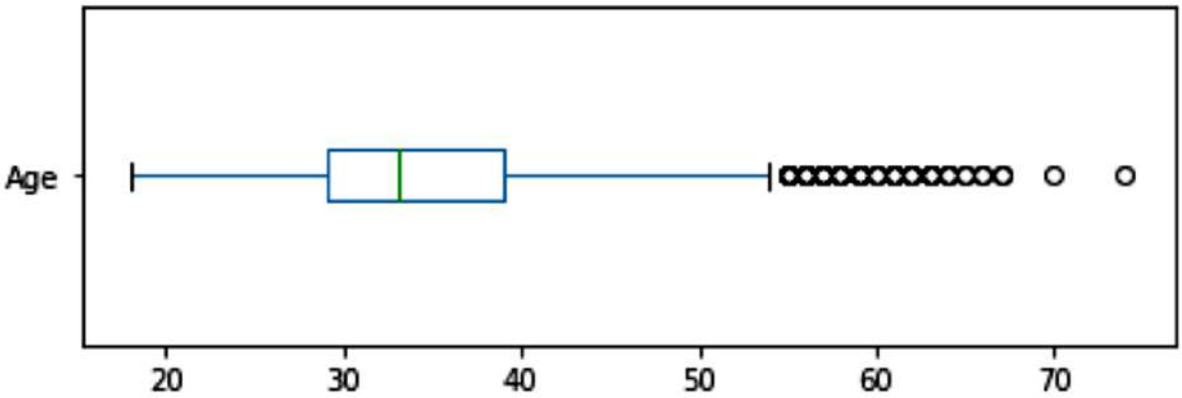


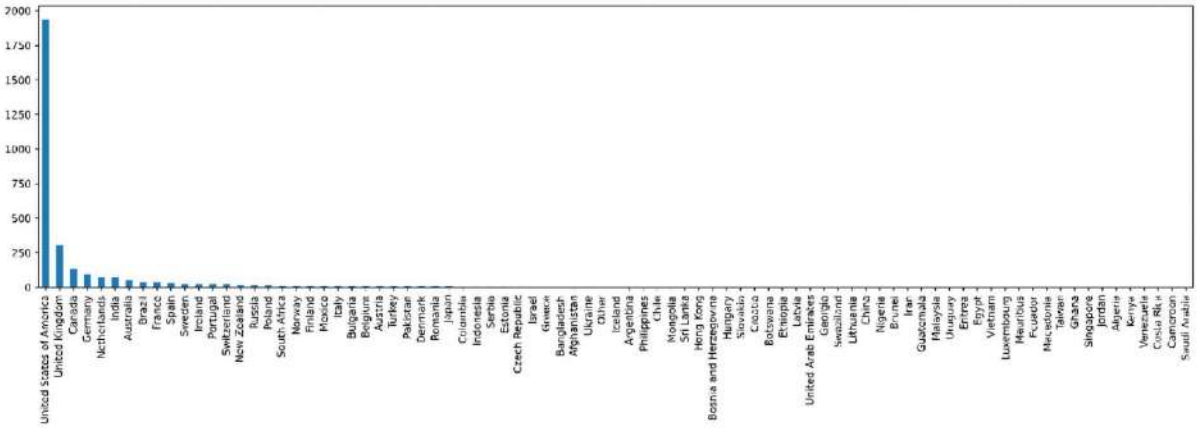
	Column 1	Column 2	.	.	.	Column m
Row 1	xxx	xxx	.	.		xxx
Row 2	xxx	xxx	.	.		xxx
.	.					.
.	.					.
.						
Row n	xxx	xxx	.	.		xxx

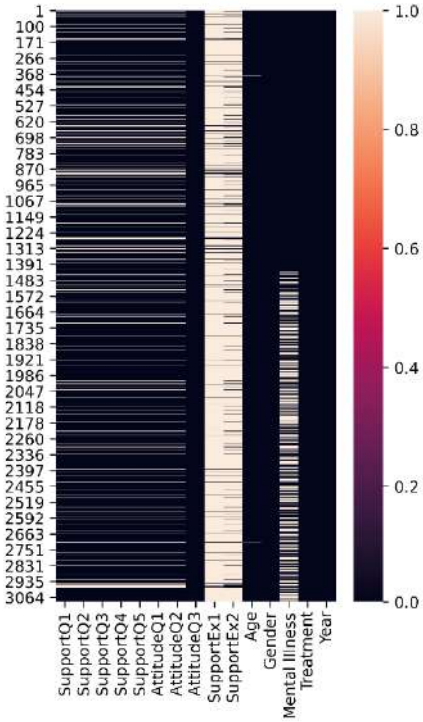
Integrated

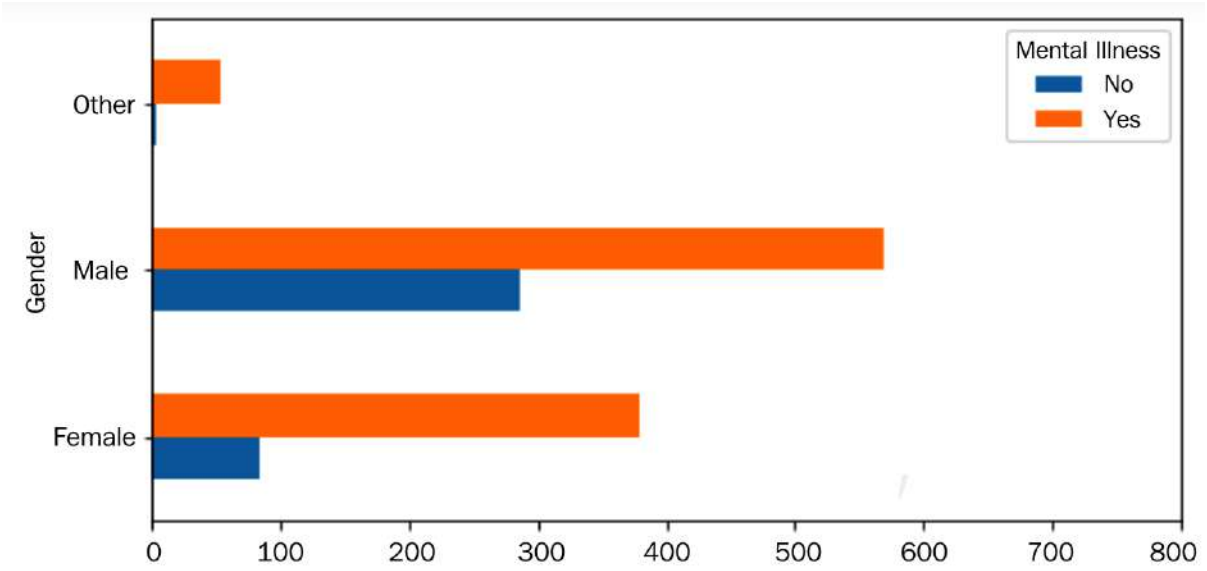

1400
1200
1000
800
600
400
200
0
Male
male
Female
female
M
F
Woman
woman
Man
man
Non-binary
cis male
non-binary
Male (cis)
Male
Cis male
Cis female
Non-binary
Genderfluid
Female (cis)
Female
Nonbinary
Cis Male
Male
Cis Female
Cis female
Trans woman
Trans female
Male-ish
Shemale
Cis man
Cis female
uhhhhhhhhh fem gender queer
Transitioned, m2f
m2f
cis male
Cis male
Cis Male
Cis Female
Trans woman
Trans woman
Female
Cis female
Male
Genderqueer
Male
Male (cis)
Female (cis)
Female
Female or Multi-Gender
Cis Male
Cis Female
Male-ish
Male (cis)
Female (cis)
Female (props for making this a freeform field, though)
Male
Male
Androgyne
Woman
Cis woman
Cis female
Female
male (hey this is the tech industry you're talking about)
Cis Male
Male
gender queer
cis male
No masculine
Male
female
Transfeminine
Male
Agender
Female
Female
God Kill Them All
Cis Female
male
gender-conforming woman
Female
Female-assigned
Male
Male
Cis Male
female bodied; no feelings about it
Male
I'm a man why didn't you make this a drop down question. You should of asked sex? And I would of answered yes please. Seriously how much fun is that?
Male
I am a woman
Male
female
Male
Non binary and genderqueer
Male
Male
Male
Male
Nonbinary/femme
Female
Male
Cisgendered Male
Agender
Male
Genderqueer demigirl
Genderqueer
Male
Male
FEMALE
Female
Female
Woman
Woman
I identify as female
Male
Female
Female
Trans non-binary/genderqueer
Male
male
Male
female
Cis Female
Female
Male (or female, but this is a terrible question)
Male
male
Male
Female
Female (cis)
Female
male
male/androgynous
Male
Male
Genderfluid (mostly female)
Male
male, born with xy chromosomes
Male
none of your business
Male
Male
Male
Male

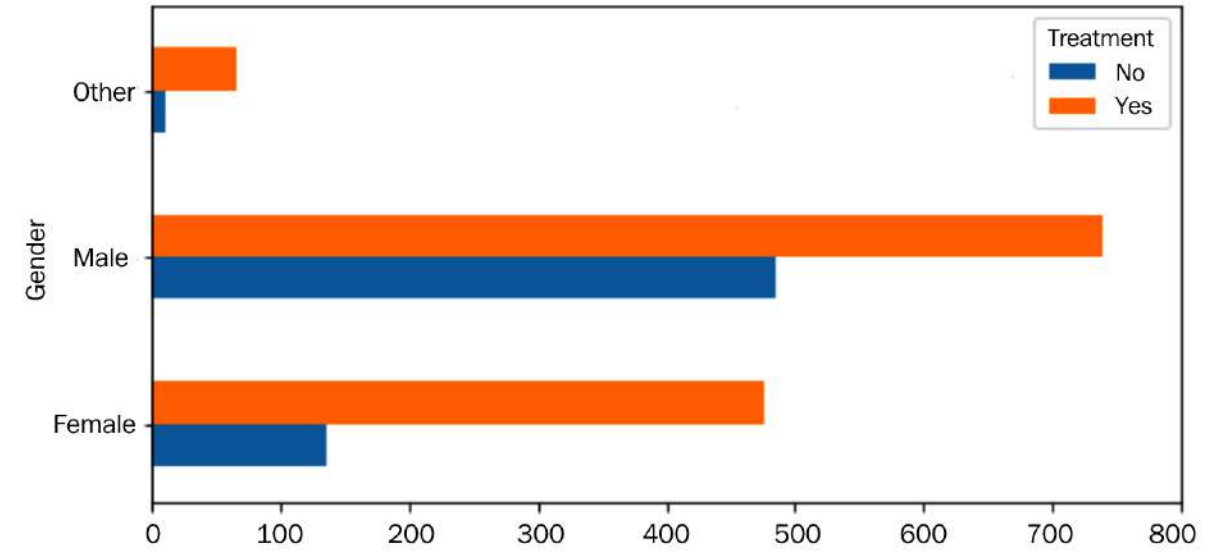

What is your age?
Frequency
750
500
250
0
20
30
40
50
60
70

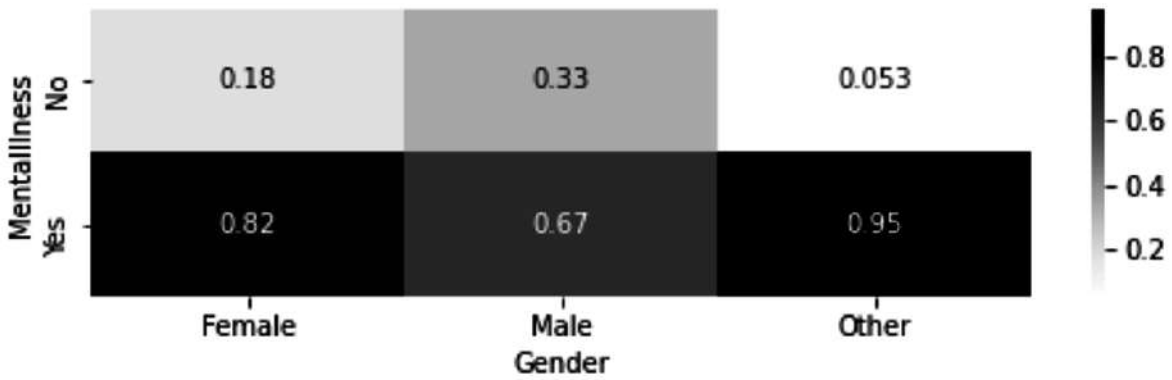

Age
20
30
40
50
60
70


2000
1750
1500
1250
1000
750
500
250
0
United States of America
United Kingdom
Canada
Germany
Netherlands
India
Australia
Brazil
France
Spain
Sweden
Ireland
Portugal
Switzerland
New Zealand
Russia
Poland
South Africa
Norway
Finland
Mexico
Italy
Bulgaria
Belgium
Austria
Turkey
Pakistan
Denmark
Romania
Japan
Jamaica
Colombia
Indonesia
Serbia
Estonia
Czech Republic
Israel
Kosovo
Bangladesh
Afghanistan
Ukraine
Oman
Iceland
Argentina
Philippines
Chile
Mongolia
Sri Lanka
Hong Kong
Bosnia and Herzegovina
Hungary
Slovakia
Costa
Botswana
Ethiopia
Latvia
United Arab Emirates
Georgia
Swaziland
Lithuania
China
Nigeria
Brunei
Iran
Guatemala
Malaysia
Uruguay
Eritrea
Egypt
Vietnam
Luxembourg
Mauritius
Ecuador
Macedonia
Taiwan
Ghana
Singapore
Jordan
Algeria
Kenya
Venezuela
Costa Rica
Cameroon
Saudi Arabia

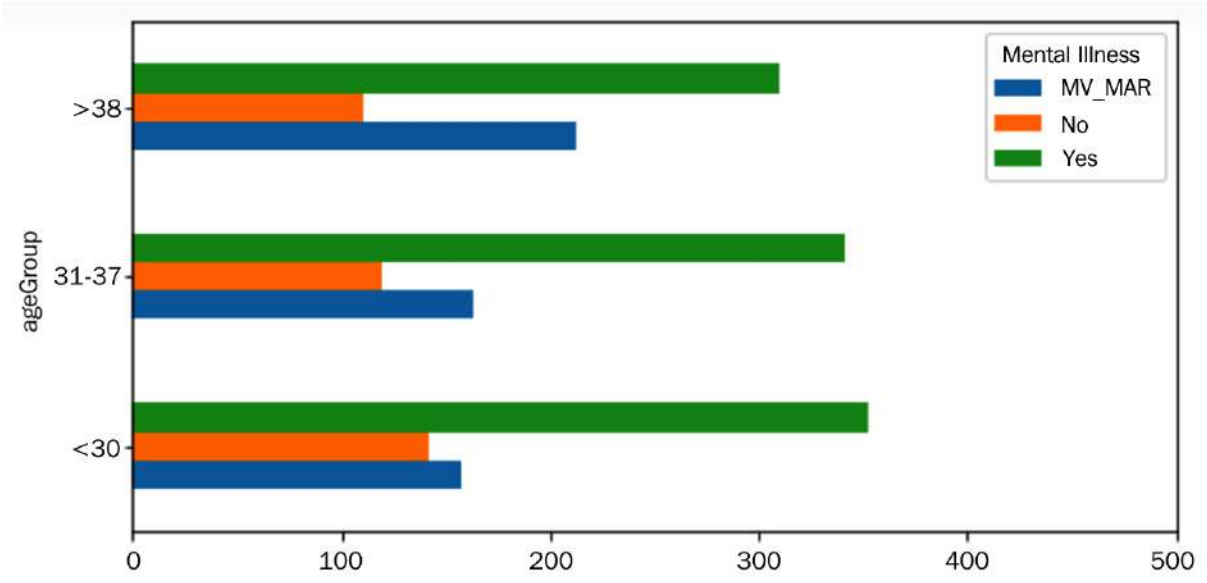

1
100
171
266
368
454
527
620
698
783
870
965
1067
1149
1224
1313
1391
1483
1572
1664
1735
1838
1921
1986
2047
2118
2178
2260
2336
2397
2455
2519
2592
2663
2751
2831
2935
3064
SupportQ1
SupportQ2
SupportQ3
SupportQ4
SupportQ5
AttitudeQ1
AttitudeQ2
AttitudeQ3
SupportEx1
SupportEx2
Age
Gender
Mental Illness
Treatment
Year
1.0
0.8
0.6
0.4
0.2
0.0

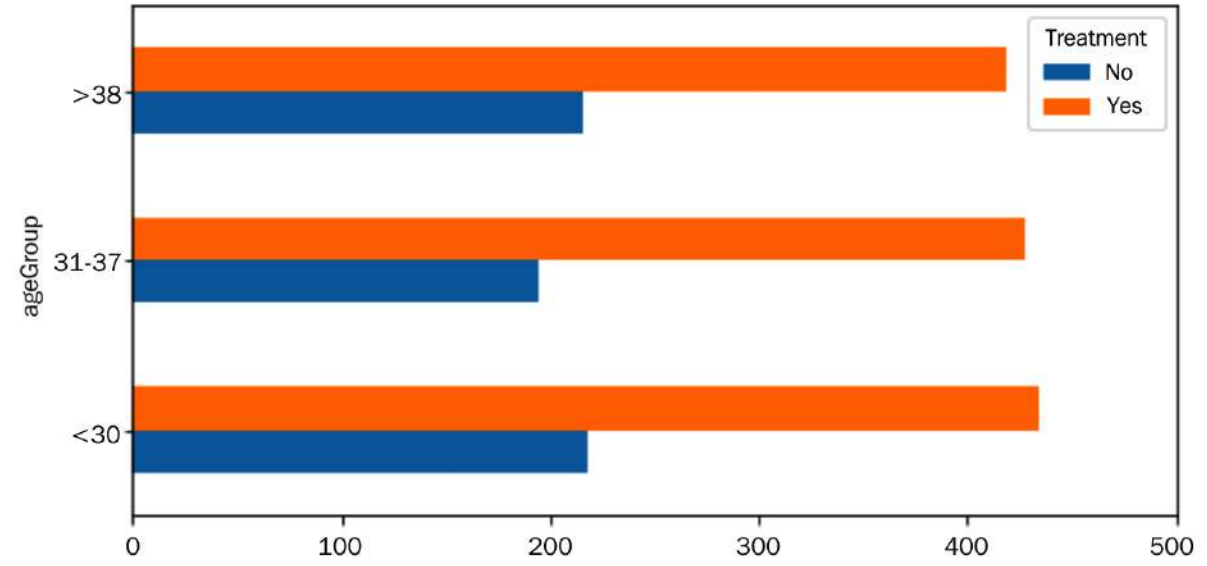

Mental Illness
No
Yes
Other
Male
Female
Gender
0
100
200
300
400
500
600
700
800

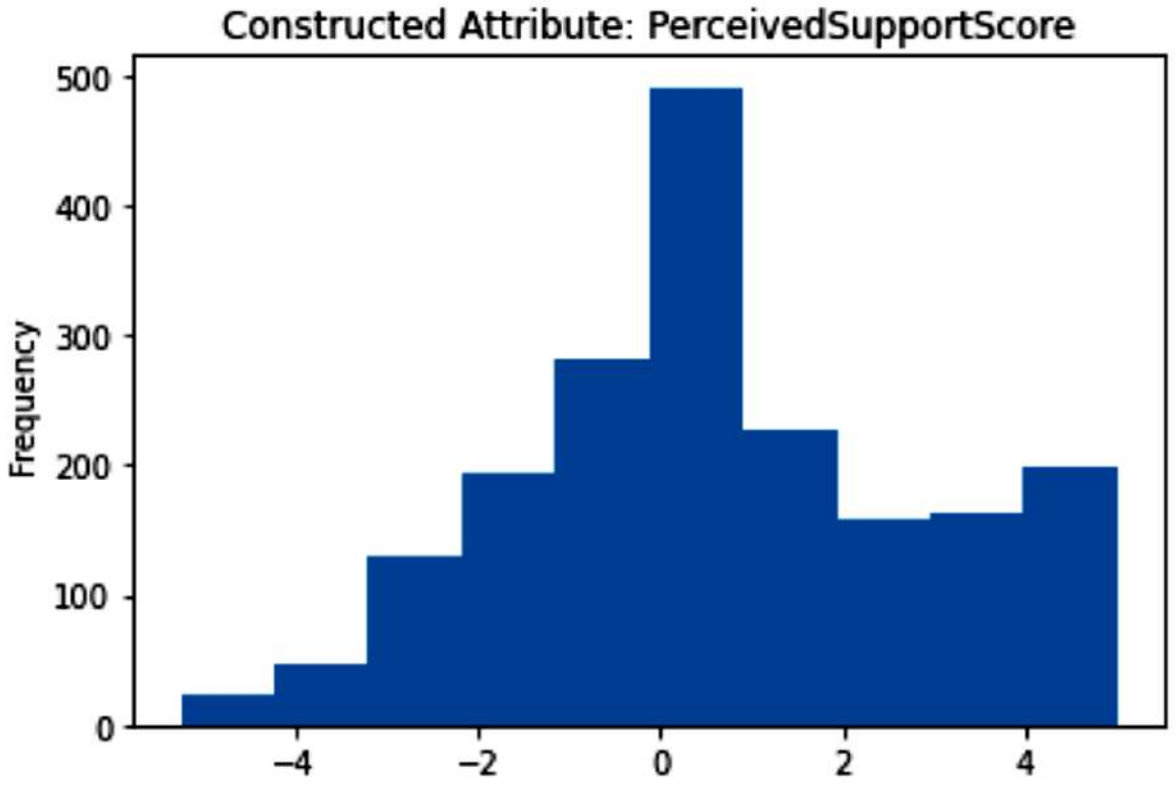

Treatment
No
Yes
Other
Male
Female
Gender
0
100
200
300
400
500
600
700
800

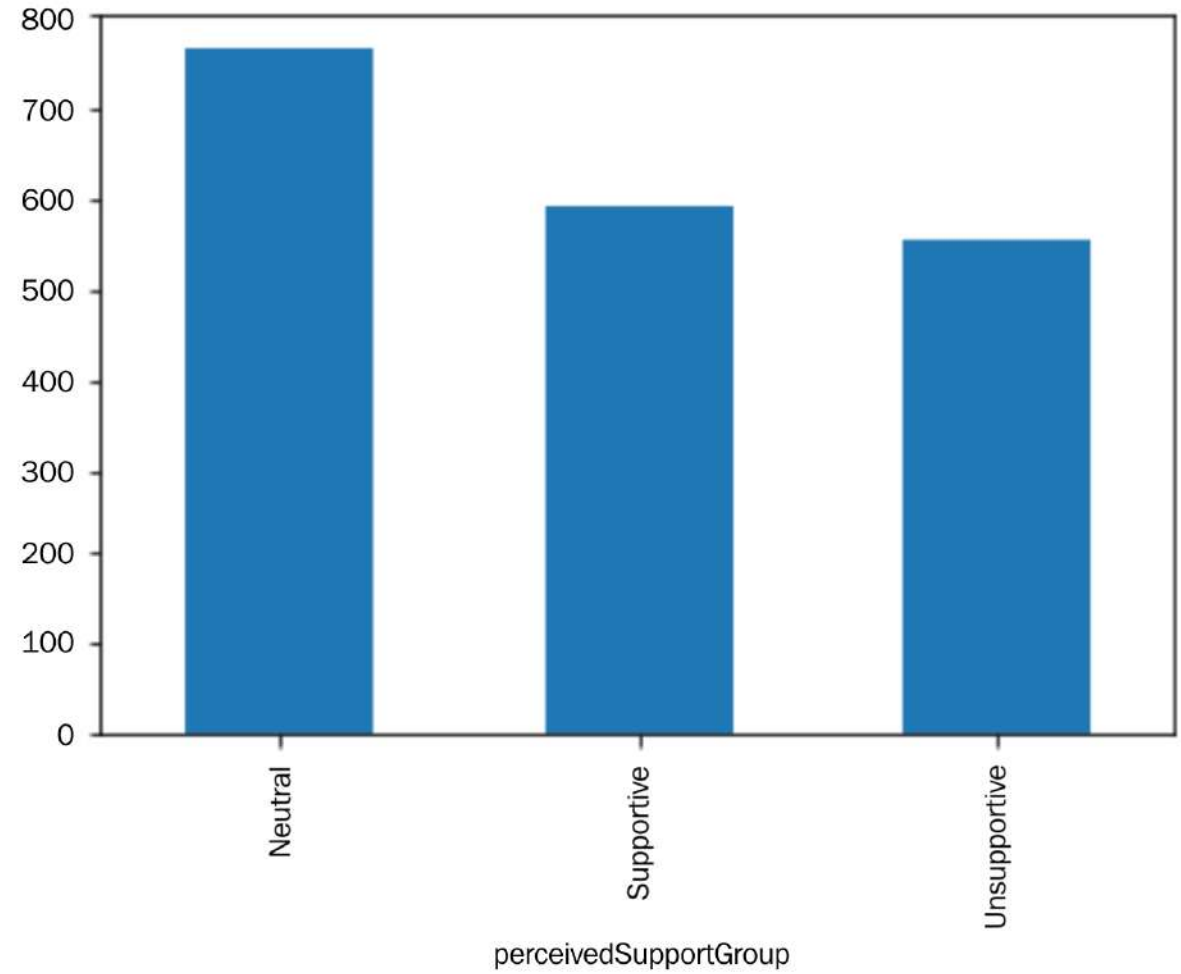

Mentalliness
No
Yes
0.18
0.33
0.053
0.82
0.67
0.95
Female
Male
Other
Gender
0.8
0.6
0.4
0.2

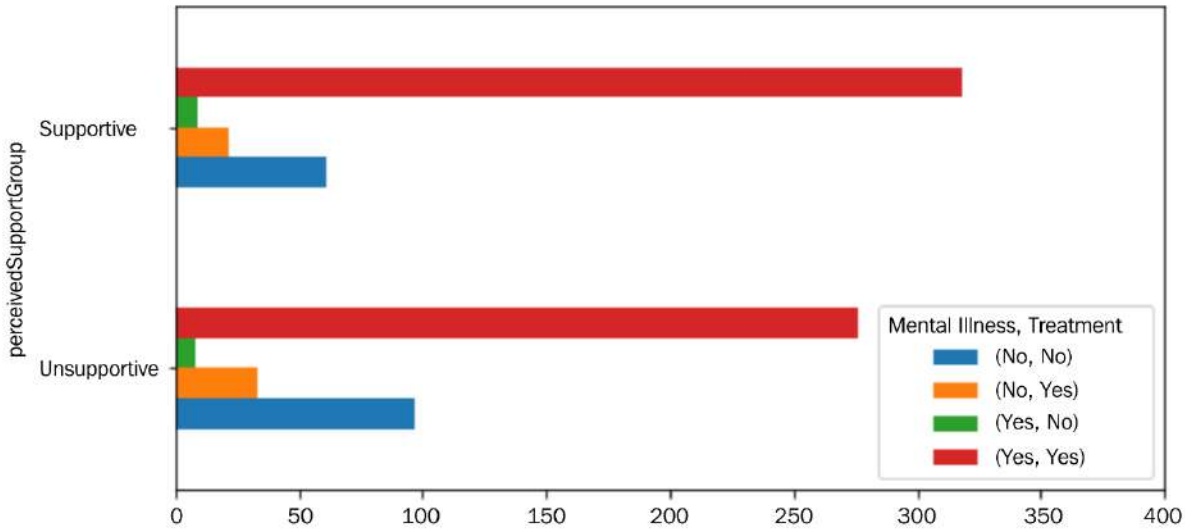
Treatment
No
Yes
0.22
0.4
0.14
0.78
0.6
0.86
Female
Male
Other
Gender
0.8
0.6
0.4
0.2

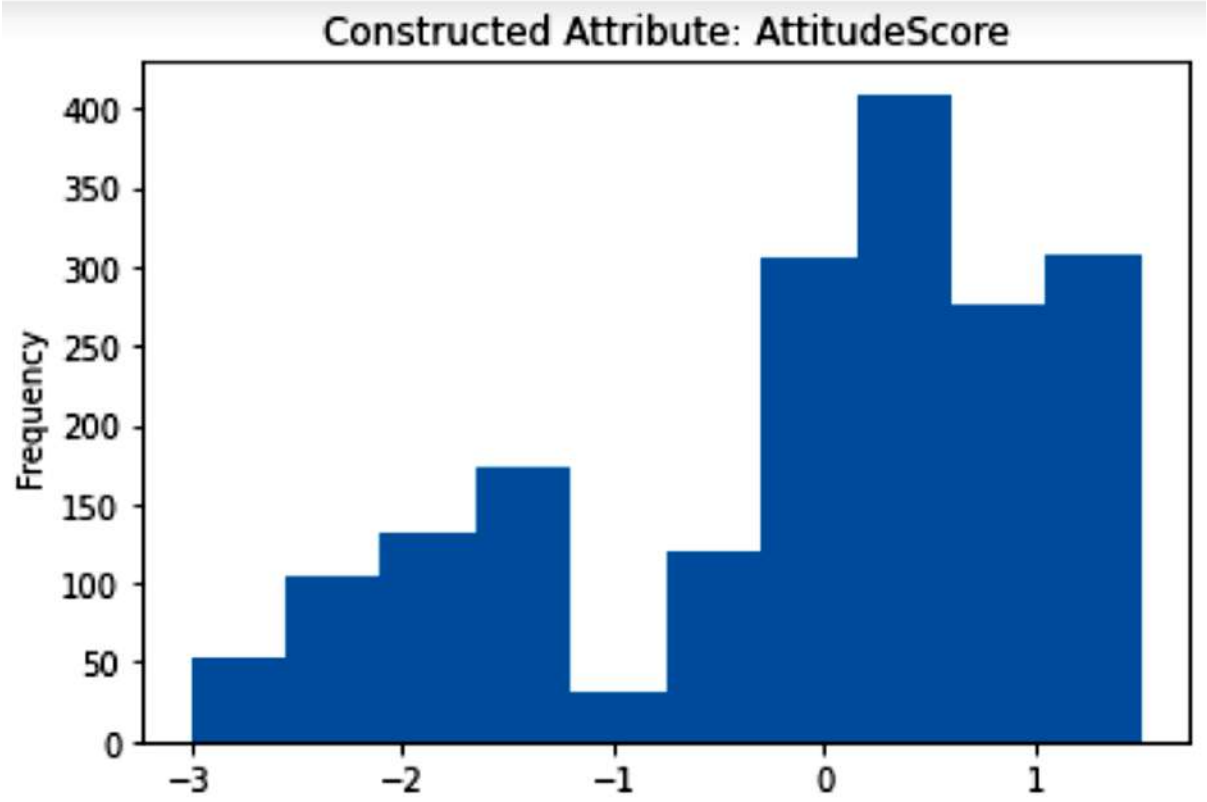

Mental Illness
MV_MAR
No
Yes
ageGroup
>38
31-37
<30
0
100
200
300
400
500

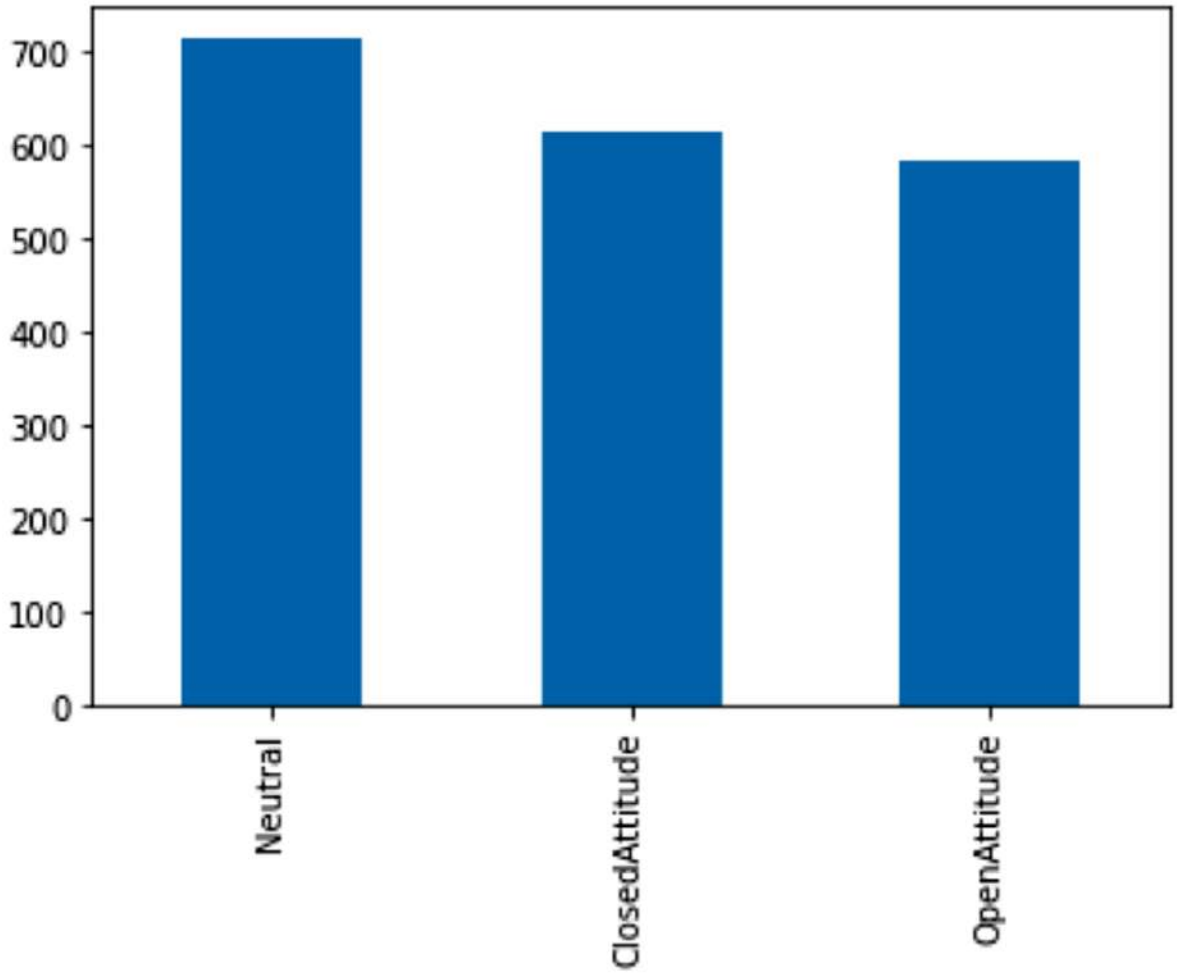

Treatment
No
Yes
ageGroup
>38
31-37
<30
0
100
200
300
400
500


Constructed Attribute: PerceivedSupportScore
Frequency
500
400
300
200
100
0
−4
−2
0
2
4

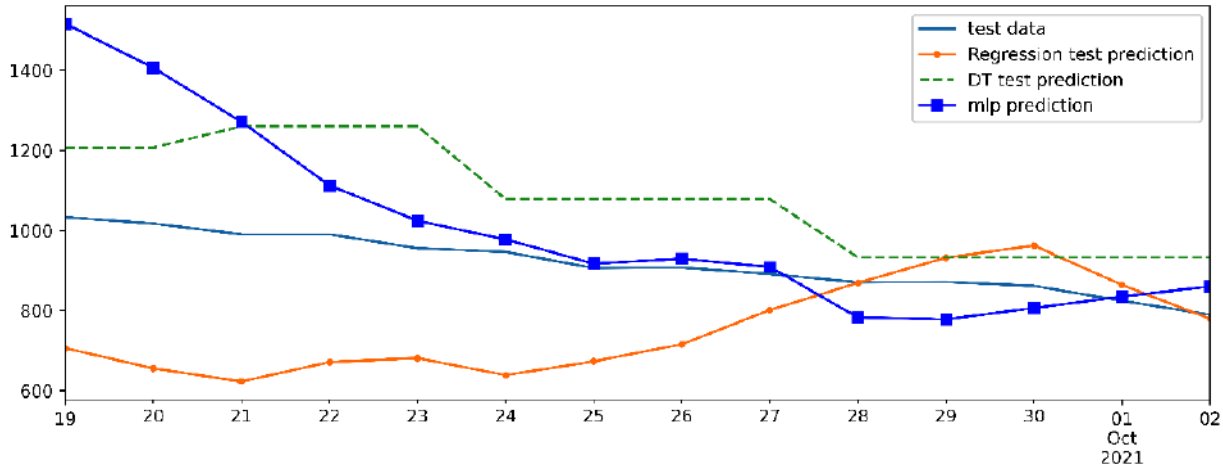

800
700
600
500
400
300
200
100
0
Neutral
Supportive
Unsupportive
perceivedSupportGroup


perceivedSupportGroup
Supportive
Unsupportive
Mental Illness, Treatment
(No, No)
(No, Yes)
(Yes, No)
(Yes, Yes)
0
50
100
150
200
250
300
350
400


Constructed Attribute: AttitudeScore
Frequency
400
350
300
250
200
150
100
50
0
−3
−2
−1
0
1


700
600
500
400
300
200
100
0
Neutral
ClosedAttitude
OpenAttitude

Chapter 16: Case Study 2 - Predicting COVID-19 Hospitalizations



LA County Daily COVID-19 Data

View Other Data Pages

Data through 6:00pm 10/02/2021

Cases

1,032

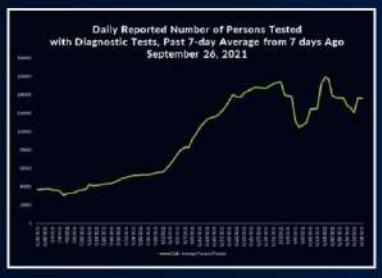
New Cases Reported (10/03)*

1,463,039

Total Cases Reported*

**including cases reported by Long Beach and Pasadena Health Departments*

Testing

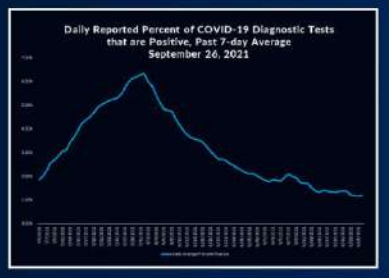


7-Day Daily Average: 135,294

Total Number of People Tested: 8,683,814

What This Means

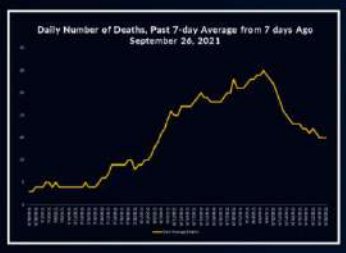
Testing Positivity Rate



7-Day Daily Average: 1.19%

What This Means

Deaths



New Deaths Reported (10/03)*: 10

Total Deaths Reported*: 26,153

**including deaths reported by Long Beach and Pasadena Health Departments*

What This Means

Death Rate

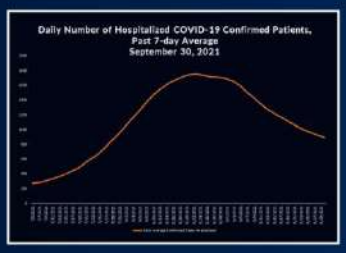
Age-Adjusted Death Rates due to COVID-19 per 100K
October 02, 2021

		Mortality Rate
Los Angeles County Total		238
Race/Ethnicity	Asian	267
	Black/African American	237
	Hispanic/Latino	583
	White	135
Area Poverty	<10% area poverty	226
	10% to <20% area poverty	246
	20% to <30% area poverty	324
	30% to 100% area poverty	436

by Race, Ethnicity and Poverty Level

What This Means

Hospitalizations

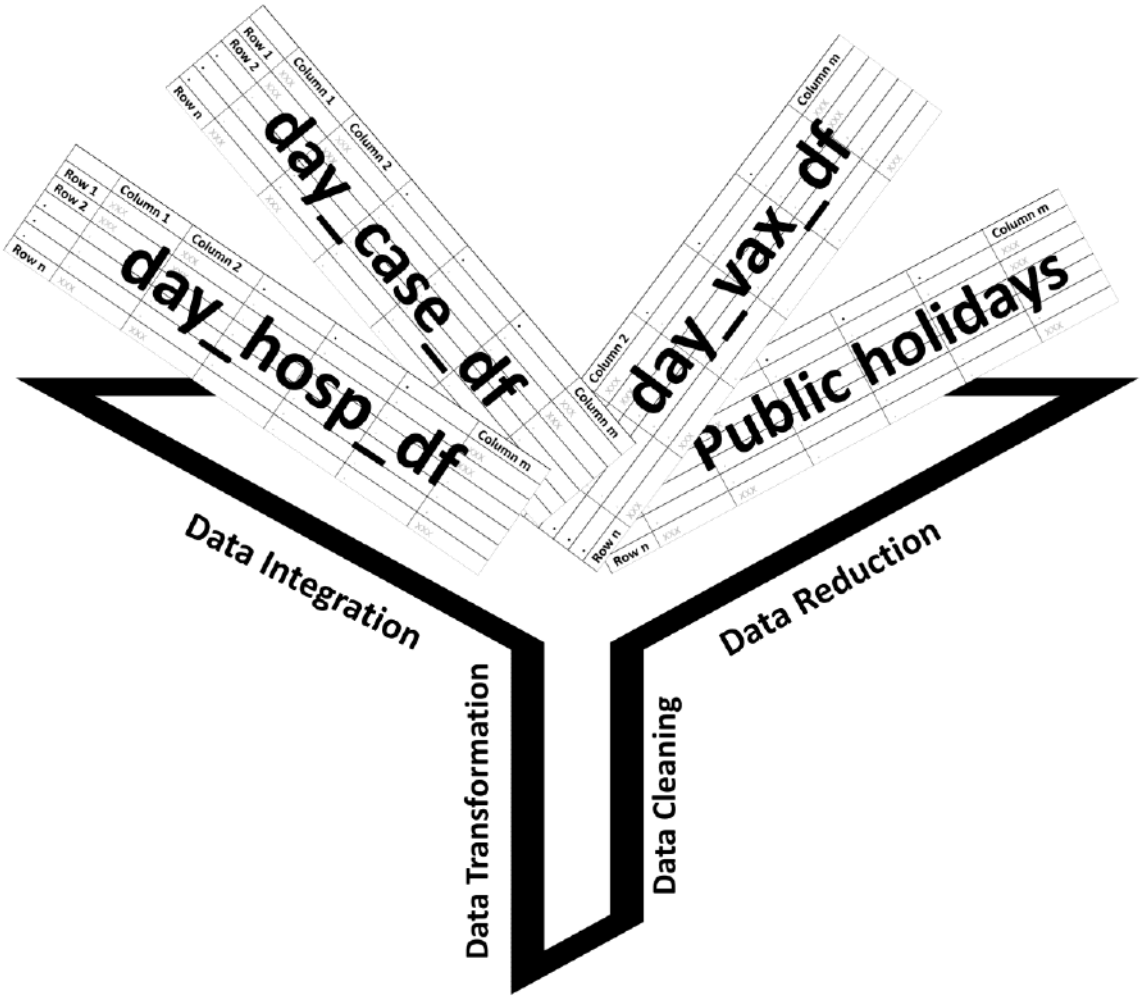


Current Hospitalizations (10/1): 825

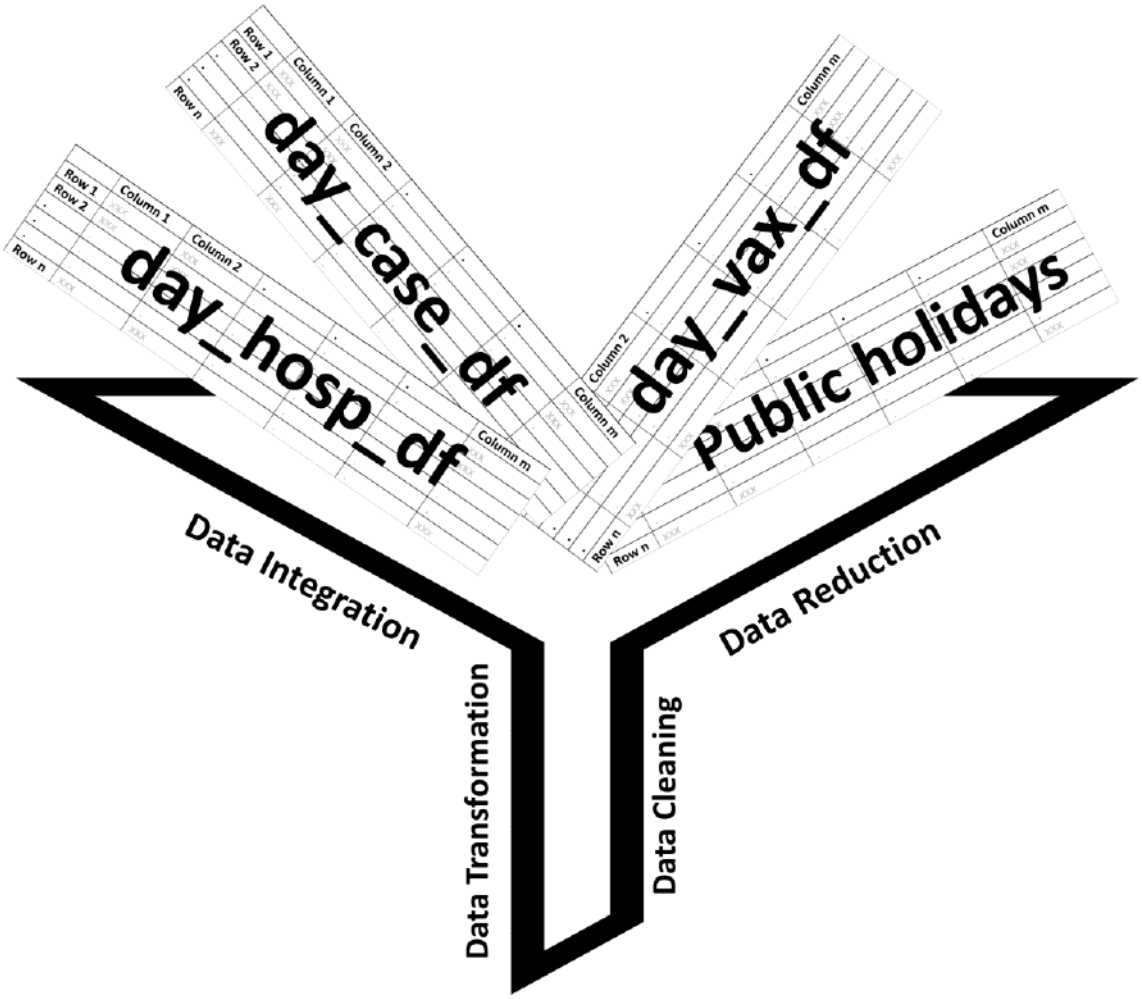
What This Means

	t0	n_Hosp_t0	s_Hosp_tn7_0	n_days_MajHol	av7_Case_tn6_0	s_Case_tn14_0	av7_Death_tn6_0	s_Death_tn14_0	p_FullVax_t0	s_FullVax_tn14_0	n_Hosp_t14
0	2020-03-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2020-03-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2020-03-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2020-04-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2020-04-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...
549	2021-09-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
550	2021-09-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
551	2021-10-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
552	2021-10-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
553	2021-10-03	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

554 rows × 11 columns



	t0	n_Hosp_t0	s_Hosp_tn7_0	n_days_MajHol	av7_Case_tn6_0	s_Case_tn14_0	av7_Death_tn6_0	s_Death_tn14_0	p_FullVax_t0	s_FullVax_tn14_0	n_Hosp_t14
0	2020-03-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2020-03-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2020-03-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2020-04-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2020-04-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...



	t0	n_Hosp_t0	s_Hosp_tn7_0	n_days_MajHol	av7_Case_tn6_0	s_Case_tn14_0	av7_Death_tn6_0	s_Death_tn14_0	p_FullVax_t0	s_FullVax_tn14_0	n_Hosp_t14
0	2020-03-29	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	2020-03-30	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	2020-03-31	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	2020-04-01	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	2020-04-02	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...

	t0	n_Hosp_t0	s_Hosp_tn7_0	n_days_MajHol	av7_Case_tn6_0	s_Case_tn14_0	av7_Death_tn6_0	s_Death_tn14_0	p_FullVax_t0	s_FullVax_tn14_0	n_Hosp_t14
t0											
2020-03-29	2020-03-29	489.0	NaN	67	NaN	NaN	NaN	NaN	0.000000	0.000000	1433.0
2020-03-30	2020-03-30	601.0	NaN	68	NaN	NaN	NaN	NaN	0.000000	0.000000	1501.0
2020-03-31	2020-03-31	713.0	NaN	69	NaN	NaN	NaN	NaN	0.000000	0.000000	1587.0
2020-04-01	2020-04-01	739.0	NaN	70	NaN	NaN	NaN	NaN	0.000000	0.000000	1624.0
2020-04-02	2020-04-02	818.0	NaN	71	NaN	NaN	NaN	NaN	0.000000	0.000000	1679.0
...	...	...	...	...	...	...	...	...	...	...	...
2021-09-29	2021-09-29	872.0	-16.928571	21	970.285714	-34.732143	13.000000	-0.528571	0.712213	0.000669	NaN
2021-09-30	2021-09-30	862.0	-13.773810	22	954.142857	-21.414284	11.857143	-0.814286	0.712213	0.000623	NaN
2021-10-01	2021-10-01	825.0	-14.380952	23	897.857143	-25.339285	9.714286	-1.021429	0.712213	0.000566	NaN
2021-10-02	2021-10-02	790.0	-15.750000	24	804.571429	-42.971429	7.857143	-1.078571	0.712213	0.000503	NaN
2021-10-03	2021-10-03	768.0	-19.500000	25	737.285714	-67.714286	6.000000	-1.228571	0.712213	0.000437	NaN

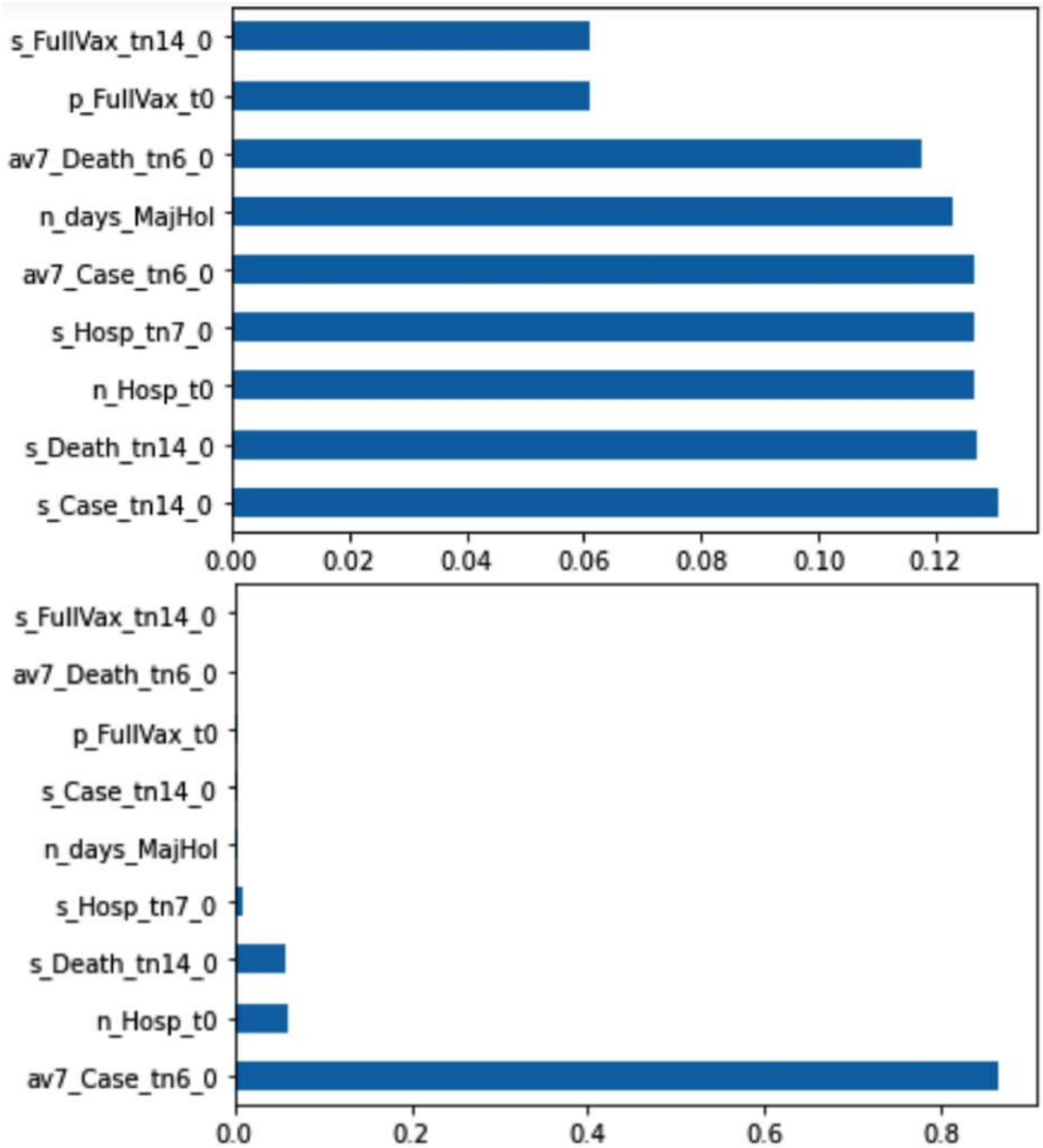
554 rows × 11 columns

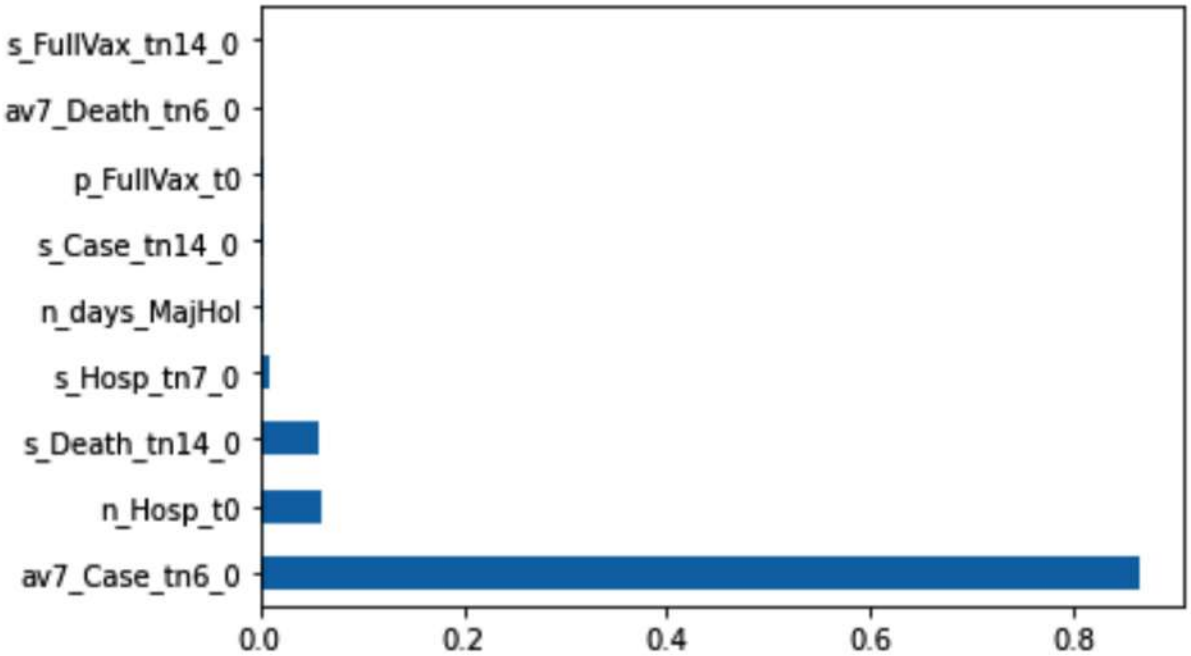
OLS Regression Results

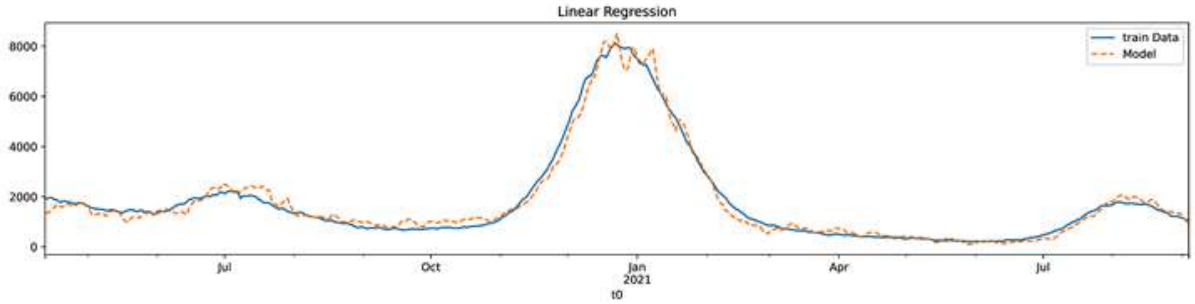
Dep. Variable:	n_Hosp_t14	**R-squared:**	0.981
Model:	OLS	**Adj. R-squared:**	0.981
Method:	Least Squares	**F-statistic:**	2937.
Date:	Mon, 04 Oct 2021	**Prob (F-statistic):**	0.00
Time:	10:03:07	**Log-Likelihood:**	-3653.5
No. Observations:	525	**AIC:**	7327.
Df Residuals:	515	**BIC:**	7370.
Df Model:	9		
Covariance Type:	nonrobust		

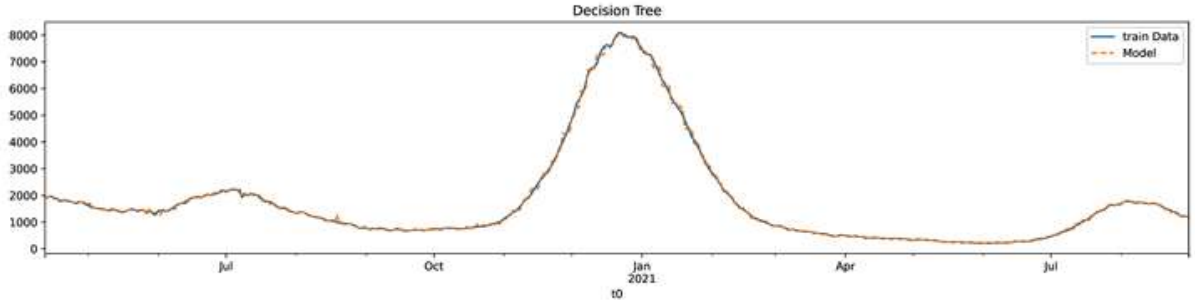
	coef	**std err**	**t**	**P>|t|**	**[0.025**	**0.975]**
const	316.7108	35.260	8.982	0.000	247.439	385.983
n_Hosp_t0	0.6368	0.053	12.043	0.000	0.533	0.741
s_Hosp_tn7_0	8.7896	0.643	13.672	0.000	7.527	10.053
n_days_MajHol	0.6326	0.422	1.499	0.135	-0.197	1.462
av7_Case_tn6_0	0.2514	0.017	14.956	0.000	0.218	0.284
s_Case_tn14_0	0.5128	0.130	3.932	0.000	0.257	0.769
av7_Death_tn6_0	-5.3348	1.324	-4.030	0.000	-7.935	-2.734
s_Death_tn14_0	-132.9086	12.351	-10.761	0.000	-157.172	-108.645
p_FullVax_t0	-496.0640	57.139	-8.682	0.000	-608.319	-383.809
s_FullVax_tn14_0	-7834.8102	8950.211	-0.875	0.382	-2.54e+04	9748.603

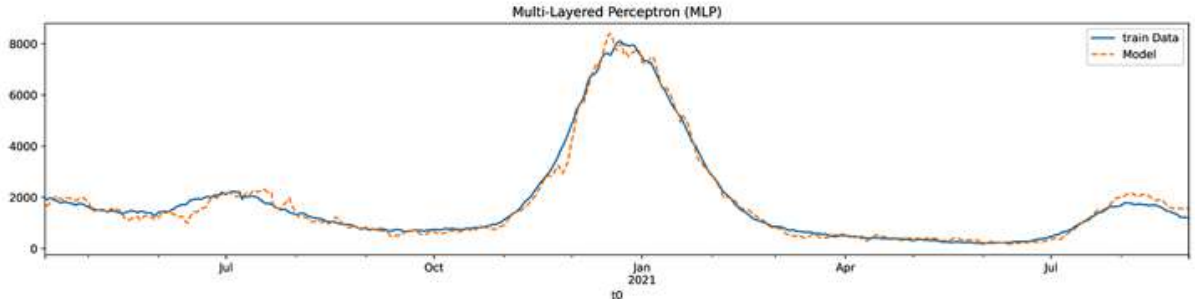
Omnibus:	27.076	**Durbin-Watson:**	0.168
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	67.116
Skew:	0.213	**Prob(JB):**	2.67e-15
Kurtosis:	4.699	**Cond. No.**	4.00e+06

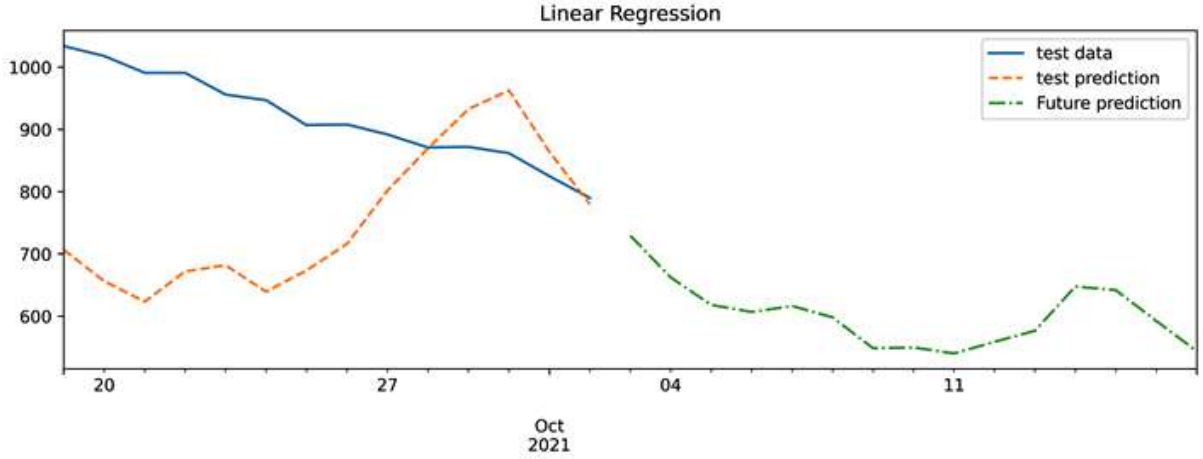

s_FullVax_tn14_0
p_FullVax_t0
av7_Death_tn6_0
n_days_MajHol
av7_Case_tn6_0
s_Hosp_tn7_0
n_Hosp_t0
s_Death_tn14_0
s_Case_tn14_0
0.00
0.02
0.04
0.06
0.08
0.10
0.12
s_FullVax_tn14_0
av7_Death_tn6_0
p_FullVax_t0
s_Case_tn14_0
n_days_MajHol
s_Hosp_tn7_0
s_Death_tn14_0
n_Hosp_t0
av7_Case_tn6_0
0.0
0.2
0.4
0.6
0.8

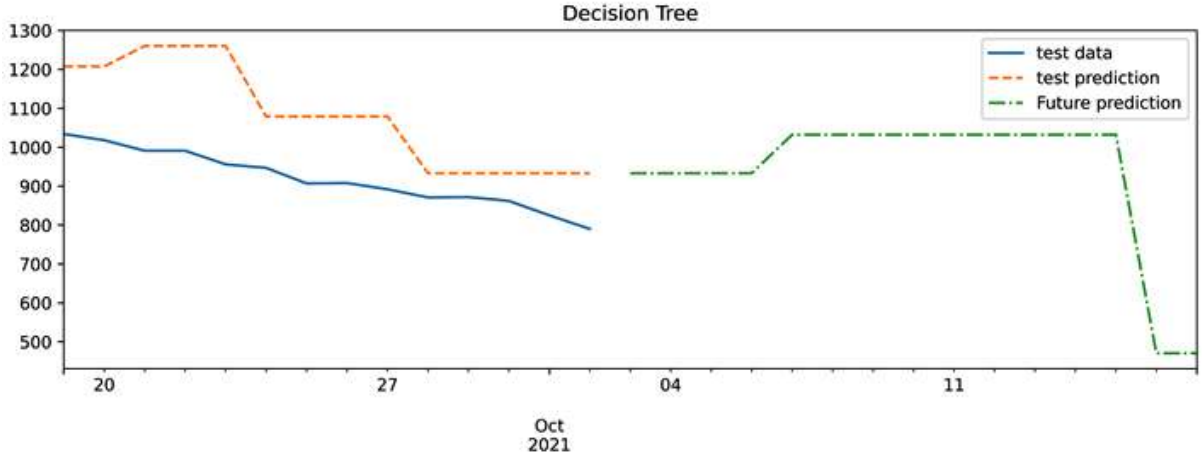

s_FullVax_tn14_0
av7_Death_tn6_0
p_FullVax_t0
s_Case_tn14_0
n_days_MajHol
s_Hosp_tn7_0
s_Death_tn14_0
n_Hosp_t0
av7_Case_tn6_0
0.0
0.2
0.4
0.6
0.8

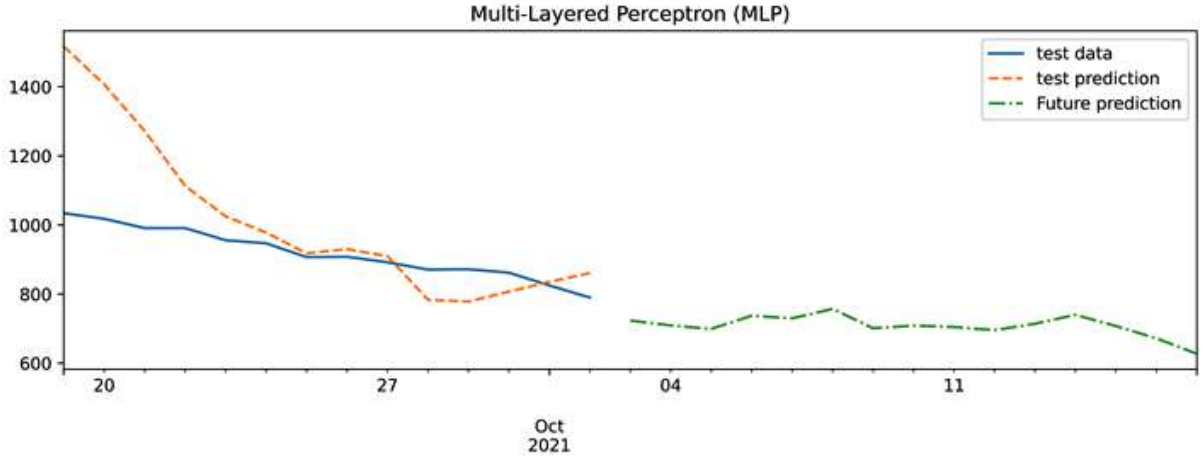

Linear Regression
train Data
Model
8000
6000
4000
2000
0
Jul
Oct
Jan
2021
Apr
Jul
t0


Decision Tree
train Data
Model
8000
7000
6000
5000
4000
3000
2000
1000
0
Jul
Oct
Jan
2021
Apr
Jul
t0

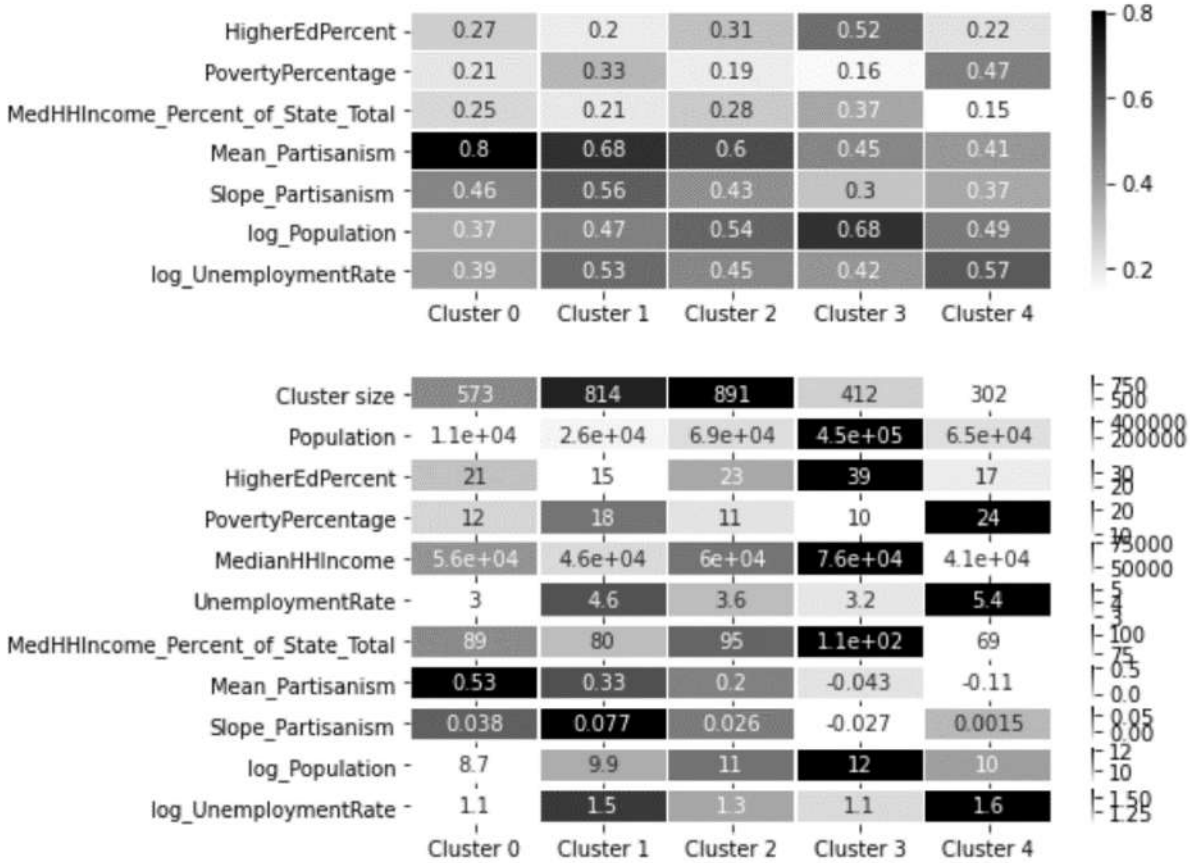

Multi-Layered Perceptron (MLP)
train Data
Model
8000
6000
4000
2000
0
Jul
Oct
Jan
2021
Apr
Jul
t0

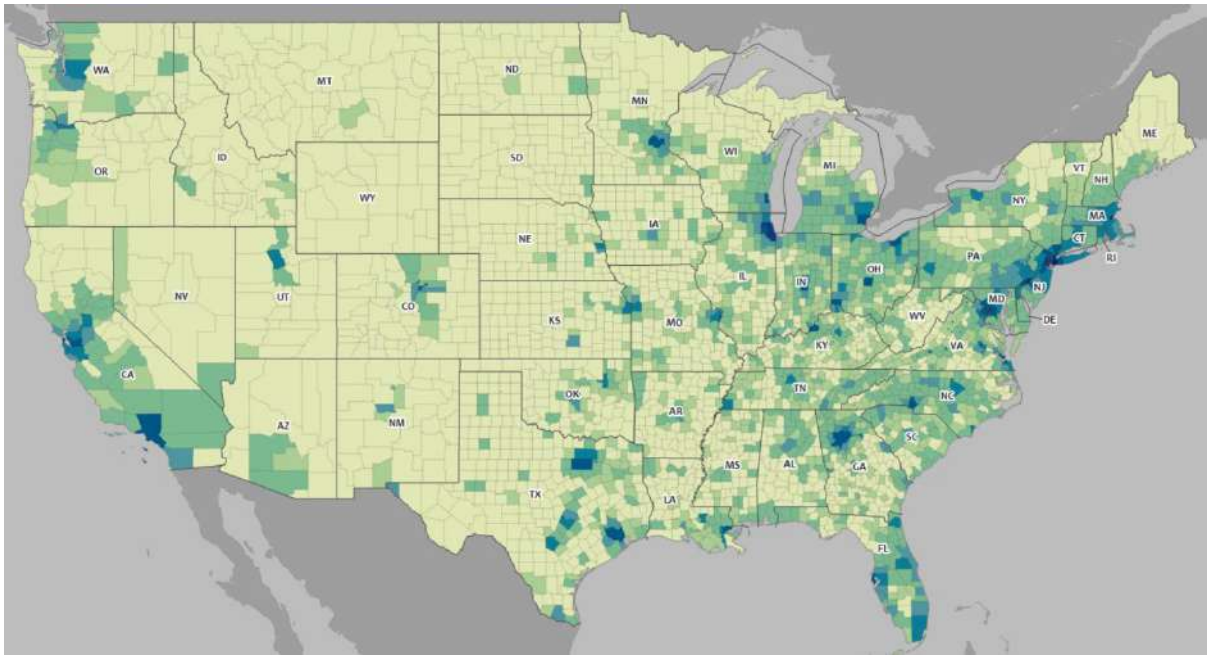

Linear Regression
test data
test prediction
Future prediction
1000
900
800
700
600
20
27
04
11
Oct
2021

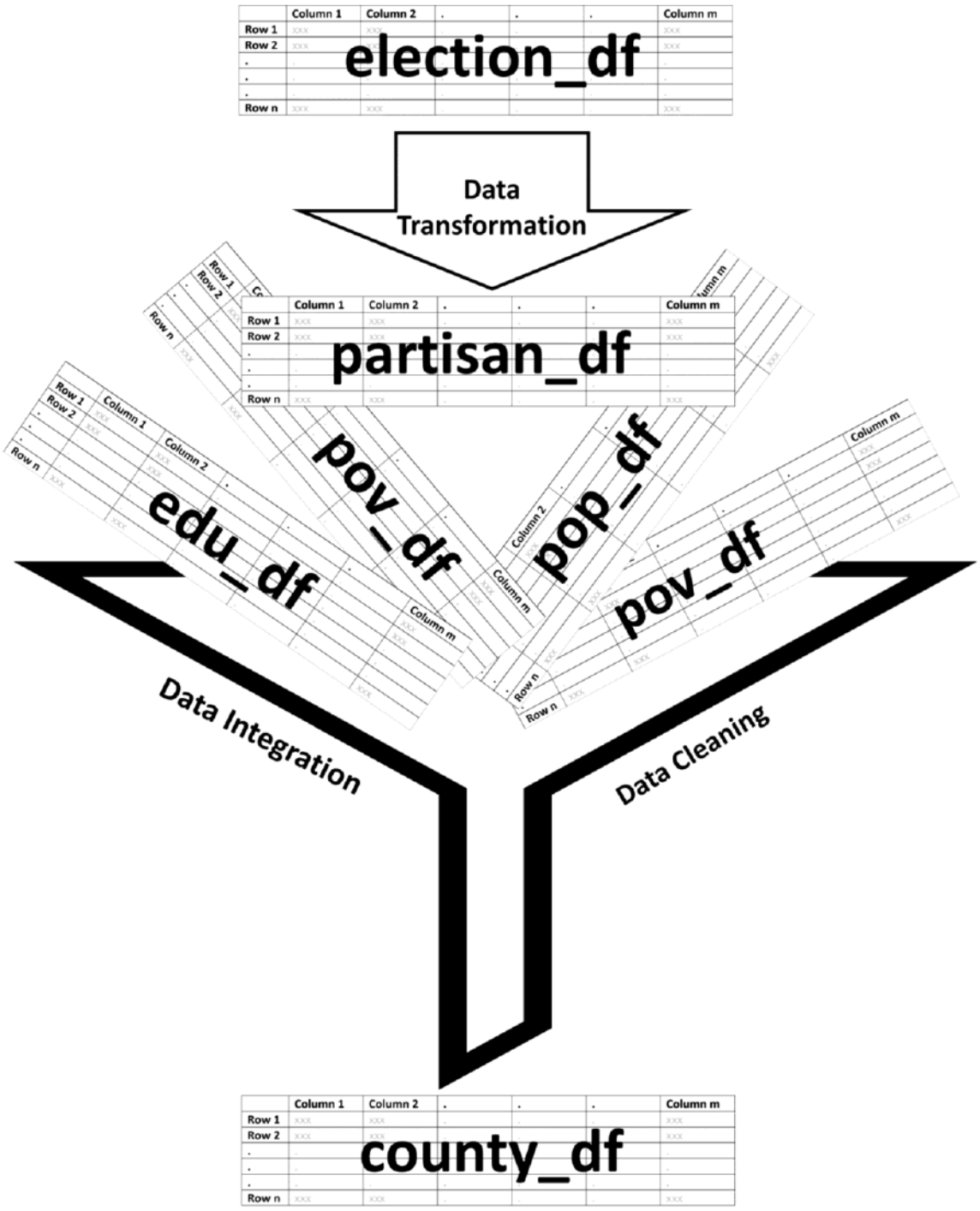

Decision Tree
test data
test prediction
Future prediction
1300
1200
1100
1000
900
800
700
600
500
20
27
04
11
Oct
2021


Multi-Layered Perceptron (MLP)
test data
test prediction
Future prediction
1400
1200
1000
800
600
20
27
04
11
Oct
2021

Chapter 17: Case Study 3: United States Counties Clustering Analysis






election_df
Data Transformation
partisan_df
edu_df
pov_df
pop_df
pov_df
Data Integration
Data Cleaning
county_df

			partisanism
state_po	**county_name**	**year**	
AK	**DISTRICT 1**	**2000**	0.510367
		2004	0.253528
		2008	0.222669
		2012	0.56734
		2016	0.0914432
...	**...**	**...**	...
WY	**WESTON**	**2004**	0.636498
		2008	0.574107
		2012	0.714201
		2016	0.775383
		2020	0.771629

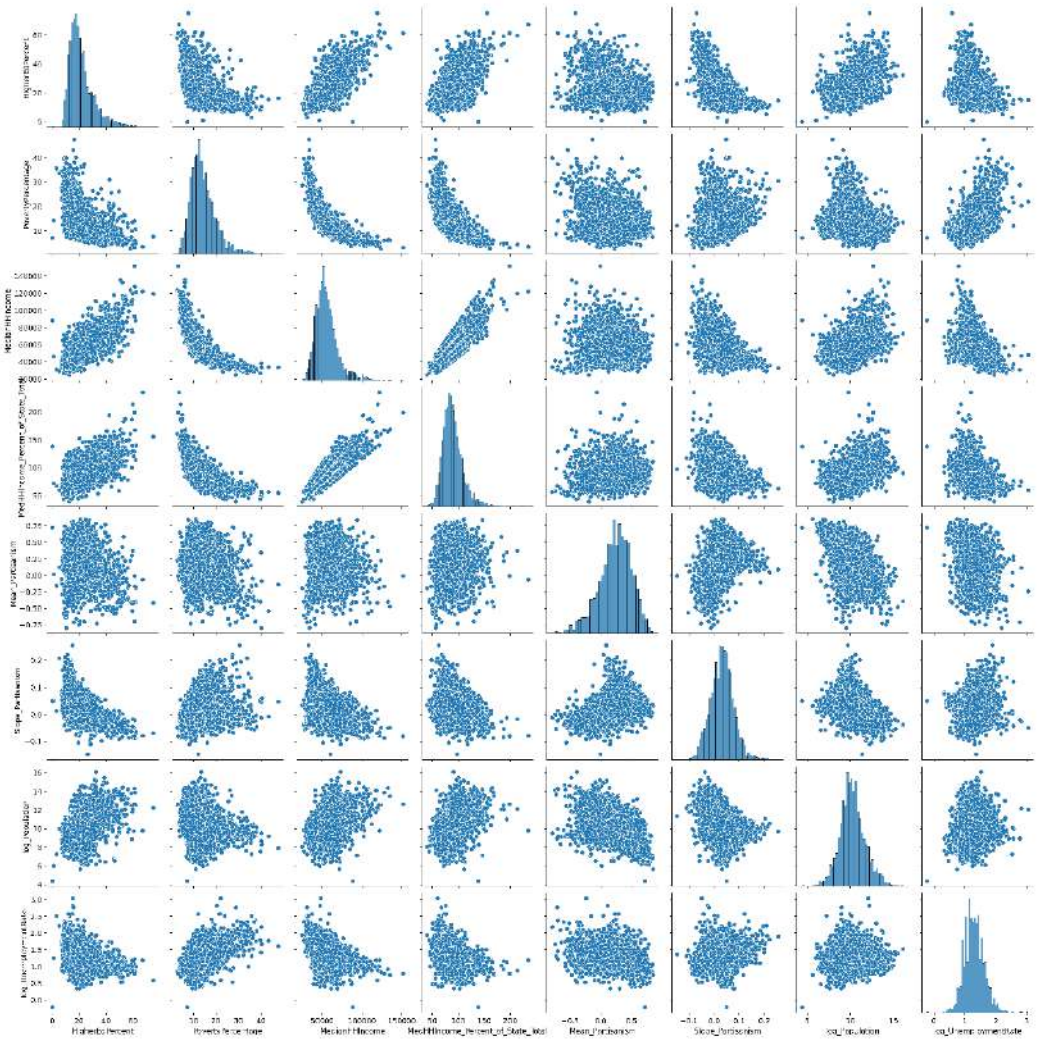
18050 rows × 1 columns

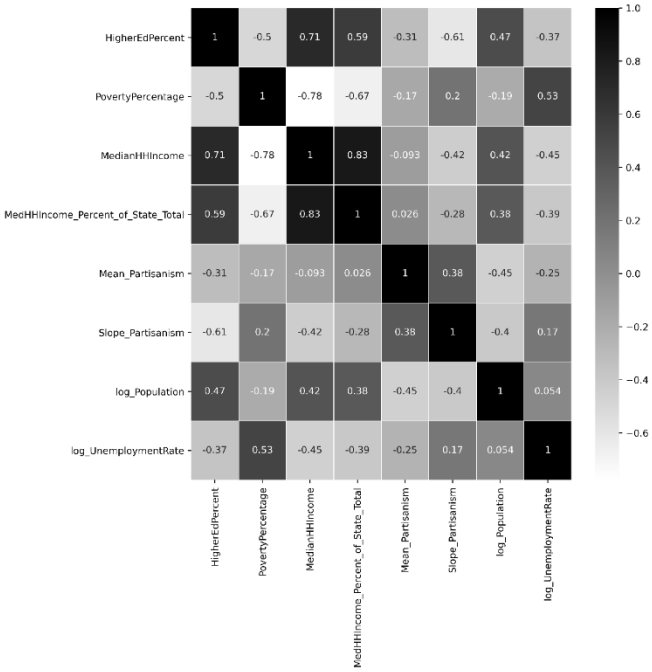
State	County_Name	Mean_Partisanism	Slope_Partisanism
AK	district 1	0.274999	-0.0762906
	district 10	0.404606	0.0388433
	district 11	0.429907	0.00906478
	district 12	0.417909	0.0196095
	district 13	0.28947	0.00391493
...	...	...	...
WY	sweetwater	0.379123	0.0557874
	teton	-0.155271	-0.086146
	uinta	0.540841	0.0254894
	washakie	0.573747	0.0170218
	weston	0.69149	0.0294079

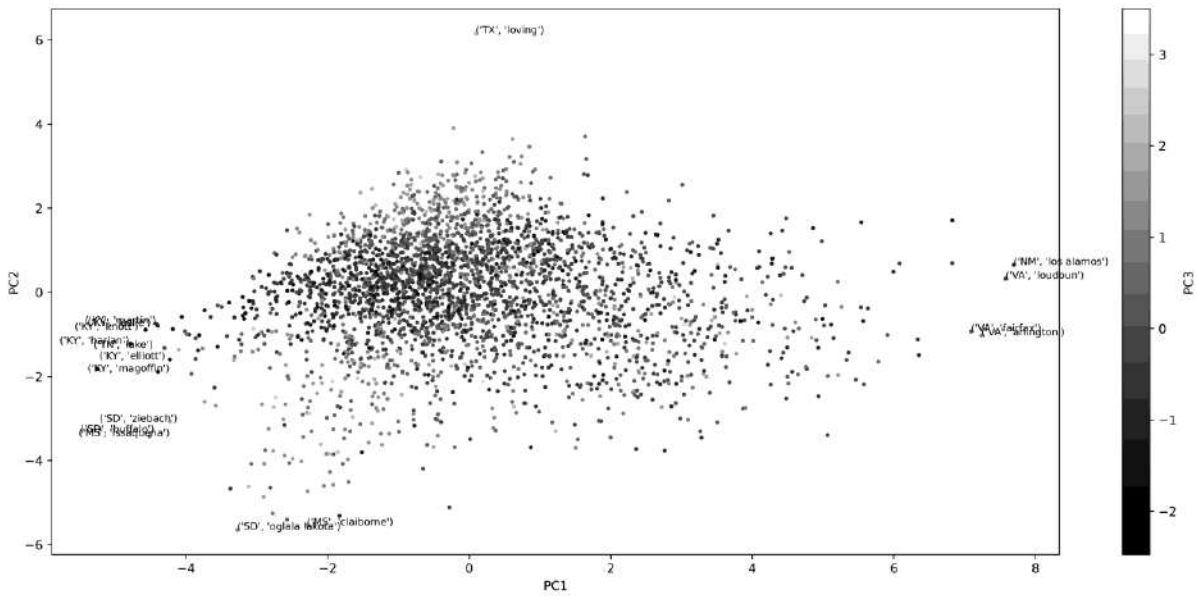
3151 rows × 2 columns

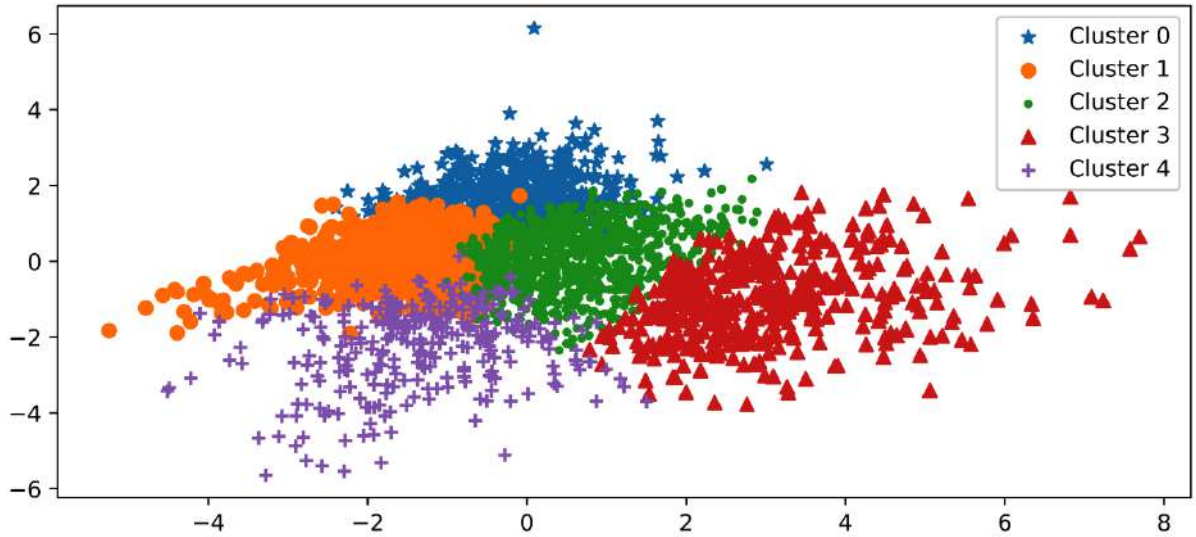
State	County_Name	Population	HigherEdPercent	PovertyPercentage	MedianHHIncome	UnemploymentRate	MedHHIncome_Percent_of_State_Total	Mean_Partisanism	Slope_Partisanism
	autauga	54571	26.5716	12.1	58233	2.7	112.482	0.467068	0.00184533
	baldwin	182265	31.8625	10.1	59871	2.8	115.646	0.532724	0.0128458
AL	barbour	27457	11.5787	27.1	35972	3.8	69.4829	0.0342589	0.00718466
	bibb	22915	10.3785	20.3	47918	3.1	92.5576	0.453212	0.0603812
	blount	57322	13.0934	16.3	52902	2.7	102.185	0.683058	0.0702235
...	...	...	...	...	...	...	...	...	...
	sweetwater	43806	22.4984	8.3	80639	4	121.9	0.379123	0.0557874
	teton	21294	57.0051	6	98837	2.8	149.409	-0.155271	-0.086146
WY	uinta	21118	16.029	8.5	70756	4	106.96	0.540841	0.0254894
	washakie	8533	23.3862	11.1	55122	4.1	83.3263	0.573747	0.0170218
	weston	7208	19.9725	10.5	59410	3	89.8083	0.69149	0.0294079

3007 rows × 8 columns

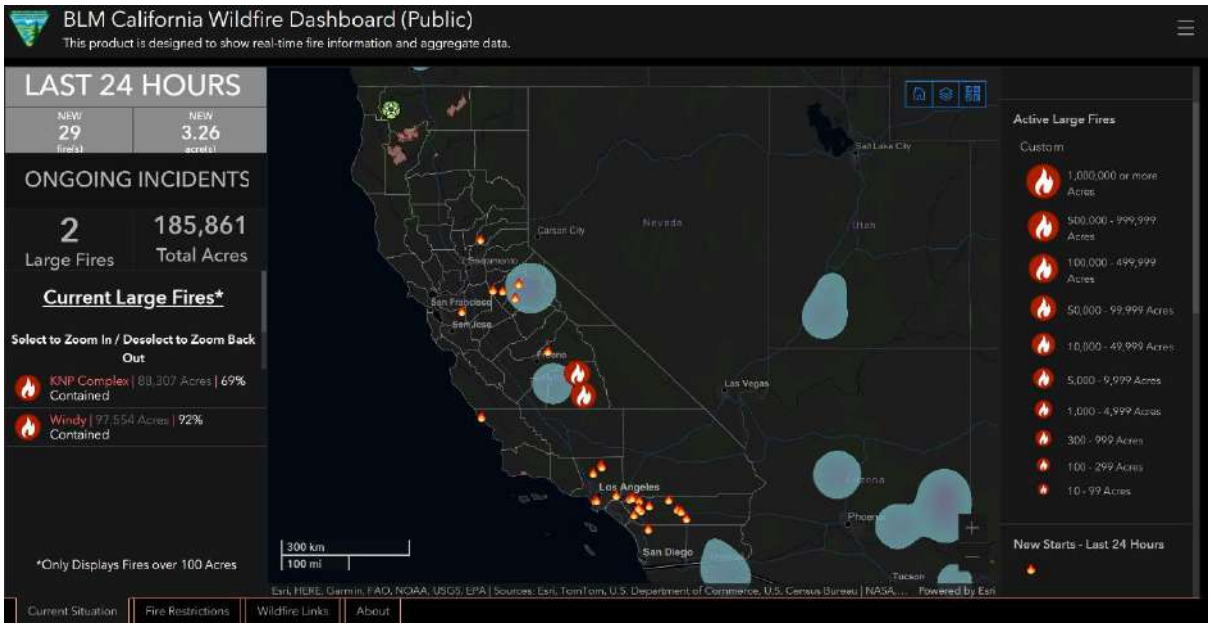


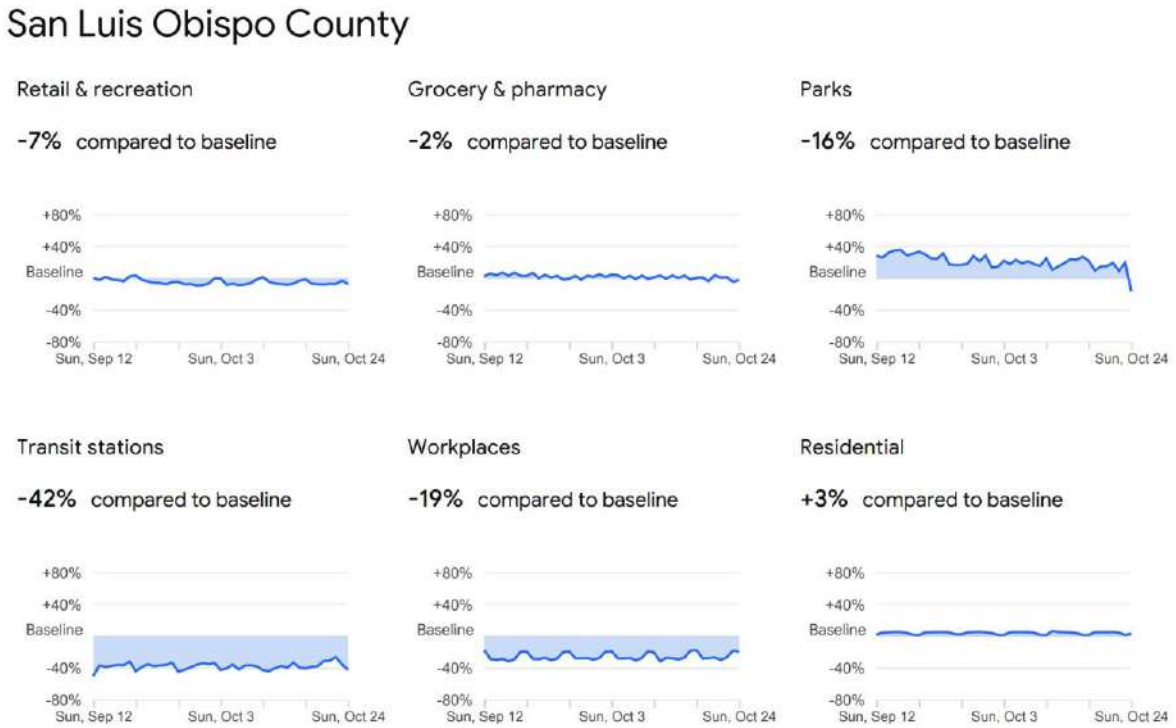

HigherEdPercent
PovertyPercentage
MedianHHIncome
MedHHIncome_Percent_of_State_Total
Mean_Partisanism
Slope_Partisanism
log_Population
log_UnemploymentRate
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6


('TX', 'loving')
('NM', 'los alamos')
('VA', 'loudoun')
('VA', 'arlington')
('KY', 'elliott')
('KY', 'magoffin')
('SD', 'ziebach')
('SD', 'oglala lakota')
('MS', 'claiborne')
PC1
PC2
PC3

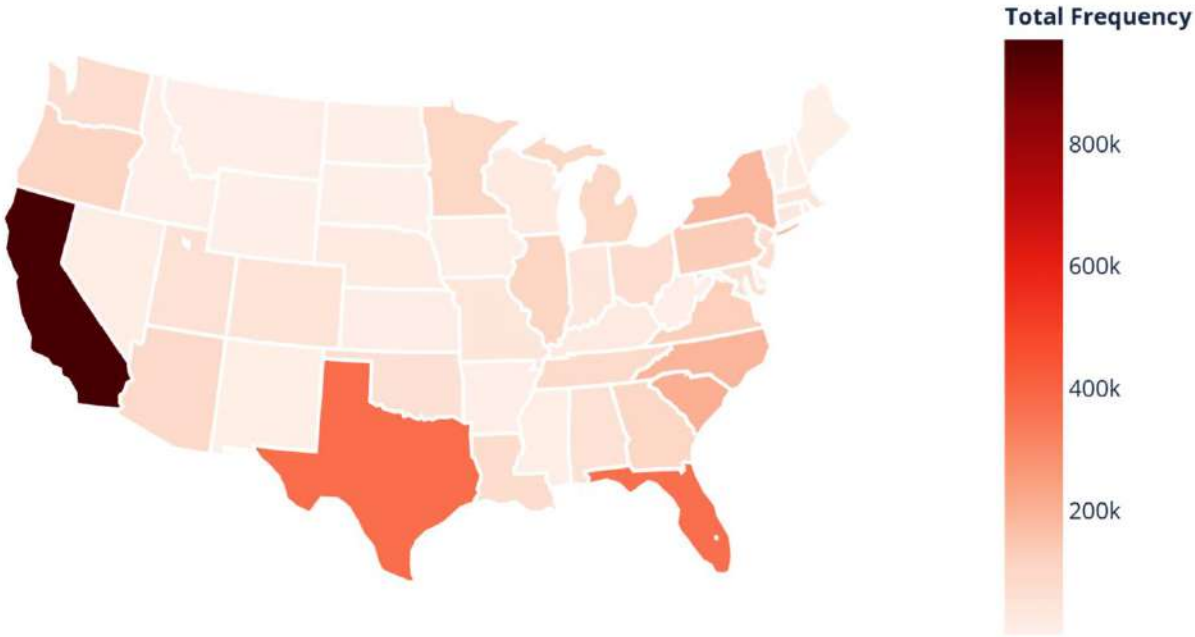

Cluster 0
Cluster 1
Cluster 2
Cluster 3
Cluster 4

Chapter 18: Summary, Practice Case Studies, and Conclusions





Frequency Distribution of US-Accidents (Dec 2020)
(Hover for breakdown)



SAN FRANCISCO POLICE DEPARTMENT

Crime Data

Comparing a Date Range within One Year to Its Prior Year

Please select:

District

- [x] (All)
- [x] Bayview
- [x] Central
- [x] Ingleside
- [x] Mission
- [x] Northern
- [x] Park
- [x] Richmond
- [x] Southern
- [x] Taraval
- [x] Tenderloin
- [x] Courtesy Reports

Type Of Crime

- [x] (All)
- [x] PART 1 PROPERTY CRIMES
- [x] PART 1 VIOLENT CRIMES

Start Date (1/1/2017 onward)

1/1/2020

End Date

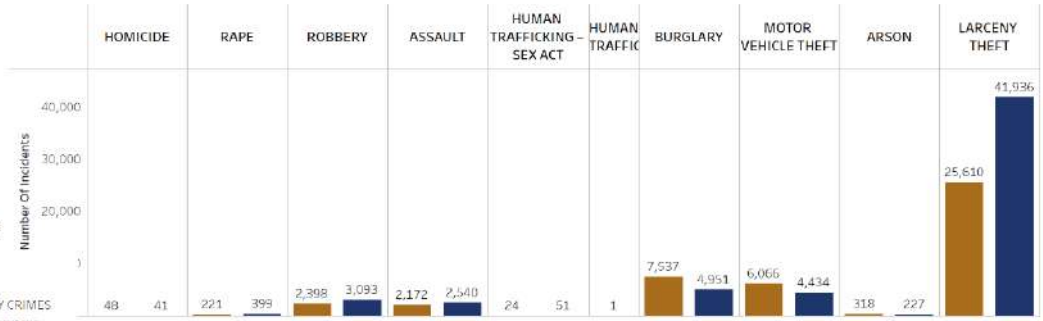
12/30/2020

Selected Date Range

Selected Date Range, Prior Year

Crime data updated weekly and valid through October 24, 2021

All
1/1/2020 - 12/30/2020



Crime	Selected Date Range	Selected Date Range, Prior Year	Year-to-Year % Decrease or Increase
HOMICIDE	48	41	17.1%
RAPE	221	399	-44.6%
ROBBERY	2,398	3,093	-22.5%
ASSAULT	2,172	2,540	-14.5%
HUMAN TRAFFICKING – SEX ACT	24	51	-52.9%
HUMAN TRAFFICKING – INV SERV		1	-100.0%
BURGLARY	7,537	4,951	52.2%
MOTOR VEHICLE THEFT	6,066	4,434	36.8%
ARSON	318	227	40.1%
LARCENY THEFT	25,610	41,936	-38.9%
TOTAL	**44,394**	**57,673**	**-23.0%**

Bud
FIFA WORLD CUP
RUSSIA
2018
MAN
OF THE
MATCH
7. Antoine GRIEZMANN