Isha Jain

Section 7.9

Conceptual

Page No.

Date

$(x-\xi)^3_+ = (x-\xi)^3$ for $x > \xi$

$(x-\xi)^3_+ = 0$ for $x \leq 0$

7.9.1. $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 (x-\xi)^3_+$

(a) $f_1(x) = a_1 + b_1 x + c_1 x^2 + d_1 x^3$

To find $f_1(x)$ for $x \leq \xi$ such that $f(x) = f_1(x)$

For $x \leq \xi$: $f(x) = \beta_0 + \beta_1 x + \beta_2 x^2 + \beta_3 x^3 + \beta_4 \cancel{(x-\xi)^3_+} \rightarrow 0$

$\therefore$ For $f(x) = f_1 x$:

$f(x) - f_1(x) = 0$

$\Rightarrow (\beta_0 - a_1) + (\beta_1 - b_1)x + (\beta_2 - c_1)x^2 + (\beta_3 - d_1)x^3 = 0$

$\Rightarrow a_1 = \beta_0$, $b_1 = \beta_1$, $c_1 = \beta_2$ and $d_1 = \beta_3$.

(b) $f_2(x) = a_2 + b_2 x + c_2 x^2 + d_2 x^3$

To find $f(x) = f_2(x)$ for all $x > \xi$ (a_2, b_2, c_2, d_2 in terms of $\beta_0, \beta_1, \beta_2, \beta_3$ and β_4)

$(x-\xi)^3 = x^3 - 3x^2\xi + 3x\xi^2 - \xi^3$

$\therefore$ for $f(x) = f_2(x)$ [Equating coefficients corresponding to the same degree of x]

$a_2 = \beta_0 - \beta_4 \xi^3$

$b_2 = \beta_1 + 3\beta_4 \xi^2$

$c_2 = \beta_2 - 3\beta_4 \xi$

$d_2 = \beta_4 + \beta_3$

(c) To prove: $f_1(\xi) = f_2(\xi)$ i.e. $f(x)$ is continuous at ξ

@ $x = \xi$:

$f_1(x) = f_1(\xi) = \beta_0 + \beta_1 \xi + \beta_2 \xi^2 + \beta_3 \xi^3$

[Using results from (a) about coefficients a_1, b_1, c_1 and d_1]

$f_2(x) = f_2(\xi) = a_2 + b_2 \xi + c_2 \xi^2 + d_2 \xi^3$

but $a_2 = \beta_0 - \beta_4 \xi^3$, $b_2 = \beta_1 + 3\beta_4 \xi^2$, $c_2 = \beta_2 - 3\beta_4 \xi$, $d_2 = \beta_4 + \beta_3$

$\cancel{\therefore f_2(\xi) = a_2 + b_2\xi + c_2\xi^2 + d_2\xi^3 = (\beta_0 - \beta_4\xi^3)}$

$\therefore f_2(\xi) = (\beta_0 - \beta_4 \xi^3) + (\beta_1 + 3\beta_4 \xi^2)\xi + (\beta_2 - 3\beta_4 \xi)\xi^2 + (\beta_3 + \beta_4)\xi^3$

$\therefore f_2(\xi) = \beta_0 + \beta_1 \xi^2 + \beta_2 \xi^2 + \beta_3 \xi^3$ $\therefore f(x)$ is continuous at ξ

Isha Jain

Page No.
Date

(d) To prove : $f_1'(\xi) = f_2'(\xi)$ i.e. $f'(x)$ is continuous at ξ

$f_1'(x) = b_1 + 2c_1 + 3d_1 x^2$ On substituting the coefficients:

$f_1'(x) = \beta_1 + 2\beta_2 x + 3\beta_3 x^2$. On substituting ξ:

$f_1'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$

$f_2'(x) = b_2 + 2c_2 x + 3d_2 x^2$ On substituting the coefficients:

$f_2'(x) = (\beta_1 + 3\beta_4 \xi^2) + 2(\beta_2 - 3\beta_4 \xi)x + 3(\beta_3 + \beta_4) x^2$

On substituting ξ:

$f_2'(\xi) = (\beta_1 + 3\beta_4 \xi^2) + 2(\beta_2 - 3\beta_4 \xi)\xi + 3(\beta_3 + \beta_4)\xi^2$

$\therefore f_2'(\xi) = \beta_1 + 2\beta_2 \xi + 3\beta_3 \xi^2$

$\therefore f'(x)$ is continuous at ξ

(e) $f_1''(x) = 2c_1 + 6d_1 x$. On substituting coefficients:

$f_1''(x) = 2\beta_2 + 6\beta_3 x$. On replacing x with ξ

$f_1''(\xi) = 2\beta_2 + 6\beta_3 \xi$

$f_2''(x) = 2c_2 x + 6d_2 x$. On substituting coefficients:

$f_2''(x) = 2(\beta_2 - 3\beta_4 \xi) + 6(\beta_3 + \beta_4)x$. On replacing x with ξ:

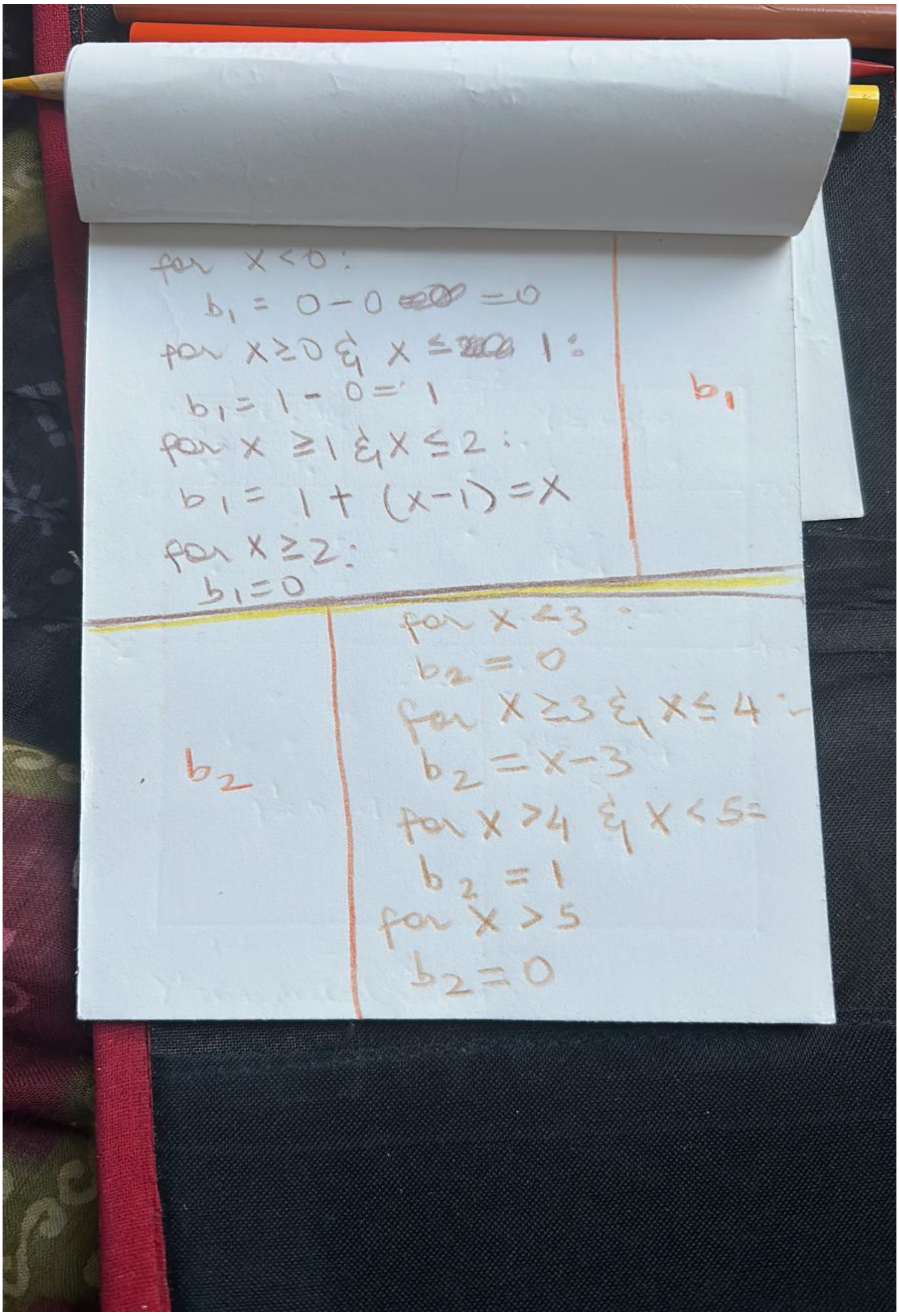
$f_2''(\xi) = 2(\beta_2 - 3\beta_4 \xi) + 6(\beta_3 + \beta_4)\xi = 2\beta_2 + 6\beta_3 \xi$.

$\therefore f''(x)$ is continuous at ξ

$\therefore f(x)$ is a cubic spline.

Isha Jain

4.



for $x < 0$:

$b_1 = 0 - 0 = 0$

for $x \geq 0$ & $x \leq 1$:

$b_1 = 1 - 0 = 1$

for $x \geq 1$ & $x \leq 2$:

$b_1 = 1 + (x-1) = x$

for $x \geq 2$:

$b_1 = 0$

b_1

b_2

for $x < 3$:

$b_2 = 0$

for $x \geq 3$ & $x \leq 4$:

$b_2 = x - 3$

for $x > 4$ & $x < 5$:

$b_2 = 1$

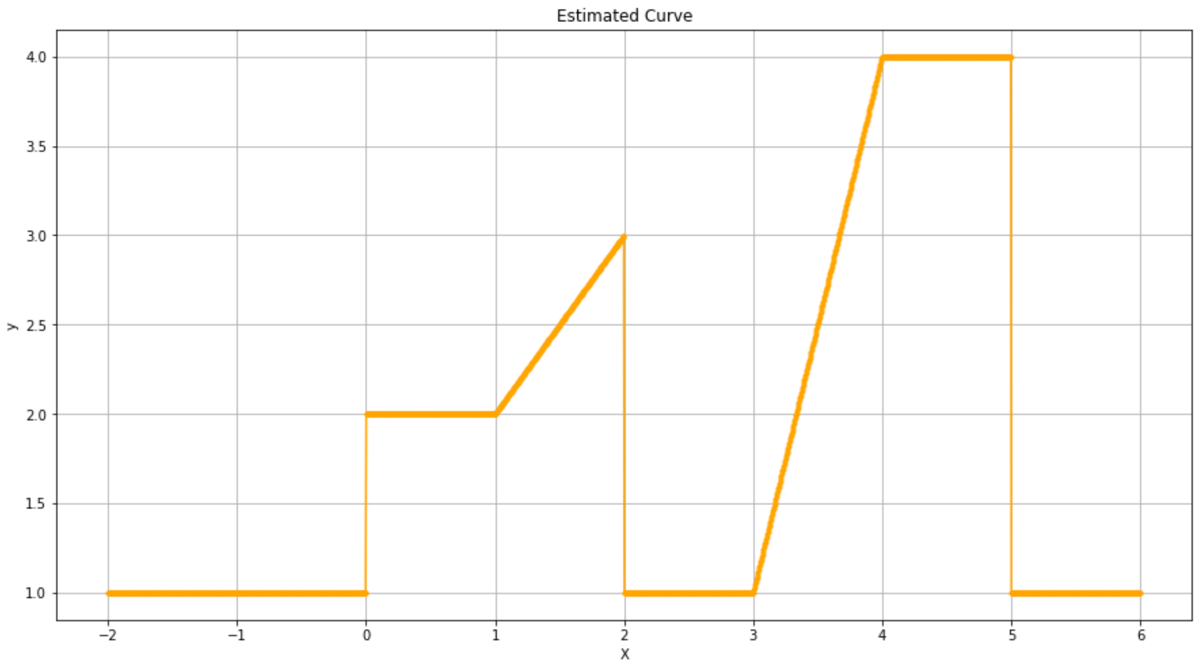
for $x > 5$

$b_2 = 0$

```
7.9
```

4

In [5]:
```
import numpy as np
import matplotlib.pyplot as plt

def curve(X):
    β0=1
    β1=1
    β2=3
    if X<0:
        b1=0
    if X<=1 and X>=0:
        b1=1
    if X>1 and X<=2:
        b1=X
    if X>2:
        b1=0
    if X<=3:
        b2=0
    if X<=4 and X>=3:
        b2=X-3
    if X<=5 and X>4:
        b2=1
    if X>5:
        b2=0
    return β0+β1*b1+β2*b2


X = np.linspace(-2,6,1700)

y=[]
for i in X:
    element=curve(i)
    y.append(element)

fig = plt.figure(figsize=(15, 8))
ax = fig.add_subplot(111)
plt.plot(X, y, marker=".", color='orange')
ax.set_xlabel('X')
ax.set_ylabel('y')
ax.set_title('Estimated Curve')
plt.grid()
plt.show()
```



Isha Jain

Applied

7

```
In [2]: pip install ISLP;
```

```
Requirement already satisfied: ISLP in c:\users\ishaj\anaconda3\lib\site-packages (0.3.19)
Requirement already satisfied: scikit-learn>=1.2 in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.3.0)
Requirement already satisfied: numpy<1.25,>=1.7.1 in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.20.3)
Requirement already satisfied: lxml in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (4.6.3)
Requirement already satisfied: pygam in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (0.8.0)
Requirement already satisfied: pytorch-lightning in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (2.0.9)
Requirement already satisfied: joblib in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.3.2)
Requirement already satisfied: lifelines in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (0.27.8)
Note: you may need to restart the kernel to use updated packages.
Requirement already satisfied: torchmetrics in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.1.2)
Requirement already satisfied: torch in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (2.0.1)
Requirement already satisfied: statsmodels>=0.13 in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (0.14.0)
Requirement already satisfied: scipy>=0.9 in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.7.1)
Requirement already satisfied: pandas<=1.9,>=0.20 in c:\users\ishaj\anaconda3\lib\site-packages (from ISLP) (1.3.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\ishaj\anaconda3\lib\site-packages (from pandas<=1.9,>=0.20->I
SLP) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\ishaj\anaconda3\lib\site-packages (from pandas<=1.9,>=0.20->ISLP) (202
1.3)
Requirement already satisfied: six>=1.5 in c:\users\ishaj\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas<=1.
9,>=0.20->ISLP) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\ishaj\anaconda3\lib\site-packages (from scikit-learn>=1.2->ISL
P) (2.2.0)
Requirement already satisfied: packaging>=21.3 in c:\users\ishaj\anaconda3\lib\site-packages (from statsmodels>=0.13->ISLP) (2
3.1)
Requirement already satisfied: patsy>=0.5.2 in c:\users\ishaj\anaconda3\lib\site-packages (from statsmodels>=0.13->ISLP) (0.5.
2)
Requirement already satisfied: matplotlib>=3.0 in c:\users\ishaj\anaconda3\lib\site-packages (from lifelines->ISLP) (3.4.3)
Requirement already satisfied: formulaic>=0.2.2 in c:\users\ishaj\anaconda3\lib\site-packages (from lifelines->ISLP) (0.6.4)
Requirement already satisfied: autograd>=1.5 in c:\users\ishaj\anaconda3\lib\site-packages (from lifelines->ISLP) (1.6.2)
Requirement already satisfied: autograd-gamma>=0.3 in c:\users\ishaj\anaconda3\lib\site-packages (from lifelines->ISLP) (0.5.0)
Requirement already satisfied: future>=0.15.2 in c:\users\ishaj\anaconda3\lib\site-packages (from autograd>=1.5->lifelines->ISL
P) (0.18.2)
Requirement already satisfied: interface-meta>=1.2.0 in c:\users\ishaj\anaconda3\lib\site-packages (from formulaic>=0.2.2->life
lines->ISLP) (1.3.0)
Requirement already satisfied: astor>=0.8 in c:\users\ishaj\anaconda3\lib\site-packages (from formulaic>=0.2.2->lifelines->ISL
P) (0.8.1)
Requirement already satisfied: typing-extensions>=4.2.0 in c:\users\ishaj\anaconda3\lib\site-packages (from formulaic>=0.2.2->l
ifelines->ISLP) (4.7.1)
Requirement already satisfied: wrapt>=1.0 in c:\users\ishaj\anaconda3\lib\site-packages (from formulaic>=0.2.2->lifelines->ISL
P) (1.12.1)
Requirement already satisfied: cycler>=0.10 in c:\users\ishaj\anaconda3\lib\site-packages (from matplotlib>=3.0->lifelines->ISL
P) (0.10.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\ishaj\anaconda3\lib\site-packages (from matplotlib>=3.0->lifelines->IS
LP) (8.4.0)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\ishaj\anaconda3\lib\site-packages (from matplotlib>=3.0->lifelines-
>ISLP) (3.0.4)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\ishaj\anaconda3\lib\site-packages (from matplotlib>=3.0->lifelines
->ISLP) (1.3.1)
Requirement already satisfied: progressbar2 in c:\users\ishaj\anaconda3\lib\site-packages (from pygam->ISLP) (4.2.0)
Requirement already satisfied: python-utils>=3.0.0 in c:\users\ishaj\anaconda3\lib\site-packages (from progressbar2->pygam->ISL
P) (3.7.0)
Requirement already satisfied: fsspec[http]>2021.06.0 in c:\users\ishaj\anaconda3\lib\site-packages (from pytorch-lightning->IS
LP) (2021.10.1)
Requirement already satisfied: lightning-utilities>=0.7.0 in c:\users\ishaj\anaconda3\lib\site-packages (from pytorch-lightning
->ISLP) (0.9.0)
Requirement already satisfied: PyYAML>=5.4 in c:\users\ishaj\anaconda3\lib\site-packages (from pytorch-lightning->ISLP) (6.0)
Requirement already satisfied: tqdm>=4.57.0 in c:\users\ishaj\anaconda3\lib\site-packages (from pytorch-lightning->ISLP) (4.62.
3)
Requirement already satisfied: aiohttp in c:\users\ishaj\anaconda3\lib\site-packages (from fsspec[http]>2021.06.0->pytorch-ligh
tning->ISLP) (3.8.5)
Requirement already satisfied: requests in c:\users\ishaj\anaconda3\lib\site-packages (from fsspec[http]>2021.06.0->pytorch-lig
htning->ISLP) (2.26.0)
Requirement already satisfied: filelock in c:\users\ishaj\anaconda3\lib\site-packages (from torch->ISLP) (3.3.1)
Requirement already satisfied: networkx in c:\users\ishaj\anaconda3\lib\site-packages (from torch->ISLP) (2.6.3)
Requirement already satisfied: jinja2 in c:\users\ishaj\anaconda3\lib\site-packages (from torch->ISLP) (2.11.3)
Requirement already satisfied: sympy in c:\users\ishaj\anaconda3\lib\site-packages (from torch->ISLP) (1.9)
Requirement already satisfied: colorama in c:\users\ishaj\anaconda3\lib\site-packages (from tqdm>=4.57.0->pytorch-lightning->IS
LP) (0.4.4)
Requirement already satisfied: multidict<7.0,>=4.5 in c:\users\ishaj\anaconda3\lib\site-packages (from aiohttp->fsspec[http]>20
21.06.0->pytorch-lightning->ISLP) (6.0.4)
Requirement already satisfied: yarl<2.0,>=1.0 in c:\users\ishaj\anaconda3\lib\site-packages (from aiohttp->fsspec[http]>2021.0
6.0->pytorch-lightning->ISLP) (1.9.2)
Requirement already satisfied: frozenlist>=1.1.1 in c:\users\ishaj\anaconda3\lib\site-packages (from aiohttp->fsspec[http]>202
1.06.0->pytorch-lightning->ISLP) (1.4.0)
Requirement already satisfied: async-timeout<5.0,>=4.0.0a3 in c:\users\ishaj\anaconda3\lib\site-packages (from aiohttp->fsspec
```

Isha Jain

```
In [3]: import numpy as np
        import pandas as pd
        from matplotlib.pyplot import subplots
        from statsmodels.datasets import get_rdataset
        import sklearn.model_selection as skm
        from ISLP import load_data, confusion_table
        from ISLP.models import ModelSpec as MS
        from ISLP.bart import BART
```

```
In [4]: Wage = load_data('Wage')
        Wage.head(5)
```

Out[4]:

	year	age	maritl	race	education	region	jobclass	health	health_ins	logwage	wage
0	2006	18	1. Never Married	1. White	1. < HS Grad	2. Middle Atlantic	1. Industrial	1. <=Good	2. No	4.318063	75.043154
1	2004	24	1. Never Married	1. White	4. College Grad	2. Middle Atlantic	2. Information	2. >=Very Good	2. No	4.255273	70.476020
2	2003	45	2. Married	1. White	3. Some College	2. Middle Atlantic	1. Industrial	1. <=Good	1. Yes	4.875061	130.982177
3	2003	43	2. Married	3. Asian	4. College Grad	2. Middle Atlantic	2. Information	2. >=Very Good	1. Yes	5.041393	154.685293
4	2005	50	4. Divorced	1. White	2. HS Grad	2. Middle Atlantic	2. Information	1. <=Good	1. Yes	4.318063	75.043154

```
In [28]: Wage.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3000 entries, 0 to 2999
Data columns (total 11 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   year        3000 non-null   int64
 1   age         3000 non-null   int64
 2   maritl      3000 non-null   object
 3   race        3000 non-null   object
 4   education   3000 non-null   category
 5   region      3000 non-null   object
 6   jobclass    3000 non-null   object
 7   health      3000 non-null   object
 8   health_ins  3000 non-null   object
 9   logwage     3000 non-null   float64
 10  wage        3000 non-null   float64
dtypes: category(1), float64(2), int64(2), object(6)
memory usage: 237.6+ KB
```

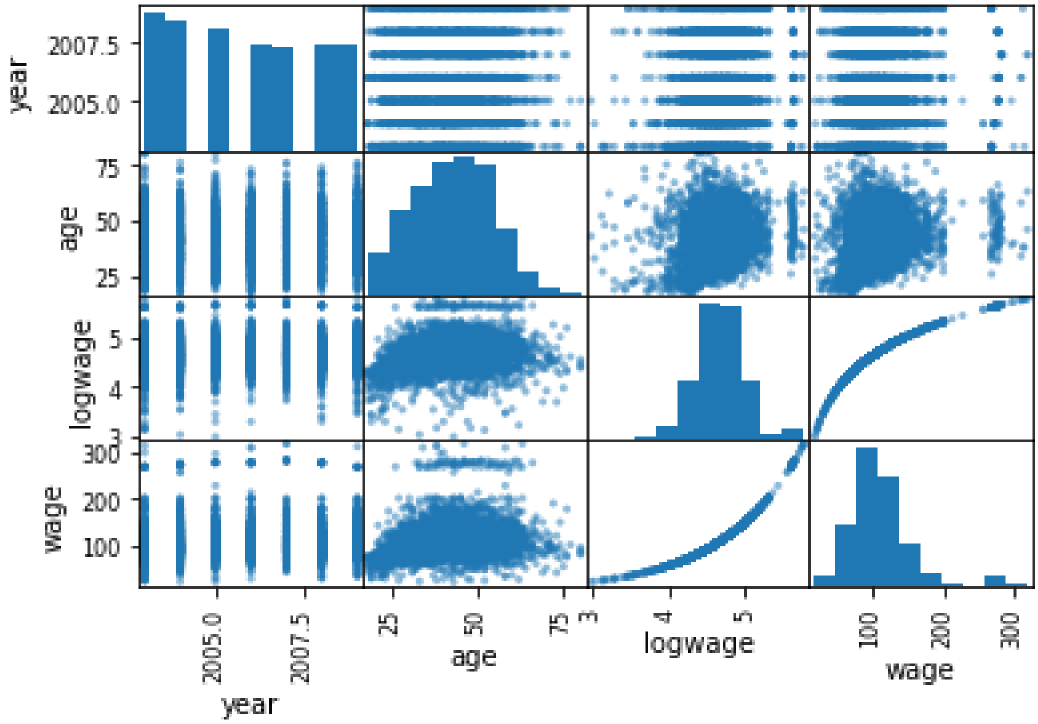
```
In [29]: Wage.describe()
```

Out[29]:

	year	age	logwage	wage
count	3000.000000	3000.000000	3000.000000	3000.000000
mean	2005.791000	42.414667	4.653905	111.703608
std	2.026167	11.542406	0.351753	41.728595
min	2003.000000	18.000000	3.000000	20.085537
25%	2004.000000	33.750000	4.447158	85.383940
50%	2006.000000	42.000000	4.653213	104.921507
75%	2008.000000	51.000000	4.857332	128.680488
max	2009.000000	80.000000	5.763128	318.342430

```
In [5]: pd.plotting.scatter_matrix(Wage)

Out[5]: array([[<AxesSubplot:xlabel='year', ylabel='year'>,
                <AxesSubplot:xlabel='age', ylabel='year'>,
                <AxesSubplot:xlabel='logwage', ylabel='year'>,
                <AxesSubplot:xlabel='wage', ylabel='year'>],
               [<AxesSubplot:xlabel='year', ylabel='age'>,
                <AxesSubplot:xlabel='age', ylabel='age'>,
                <AxesSubplot:xlabel='logwage', ylabel='age'>,
                <AxesSubplot:xlabel='wage', ylabel='age'>],
               [<AxesSubplot:xlabel='year', ylabel='logwage'>,
                <AxesSubplot:xlabel='age', ylabel='logwage'>,
                <AxesSubplot:xlabel='logwage', ylabel='logwage'>,
                <AxesSubplot:xlabel='wage', ylabel='logwage'>],
               [<AxesSubplot:xlabel='year', ylabel='wage'>,
                <AxesSubplot:xlabel='age', ylabel='wage'>,
```

Log wage isn't considered a predictor as it would lead to data snooping. Year and age can be considered quantitative predictors whereas the rest can be considered categorical.

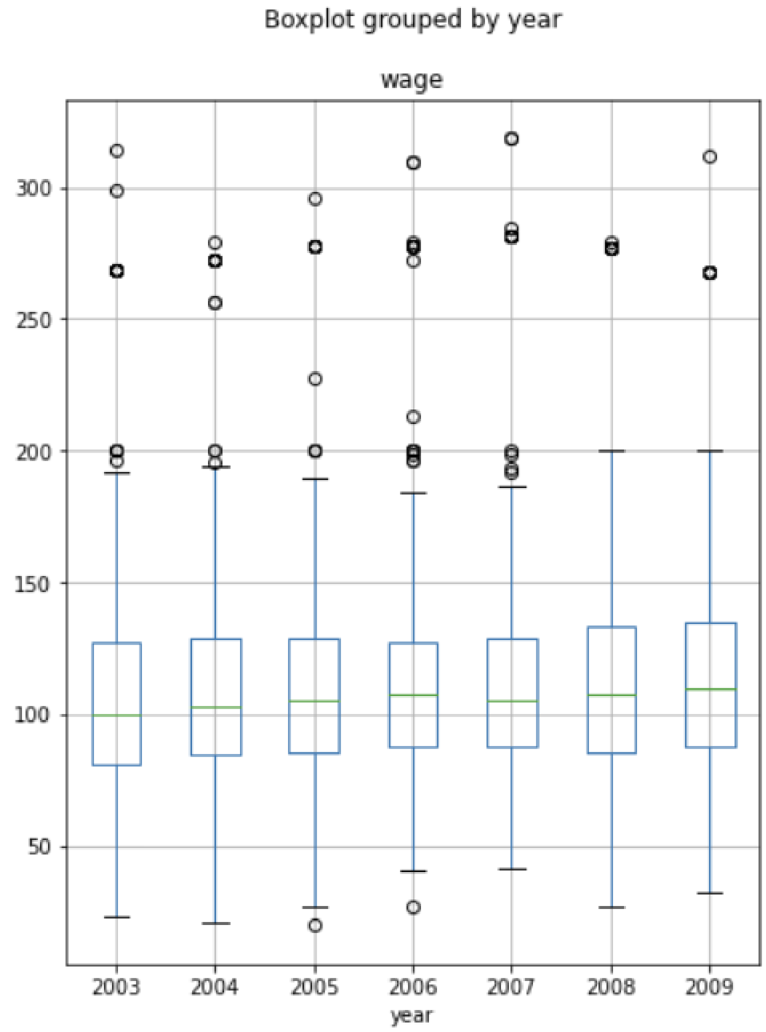


Above is a scatterplot involving age, year, logwage and wage. From the histogrms we can conclude that there’s relative similar number of datapoints for all years. For age, the histogram hints to a higher number of datapoints for ages in the 30 to 60 bracket. Extreme wages have a lower frequency as expected, with maximum observations correponding to the 100 wage range.

Use boxplots, to discover trends of wage across the predictor variables:

- Wage by year-

In [15]:
```
import matplotlib.pyplot as plt

fig, ax = plt.subplots(figsize=(6,8))
Wage.boxplot('wage', by='year', ax=ax);
```



The above boxplot suggests a slight increase in wage through the years.

- Wage by age-

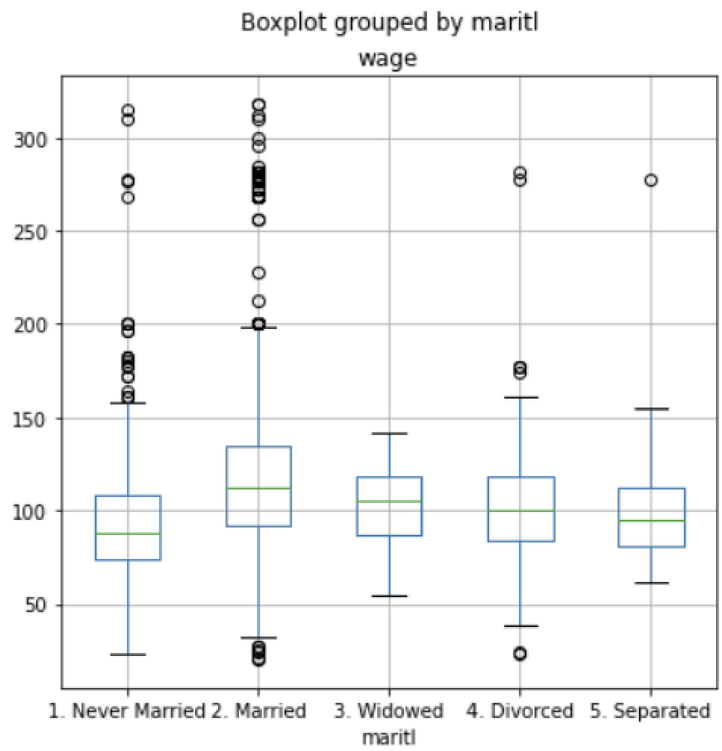
In [10]:
```
fig, ax = plt.subplots(figsize=(29, 20))
Wage.boxplot('wage', by='age', ax=ax);
```

From the above boxplot, we can conclude that ages 30 through 60 seem to have a higher wage than the other ages.

- Wage by marital status

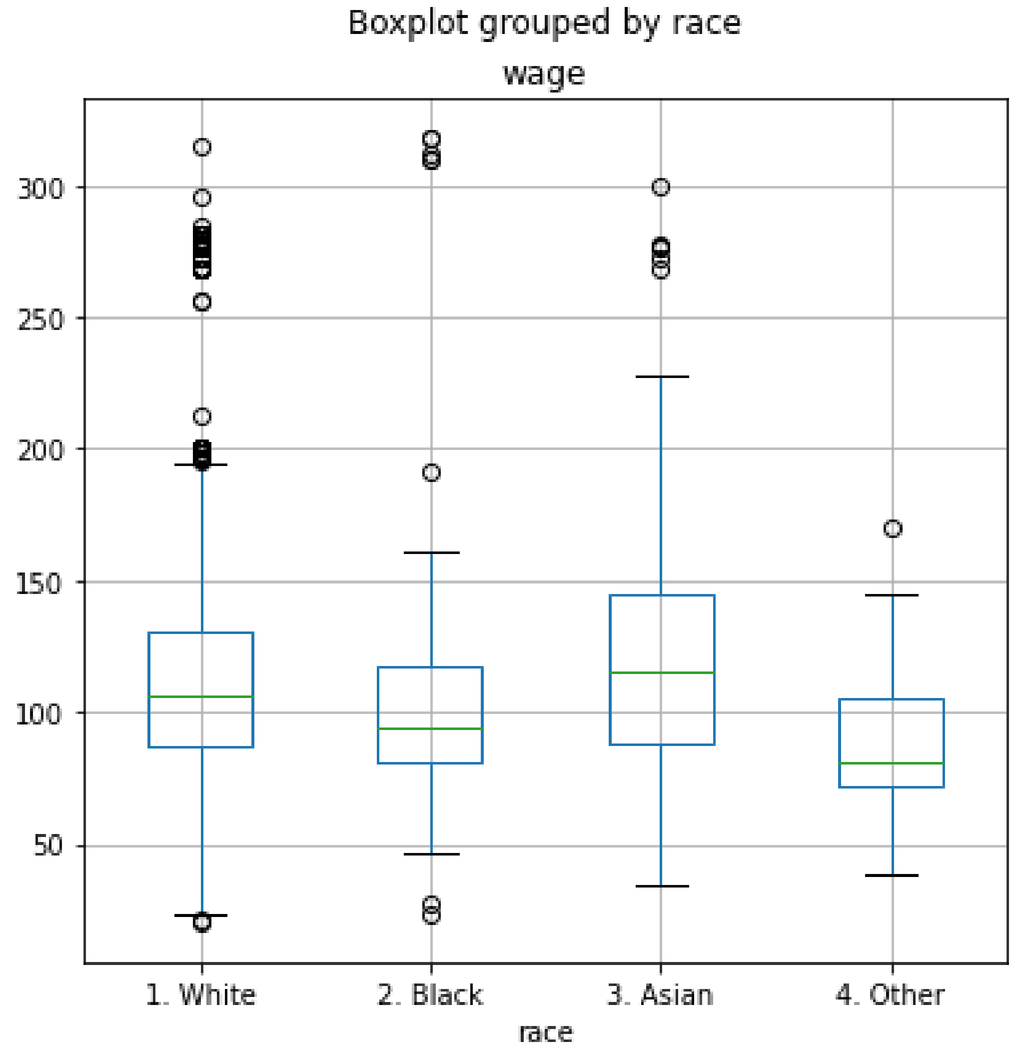
Married individuals seem to have the highest wage, followed by widowed people, followed by divorcees. Those that have never been married or have separated seem to have the lowest wage (based on mean wage for each category).

In [11]:
```
fig, ax = plt.subplots(figsize=(6, 6))
Wage.boxplot('wage', by='maritl', ax=ax);
```



In [12]:
```
fig, ax = plt.subplots(figsize=(6, 6))
Wage.boxplot('wage', by='race', ax=ax);
```

- Wage by race-



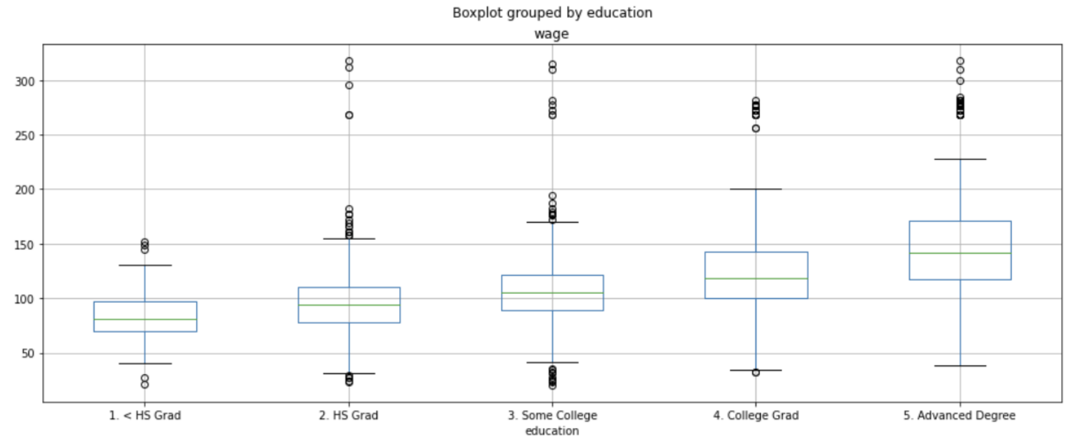
Asians seem to have the highest wage, followed by whites, then blacks and finally others.

- Wage by education

As expected, those with higher education seem to have higher wage, i.e , advance degree holders earning the highest wage, followed by collegae graduates, followed by thse with some college education, followed by highschool graduates, followed by those who haven't completed high school.

Isha Jain

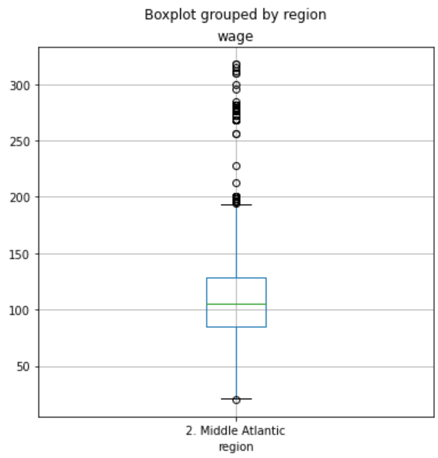
```
In [14]: fig, ax = plt.subplots(figsize=(16, 6))
         Wage.boxplot('wage', by='education', ax=ax);
```




```
In [16]: fig, ax = plt.subplots(figsize=(6, 6))
         Wage.boxplot('wage', by='region', ax=ax);
```

- Wage by region:

The dataset seems to contain only one region.

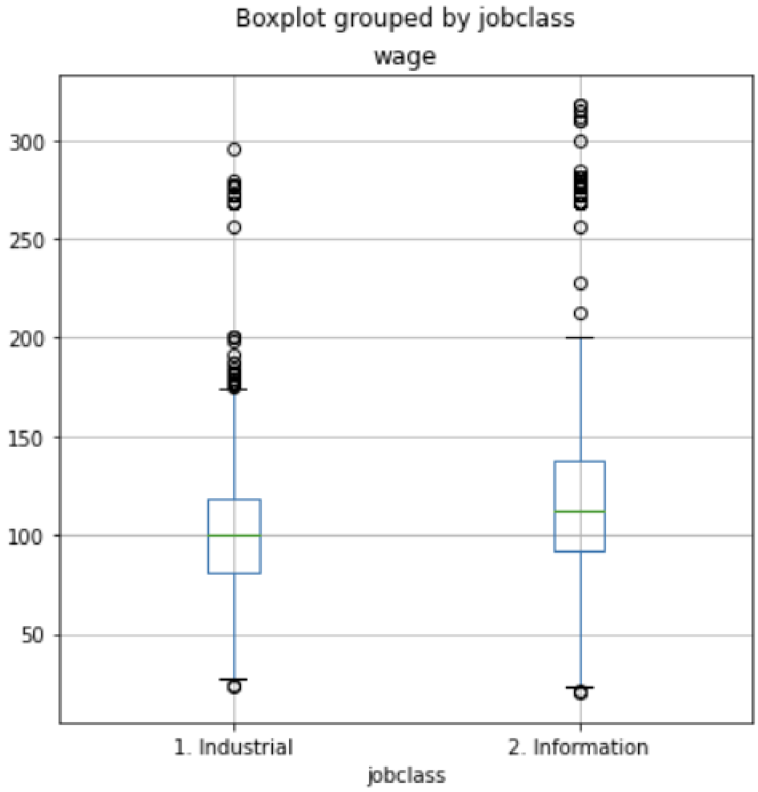


- Wage by jobclass:

Information jobclass seems to have a higher wage than industrial.

```
In [24]: Wage['region'].unique()
Out[24]: array(['2. Middle Atlantic'], dtype=object)
```

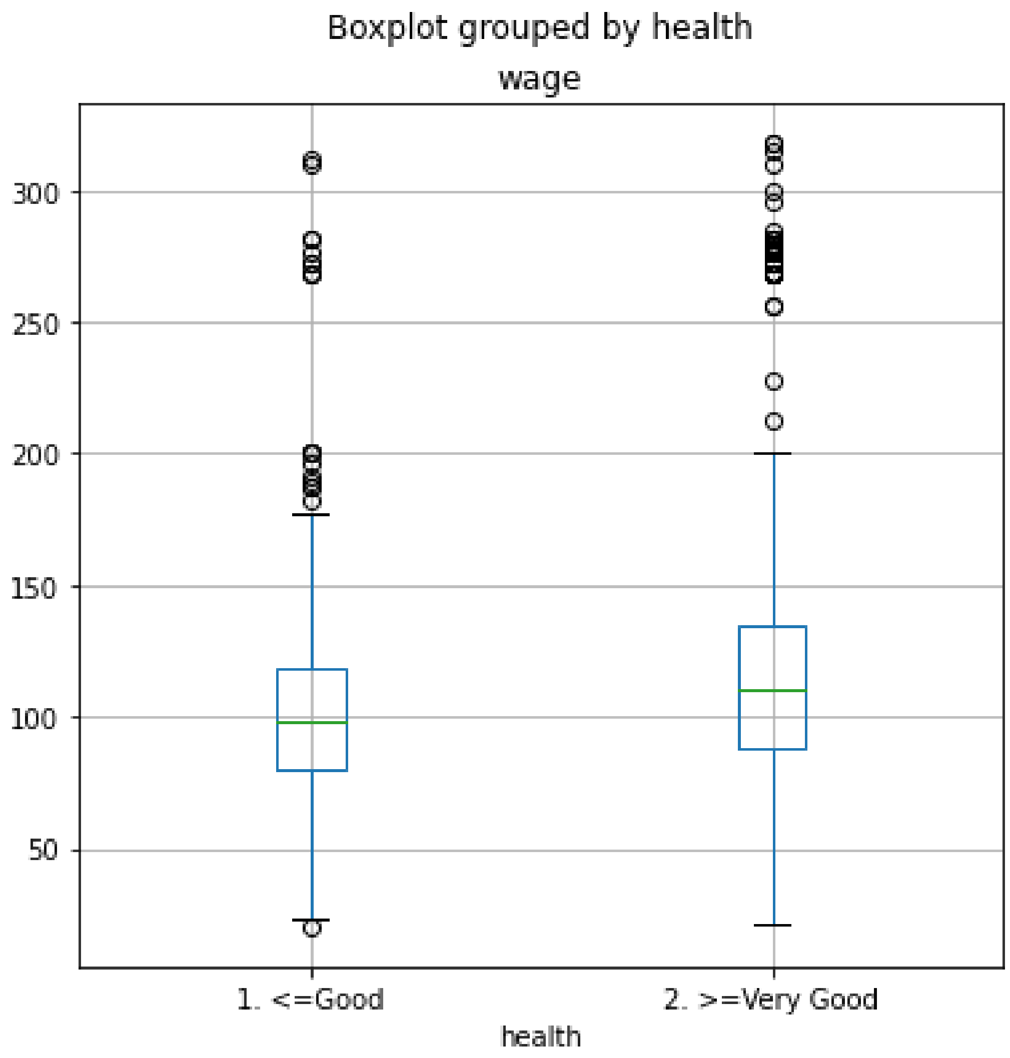
```
In [25]: fig, ax = plt.subplots(figsize=(6, 6))
         Wage.boxplot('wage', by='jobclass', ax=ax);
```




```
In [26]: fig, ax = plt.subplots(figsize=(6, 6))
         Wage.boxplot('wage', by='health', ax=ax);
```

- Wage by health:

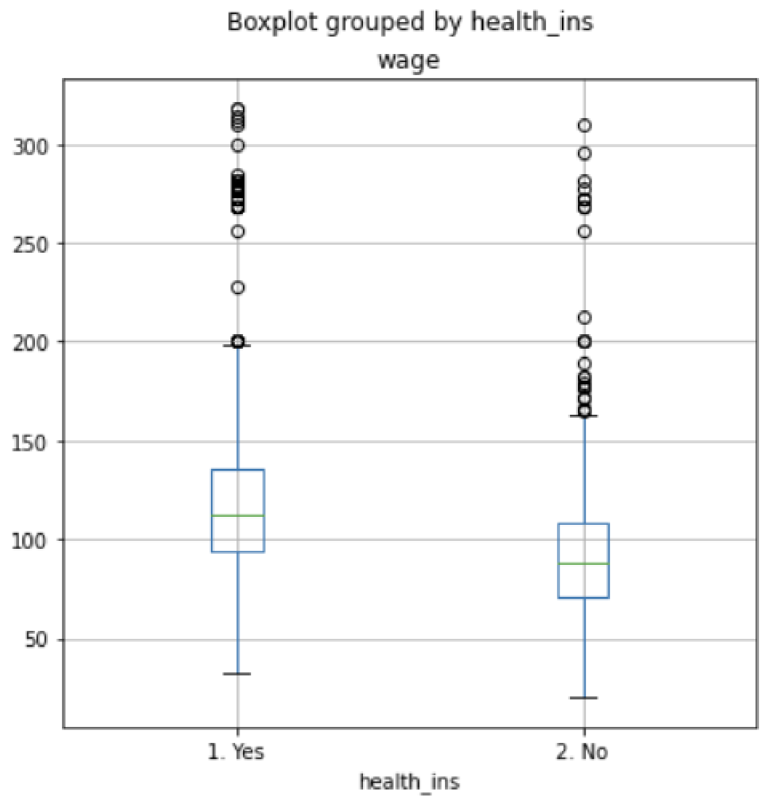
Those with very good health or better seem to have higher wages than those with good or less than good health.



- Wage by health insurance:

People with health insurance seem to have higher wages than those without a health insurance.

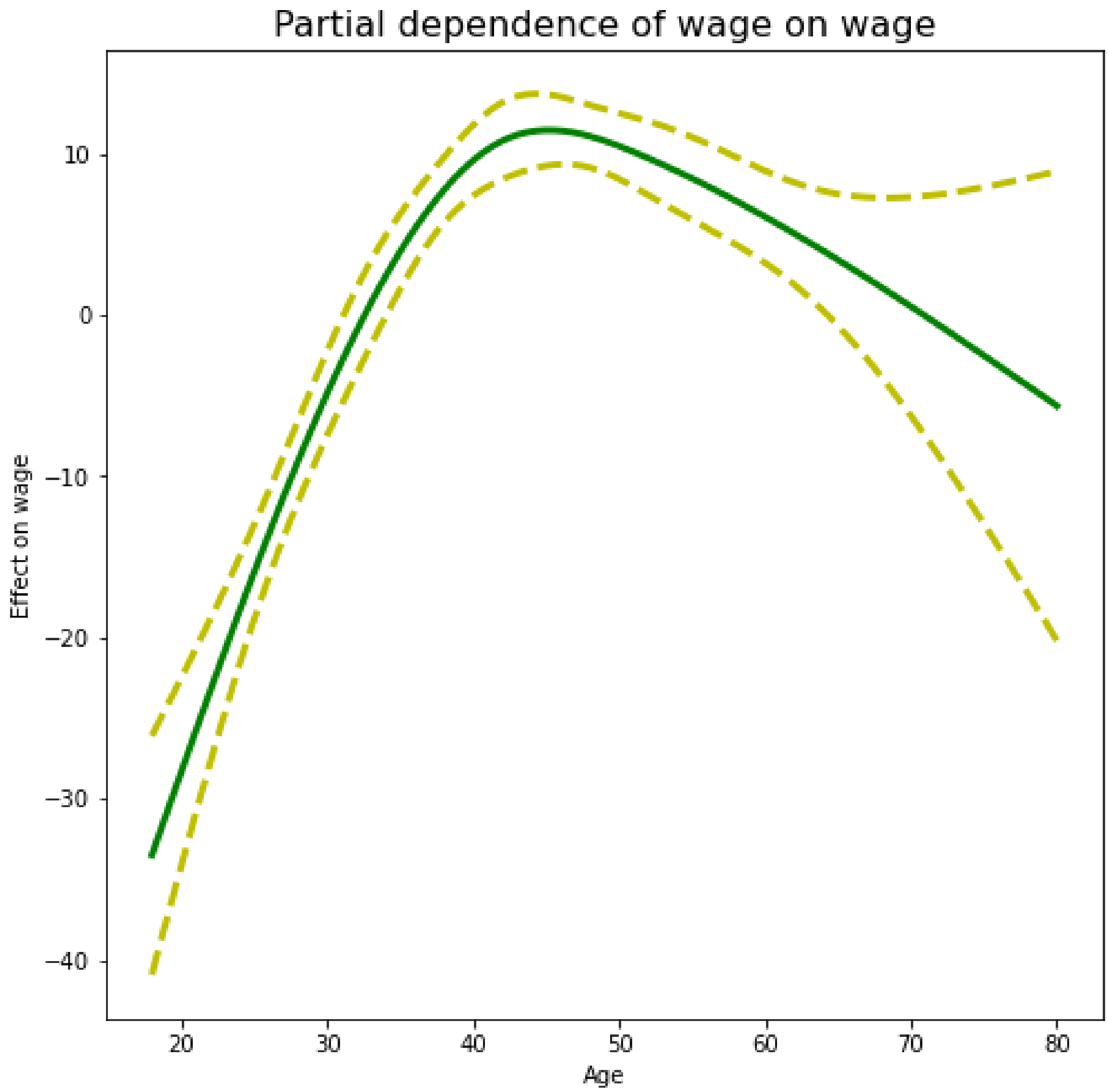
In [27]:
```
fig, ax = plt.subplots(figsize=(6, 6))
Wage.boxplot('wage', by='health_ins', ax=ax);
```



In [53]:
```
from pygam import (s as s_gam,
  l as l_gam,
  f as f_gam,
  LinearGAM,
  LogisticGAM)
from ISLP.transforms import (BSpline,
                             NaturalSpline)

from ISLP.models import bs, ns
from ISLP.pygam import (approx_lam,
  degrees_of_freedom,
  plot as plot_gam,
  anova as anova_gam)
from matplotlib.pyplot import subplots
import statsmodels.api as sm
from ISLP.models import (summarize,
                         poly,
                         ModelSpec as MS)
from statsmodels.stats.anova import anova_lm

ns_age = NaturalSpline(df=4).fit(Wage['age'])
ns_year = NaturalSpline(df=5).fit(Wage['year'])
Xs = [ns_age.transform(Wage['age']),
    ns_year.transform(Wage['year']),
    pd.get_dummies(Wage['education']).values]
X_bh = np.hstack(Xs)
gam_bh = sm.OLS(Wage['wage'], X_bh).fit()
```

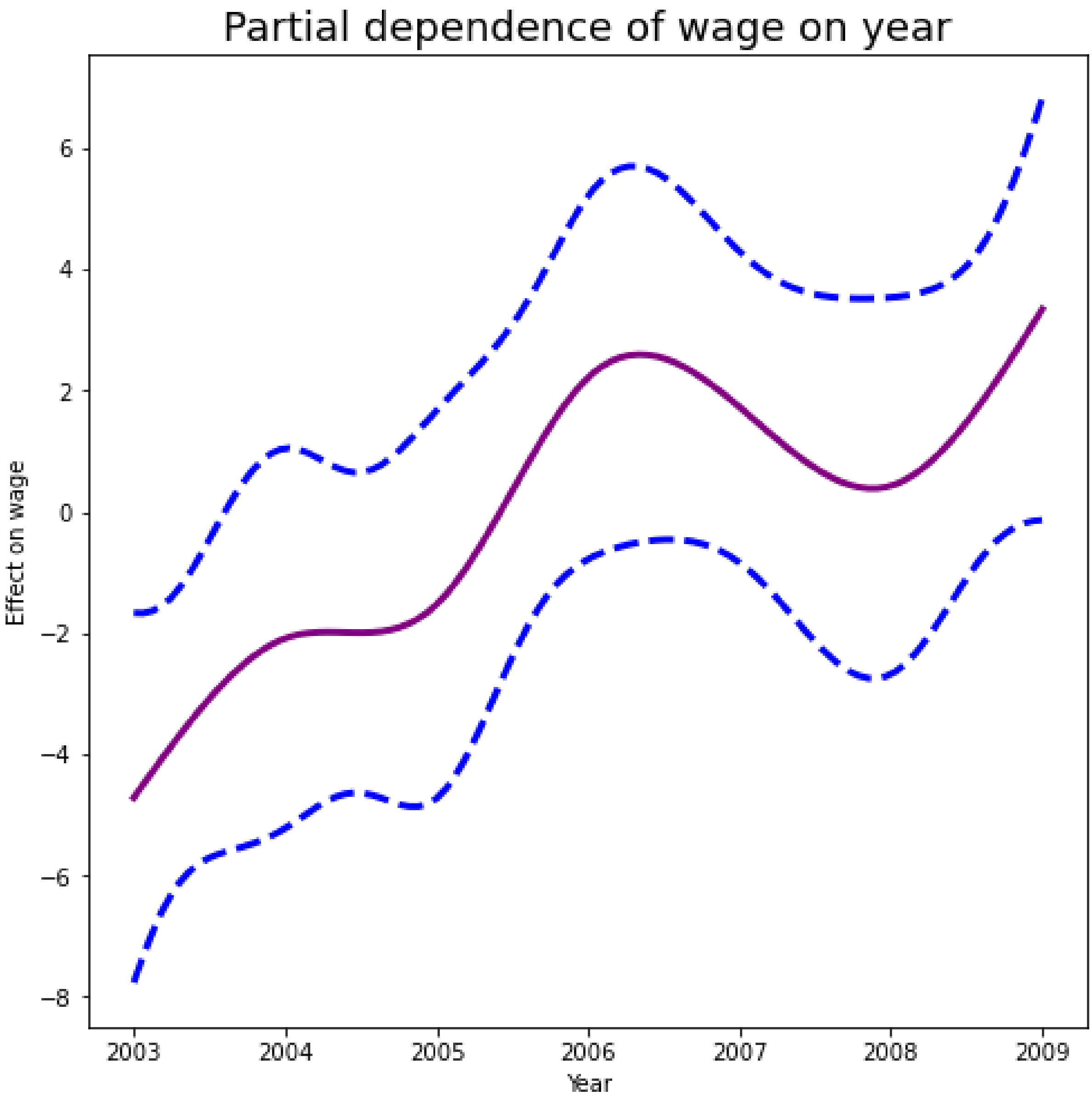
In [54]:
```
age=Wage['age']
```

In [56]:
```
age_grid = np.linspace(age.min(),
                       age.max(),
                       100)
X_age_bh = X_bh.copy()[:100]
X_age_bh[:] = X_bh[:].mean(0)[None,:]
X_age_bh[:,:4] = ns_age.transform(age_grid)
preds = gam_bh.get_prediction(X_age_bh)
bounds_age = preds.conf_int(alpha=0.05)
partial_age = preds.predicted_mean
center = partial_age.mean()
partial_age -= center
bounds_age -= center
fig, ax = subplots(figsize=(8,8))
ax.plot(age_grid, partial_age, 'g', linewidth=3)
ax.plot(age_grid, bounds_age[:,0], 'y--', linewidth=3)
ax.plot(age_grid, bounds_age[:,1], 'y--', linewidth=3)
ax.set_xlabel('Age')
ax.set_ylabel('Effect on wage')
ax.set_title('Partial dependence of wage on wage', fontsize=16);
```



In [57]:
```
year_grid = np.linspace(2003, 2009, 100)
year_grid = np.linspace(Wage['year'].min(),
                        Wage['year'].max(),
                        100)
X_year_bh = X_bh.copy()[:100]
X_year_bh[:] = X_bh[:].mean(0)[None,:]
X_year_bh[:,4:9] = ns_year.transform(year_grid)
preds = gam_bh.get_prediction(X_year_bh)
bounds_year = preds.conf_int(alpha=0.05)
partial_year = preds.predicted_mean
center = partial_year.mean()
partial_year -= center
bounds_year -= center
fig, ax = subplots(figsize=(8,8))
ax.plot(year_grid, partial_year , 'purple', linewidth=3)
ax.plot(year_grid, bounds_year[:,0], 'b--', linewidth=3)
ax.plot(year_grid, bounds_year[:,1], 'b--', linewidth=3)
ax.set_xlabel('Year')
ax.set_ylabel('Effect on wage')
ax.set_title('Partial dependence of wage on year', fontsize=18);
```

Isha Jain


Partial dependence of wage on year
Effect on wage
Year
6
4
2
0
−2
−4
−6
−8
2003
2004
2005
2006
2007
2008
2009

In [60]:
```
gam_bh.summary()
```

Out[60]:

OLS Regression Results

Dep. Variable:	wage	**R-squared:**	0.292
Model:	OLS	**Adj. R-squared:**	0.289
Method:	Least Squares	**F-statistic:**	94.86
Date:	Thu, 16 Nov 2023	**Prob (F-statistic):**	8.30e-213
Time:	21:54:36	**Log-Likelihood:**	-14931.
No. Observations:	3000	**AIC:**	2.989e+04
Df Residuals:	2986	**BIC:**	2.997e+04
Df Model:	13		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
x1	46.4460	3.732	12.446	0.000	39.129	53.763
x2	28.9349	3.884	7.449	0.000	21.319	36.551
x3	63.6722	9.231	6.898	0.000	45.572	81.772
x4	10.9669	7.650	1.434	0.152	-4.034	25.967
x5	1.8374	3.177	0.578	0.563	-4.392	8.067
x6	10.4409	3.790	2.755	0.006	3.010	17.872
x7	2.0020	3.399	0.589	0.556	-4.663	8.667
x8	9.6055	4.053	2.370	0.018	1.659	17.552
x9	5.8989	2.419	2.438	0.015	1.155	10.642
x10	43.8013	4.383	9.993	0.000	35.207	52.396
x11	54.7329	4.037	13.558	0.000	46.817	62.649
x12	67.1982	4.159	16.156	0.000	59.043	75.354
x13	81.9664	4.231	19.371	0.000	73.670	90.263
x14	106.3711	4.456	23.872	0.000	97.634	115.108

Omnibus:	1040.093	**Durbin-Watson:**	1.978
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	5576.947
Skew:	1.556	**Prob(JB):**	0.00
Kurtosis:	8.910	**Cond. No.**	16.2

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Isha Jain

In [67]:
```
model1=gam_bh
model1
```

Out[67]: <statsmodels.regression.linear_model.RegressionResultsWrapper at 0x271b25660a0>

In [63]:
```
from sklearn import preprocessing

poly = preprocessing.PolynomialFeatures(degree=5)
Xage = poly.fit_transform(Wage[['age']])
model2 = sm.OLS(Wage[['wage']], Xage)
model2 = model2.fit()
```

In [64]:
```
model2.summary()
```

Out[64]:

OLS Regression Results

Dep. Variable:	wage	**R-squared:**	0.087
Model:	OLS	**Adj. R-squared:**	0.085
Method:	Least Squares	**F-statistic:**	56.71
Date:	Thu, 16 Nov 2023	**Prob (F-statistic):**	1.67e-56
Time:	21:57:57	**Log-Likelihood:**	-15314.
No. Observations:	3000	**AIC:**	3.064e+04
Df Residuals:	2994	**BIC:**	3.068e+04
Df Model:	5		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-49.7046	161.435	-0.308	0.758	-366.239	266.830
x1	3.9930	20.110	0.199	0.843	-35.438	43.424
x2	0.2760	0.958	0.288	0.773	-1.603	2.155
x3	-0.0126	0.022	-0.577	0.564	-0.056	0.030
x4	0.0002	0.000	0.762	0.446	-0.000	0.001
x5	-9.157e-07	1.02e-06	-0.897	0.370	-2.92e-06	1.09e-06

Omnibus:	1094.840	**Durbin-Watson:**	1.961
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4940.229
Skew:	1.718	**Prob(JB):**	0.00
Kurtosis:	8.265	**Cond. No.**	9.39e+10

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 9.39e+10. This might indicate that there are
strong multicollinearity or other numerical problems.

In [65]:
```
poly = preprocessing.PolynomialFeatures(degree=2)
Xaged2 = poly.fit_transform(Wage[['age']])
model3 = sm.OLS(Wage[['wage']], Xaged2)
model3 = model3.fit()
```

In [66]:
```
model3.summary()
```

Out[66]:

OLS Regression Results

Dep. Variable:	wage	**R-squared:**	0.082
Model:	OLS	**Adj. R-squared:**	0.081
Method:	Least Squares	**F-statistic:**	134.0
Date:	Thu, 16 Nov 2023	**Prob (F-statistic):**	1.82e-56
Time:	22:00:23	**Log-Likelihood:**	-15321.
No. Observations:	3000	**AIC:**	3.065e+04
Df Residuals:	2997	**BIC:**	3.067e+04
Df Model:	2		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-10.4252	8.190	-1.273	0.203	-26.483	5.633
x1	5.2940	0.389	13.620	0.000	4.532	6.056
x2	-0.0530	0.004	-11.960	0.000	-0.062	-0.044

Omnibus:	1092.673	**Durbin-Watson:**	1.964
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	4915.802
Skew:	1.715	**Prob(JB):**	0.00
Kurtosis:	8.250	**Cond. No.**	2.45e+04

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.45e+04. This might indicate that there are
strong multicollinearity or other numerical problems.

In [68]:
```
sm.stats.anova_lm(model1, model2, model3)
```

Out[68]:

	df_resid	ssr	df_diff	ss_diff	F	Pr(>F)
0	2986.0	3.695815e+06	0.0	NaN	NaN	NaN
1	2994.0	4.770322e+06	-8.0	-1.074507e+06	83.976841	NaN
2	2997.0	4.793430e+06	-3.0	-2.310841e+04	4.816029	NaN

3 non-linear models were created to predict wage, model1 is a GAM using age, year and education. Model2 is a degree 5 polynomial function using age as the predictor and model3 is a 2-degree polynomial using age as the predictor.

Also, the dependence of wage is plotted based on age and year with 95 % confidence intervals plotted as dashed curves. The 2 curves indicate that wage is maximum around mid-40's and follows a somewhat inverted parabolic structure, whereas wage fluctuates over the years but shows a general increase with the same over a larger period.

From the ANOVA for the 3 models, we can conclude that model1 is probably better due to lower residual values.

Section 12.6

Conceptual

4. (a) Assuming 1,2,3,4 and 5 to be points, if the inter-cluster distances are not the same, which seems like a more likely scenario, complete linkage will occur higher on the tree as it uses the highest intra-cluster/ observation distance as compared to single linkage which uses the smallest intra- cluster distance. If the inter-observation distances are all the same [i.e., d (1,4) =d (1,5) =d (2,4) =d (2,5) =d (3,4) =d (3,5)], the fusion will occur at the same height.

(b) If the 2 clusters fusing are {5} and {6}, they would do so at the same height in both cases as there is just one point in each cluster, i.e., only one distance/ measure of dissimilarity which would be equal for both methods (single leveraging minimum inter-cluster distance as well as complete linkage which makes use of maximum inter-cluster distance)

Applied

10.

10

(a)

In [99]:
```
np.random.seed(22)

#Creating labels with 20 rows and 50 attributes using distributions in np.random and merging them
predictors1 = np.random.normal(2, 6, (20, 50))
label1_value=np.full((20, 1), 2)
predictors2 = np.random.uniform(-4, 4, (20, 50))
label2_value=np.full((20, 1), 3.6)
predictors3 = np.random.logistic(5,1.5, (20, 50))
label3_value=np.full((20, 1), 6.6)

dflabel1=np.append(predictors1,label1_value,axis=1)
dflabel2=np.append(predictors2,label2_value,axis=1)
dflabel3=np.append(predictors3,label3_value,axis=1)

df = pd.DataFrame(np.vstack((dflabel1,dflabel2,dflabel3)))
```

In [100]:
```
df.head()
```

Out[100]:

	0	1	2	3	4	5	6	7	8	9	...	41	42	43	4
0	1.448300	-6.780104	8.490750	0.564049	-0.946775	-4.013632	7.512929	-4.621793	5.758961	-1.369083	...	-1.833448	-4.647021	14.636875	-1.40432
1	6.472035	5.221460	-2.396331	5.334294	4.593736	1.185176	-3.646637	4.908604	-7.196929	4.429873	...	-14.617198	-0.953066	2.414889	-0.10001
2	10.983488	-1.879267	5.626947	3.289701	-0.541604	2.626515	3.869432	-0.802379	15.203294	-4.082427	...	-5.592429	5.105888	6.689630	10.33030
3	1.194186	9.536745	6.180521	20.452382	2.533078	-1.490061	10.112483	4.533632	0.332491	10.921192	...	-6.323294	5.336784	-7.627465	6.39327
4	0.115750	9.526357	-10.684383	4.053957	-5.453920	16.285404	-7.188348	-12.084062	7.538281	7.080706	...	7.843979	4.784937	-5.388304	-6.82606

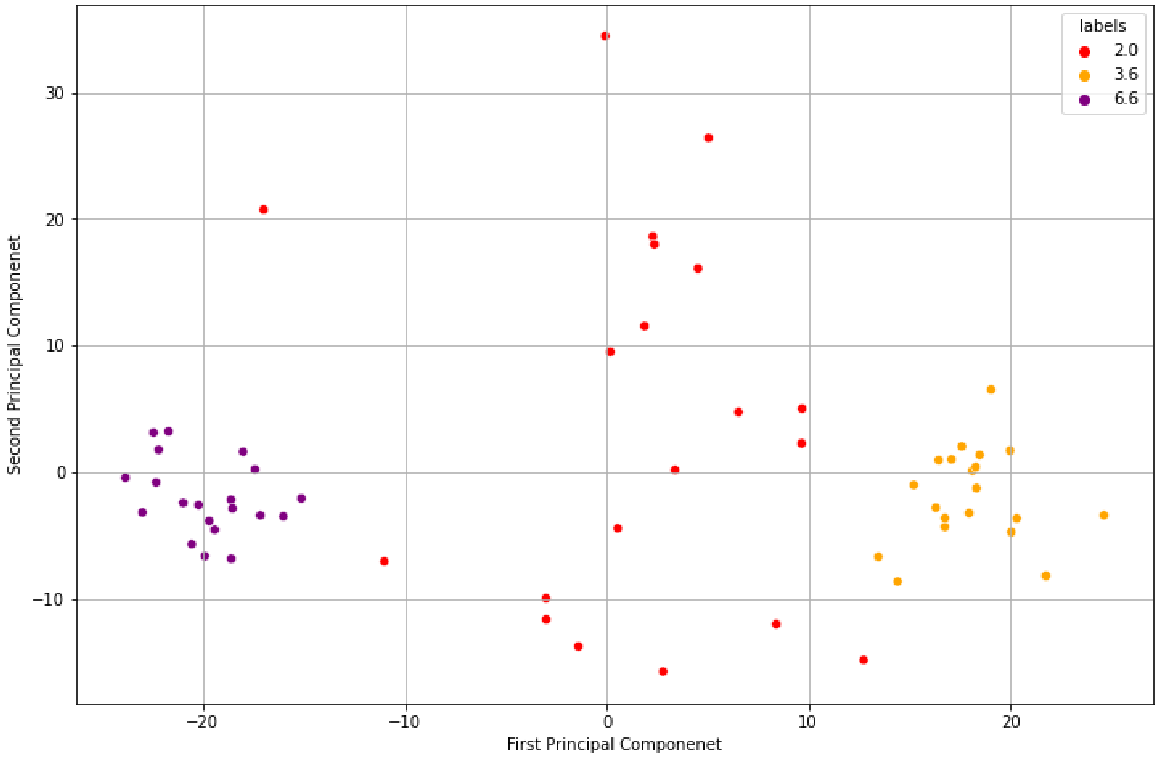
(b)

In [106]:
```
from sklearn.decomposition import PCA

pca = PCA(n_components=2)
#Excluding label column
principalComponents = pca.fit_transform(df.iloc[:,0:50])
df_pca = pd.DataFrame(principalComponents, columns=['first principal component', 'second principal component'])
df_pca['labels'] = df[[50]]
df_pca.head()
```

Out[106]:

	first principal component	second principal component	labels
0	6.538021	4.756910	2.0
1	2.294983	18.621326	2.0
2	-3.003483	-9.952631	2.0
3	-16.987435	20.727595	2.0
4	4.532156	16.087117	2.0

In [107]:
```
import seaborn as sns

fig = plt.figure(figsize=(12, 8))
sns.scatterplot(x="first principal component", y="second principal component", hue="labels", palette={2:'red', 3.6:'orange', 6.6:
plt.grid()
plt.xlabel("First Principal Componenet")
plt.ylabel("Second Principal Componenet")
plt.show()
```



(c)

In [22]:
```
from sklearn.cluster import KMeans

kmeans3 = KMeans(n_clusters=3, random_state=7,n_init=20).fit(df.iloc[:,0:50])
print(kmeans3.labels_)
```

```
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans i
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setti
P_NUM_THREADS=1.
  warnings.warn(
```

```
[0 0 2 2 0 0 0 2 1 0 0 0 1 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

In [23]:
```
print(kmeans3)
```

```
KMeans(n_clusters=3, n_init=20, random_state=7)
```

In [24]:
```
print(kmeans3.inertia_)
```

```
49518.50682749842
```

In [25]:
```
pd.crosstab(pd.Series(df[50]),
            pd.Series(kmeans3.labels_, name='K-means'))
```

Out[25]:

K-means	0	1	2
50			
2.0	14	2	4
3.6	20	0	0
6.6	0	0	20

The value of k (3) is a good choice as 3 labels exist; however, the clustering seems to do a poor job differentiating between 3.6 and 2, with one cluster containing only 2 points. This might be due to the large dimensionality.

(d)

In [26]:
```
kmeans2 = KMeans(n_clusters=2, random_state=7,n_init=20).fit(df.iloc[:,0:50])
print(kmeans2.labels_)
```

```
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWa
on Windows with MKL, when there are less chunks than available threads. You can av
P_NUM_THREADS=1.
  warnings.warn(
```

```
[1 1 0 0 1 1 1 0 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [27]:
```
print(kmeans2)
```

```
KMeans(n_clusters=2, n_init=20, random_state=7)
```

In [28]:
```
print(kmeans2.inertia_)
```

```
52432.9608525129
```

In [29]:
```
pd.crosstab(pd.Series(df[50]),
            pd.Series(kmeans2.labels_, name='K-means'))
```

Out[29]:

K-means	0	1
50		
2.0	4	16
3.6	0	20
6.6	20	0

Above are the results for k means clustering with k=2. From this we can conclude that the datapoints with labels 2 and 3.6 seem closer (as they are put majorly in the same cluster)

(e)

In [30]:
```
kmeans4 = KMeans(n_clusters=4, random_state=7,n_init=20).fit(df.iloc[:,0:50])
print(kmeans4.labels_)
```

```
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMe
on Windows with MKL, when there are less chunks than available threads. You can avoid it by
P_NUM_THREADS=1.
  warnings.warn(
```

```
[0 0 3 1 0 3 0 1 2 3 0 3 2 3 0 0 1 0 0 0 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3 3
 3 3 3 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [31]:
```
print(kmeans4)
```

```
KMeans(n_clusters=4, n_init=20, random_state=7)
```

In [32]:
```
print(kmeans4.inertia_)
```

```
45500.79092808049
```

In [33]:
```
pd.crosstab(pd.Series(df[50]),
            pd.Series(kmeans4.labels_, name='K-means'))
```

Out[33]:

K-means	0	1	2	3
50				
2.0	10	3	2	5
3.6	0	0	0	20
6.6	0	20	0	0

Above are the results for k-means clustering with k=4. Even with k=4, some points with label 2 are clustered with points having label 3.6 and 6.6.

(f)

In [34]:
```
df_pca[['first principal component', 'second principal component']]
```

Out[34]:

	first principal component	second principal component
0	6.538021	4.756910
1	2.294983	18.621326
2	-3.003483	-9.952631
3	-16.987435	20.727595
4	4.532156	16.087117
5	1.881930	11.545489
6	2.370799	17.995765
7	-11.005126	-7.045617
8	0.545708	-4.425503
9	2.793559	-15.738239
10	3.392447	0.171000
11	12.741237	-14.836237
12	-2.993646	-11.622669
13	9.663101	2.280302
14	9.689828	5.027906
15	-0.072335	34.433186
16	-1.394507	-13.746910
17	5.045920	26.406936
18	8.410040	-11.984462

In [37]:
```
kmeanspca = KMeans(n_clusters=3, random_state=0).fit(df_pca[['first principal component', 'second principal component']])
print(kmeanspca.labels_)
```

```
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1412: FutureWarning: The default value of `n_init` will
hange from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is known to have a memory lea
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable
P_NUM_THREADS=1.
  warnings.warn(
```

```
[1 2 0 2 2 2 2 0 1 1 1 1 0 1 1 2 0 2 1 2 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
```

In [38]:
```
pd.crosstab(pd.Series(df[50]),
            pd.Series(kmeanspca.labels_, name='K-means'))
```

Out[38]:

K-means	0	1	2
50			
2.0	4	8	8
3.6	0	20	0
6.6	20	0	0

In [41]:
```
kmeanspca.inertia_
```

Out[41]: 3981.5355595696533

k-means clustering with k=3 on the first 2 principal components gives a lower inertia than k=3 on the original dataset as the points end up closer (less sparse dataset).

(g)

In [40]:
```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler(with_std=True,
                        with_mean=True)
df_scaled = scaler.fit_transform(df)
```

In [44]:
```
type(df_scaled)
```

Out[44]:
```
numpy.ndarray
```

In [45]:
```
df_scaled = pd.DataFrame(data=df_scaled)
```

In [46]:
```
df_scaled
```

Out[46]:

	0	1	2	3	4	5	6	7	8	
0	-0.225492	-2.180885	1.361168	-0.427359	-0.852374	-1.834303	1.246368	-1.215301	0.697963	-0
1	1.077849	0.631822	-0.945341	0.447935	0.400328	-0.436564	-1.268286	0.431024	-1.954319	0
2	2.248285	-1.032317	0.754450	0.072772	-0.760765	-0.049049	0.425356	-0.555518	2.631372	-1
3	-0.291418	1.643160	0.871729	3.221959	-0.065584	-1.155822	1.832141	0.366250	-0.412924	1
4	-0.571204	1.640725	-2.701226	0.213005	-1.871433	3.623248	-2.066361	-2.504368	1.062219	0
5	0.693508	2.930594	-1.556165	1.646545	-1.993173	0.029943	1.638255	2.608093	-2.091529	1
6	-0.166251	0.100939	1.877907	0.447423	-0.540334	1.218081	-0.836858	1.013218	-2.018114	1
7	-1.048000	-1.225118	1.207758	-0.207837	0.503143	1.481727	-0.996432	1.694214	1.239980	-1
8	-0.771508	0.232033	-3.180201	0.320157	0.509853	1.785090	0.673831	1.201510	0.049610	-0
9	0.041635	-0.990986	-0.170057	-2.676600	-0.759878	0.063013	0.018765	0.360340	-0.950845	0

In [47]:
```
type(df_scaled)
```

Out[47]:
```
pandas.core.frame.DataFrame
```

In [48]:
```
kmeans_scaled = KMeans(n_clusters=3, random_state=7,n_init=20).fit(df_scaled.iloc[:,0:50])
print(kmeans_scaled.labels_)
```

```
C:\Users\ishaj\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1436: UserWarning: KMeans is
on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting
P_NUM_THREADS=1.
  warnings.warn(
```

```
[0 0 2 2 1 0 0 2 0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2]
```

In [49]:
```
kmeans_scaled.inertia_
```

Out[49]:
```
2236.167837836526
```

In [50]:
```
pd.crosstab(pd.Series(df[50]),
            pd.Series(kmeans_scaled.labels_, name='K-means'))
```

Out[50]:

K-means	0	1	2
50			
2.0	15	2	3
3.6	20	0	0
6.6	0	0	20

k-means clustering with k=3 on scaled data with unit standard deviation seems to provide the best result having lowest inertia and relatively accurate clustering based on labels.