1. Clustering

Sex

```
Values 1: Male 2: Female
```

Age

How have you been feeling over the last two weeks

q1 - I have felt cheerful and in good spirits

q2 - I have felt calm and relaxed

q3 - I have felt active and vigorous

q4 - I woke up feeling fresh and rested

q5 - My daily life has been filled with things that interest me

Values:

1. All of the time.
2. Most of the time
3. More than half of the time
4. Less than half of the time
5. Some of the time
6. At no time

Please tell me how often you feel this way

w1 - At my work I feel full of energy

w2 - I am enthusiastic about my job

w3 - Time flies when I am working

w4 - In my opinion, I am good at my job

Values for variables Q90a to Q90f

1. Always.
2. Most of the time
3. Sometimes
4. Rarely
5. Never

PCA, Elbow method Clustering, K-means clustering

In [204]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

plt.style.use('seaborn')
plt.rc('font', size=14)
plt.rc('figure', titlesize=18)
plt.rc('axes', labelsize=15)
plt.rc('axes', titlesize=18)

init_df = pd.read_csv('EWCS_2016.csv')
init_df = init_df.replace(-999, np.nan)
init_df = init_df.dropna()
init_df.columns = ["sex", "age", "q1", "q2", "q3", "q4", "q5", "w1", "w2", "w3", "w4"]
init_df.head()
```

Out[204]:

	sex	age	q1	q2	q3	q4	q5	w1	w2	w3	w4
0	1.0	63.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0
1	2.0	58.0	2.0	3.0	2.0	3.0	2.0	2.0	3.0	2.0	2.0
2	2.0	32.0	2.0	2.0	3.0	2.0	3.0	2.0	2.0	2.0	2.0
3	1.0	35.0	3.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0
4	2.0	27.0	2.0	2.0	3.0	3.0	2.0	2.0	4.0	2.0	2.0

PCA

In [554]:

```python
from sklearn.decomposition import PCA
from sklearn import preprocessing


df = init_df.copy(deep=True)

def do_pca(init_df, delta=None, devide=True,
           return_scree=True, return_info=True):

    """
    A cluster cannot be smaller than
    5% of the data.

    Samples are grouped in clusters if
    length between them is less than
    the sum of 10% of each dimention.
    """

    # Scaling
    df = init_df.copy(deep=True)
    df = preprocessing.scale(df)

    # Model
    pca = PCA()
    pca.fit(df)
    pca_data = pca.transform(df)

    per_var = np.round(pca.explained_variance_ratio_ * 100, 1)

    labels = ['PC' + str(i) for i in range(1, len(per_var) + 1)]

    # Plots
    # Scree plot
    if return_scree:
        plt.bar(x=range(1,len(per_var)+1), height=per_var, tick_label=labels)
        plt.ylabel('Percentage of Explained Variance')
        plt.xlabel('Principal Component')
        plt.title('Scree Plot')
        plt.show()


    # PC1 vs PC2
    pca_df = pd.DataFrame(pca_data, index=init_df.index, columns=labels)

    plt.scatter(pca_df.PC1, pca_df.PC2)
    plt.title('PCA Graph')
    plt.xlabel(f'PC1 - {per_var[0]}')
    plt.ylabel(f'PC2 - {per_var[1]}')

    if devide:
        seen = set()
        clusters = {}
        if not delta:
            d1 = (pca_df.PC1.max() - pca_df.PC1.min()) / 10
            d2 = (pca_df.PC2.max() - pca_df.PC2.min()) / 10
            delta = (d1 + d2) / 2

        for sample in pca_df.index:
            x, y = pca_df.PC1.loc[sample], pca_df.PC2.loc[sample]

            found = False
            for s in seen:
                x1, y1 = pca_df.PC1.loc[s], pca_df.PC2.loc[s]
                if sum(np.power(np.array(x1, y1) - np.array((x, y)), 2)) < delta**2:
                    found = True
                    break

            if found:
                clusters[s].append(sample)
            else:
                clusters[sample] = [sample]
                seen.add(sample)

        clusts = {}
        for c in clusters:
            l = len(clusters[c])
            if l > len(df)/20:
                clusts[c] = clusters[c]
                x, y = pca_df.PC1.loc[c], pca_df.PC2.loc[c]
                plt.annotate(f'#{c}, sz={l}', (x, y))

        if return_info:
            print(f'Centers of clusters are represented by the foolowing samples: \n', *list(clusts.ke
ys()))
            return clusts

    plt.show()
    return None
```

Raw Data

In [555]:

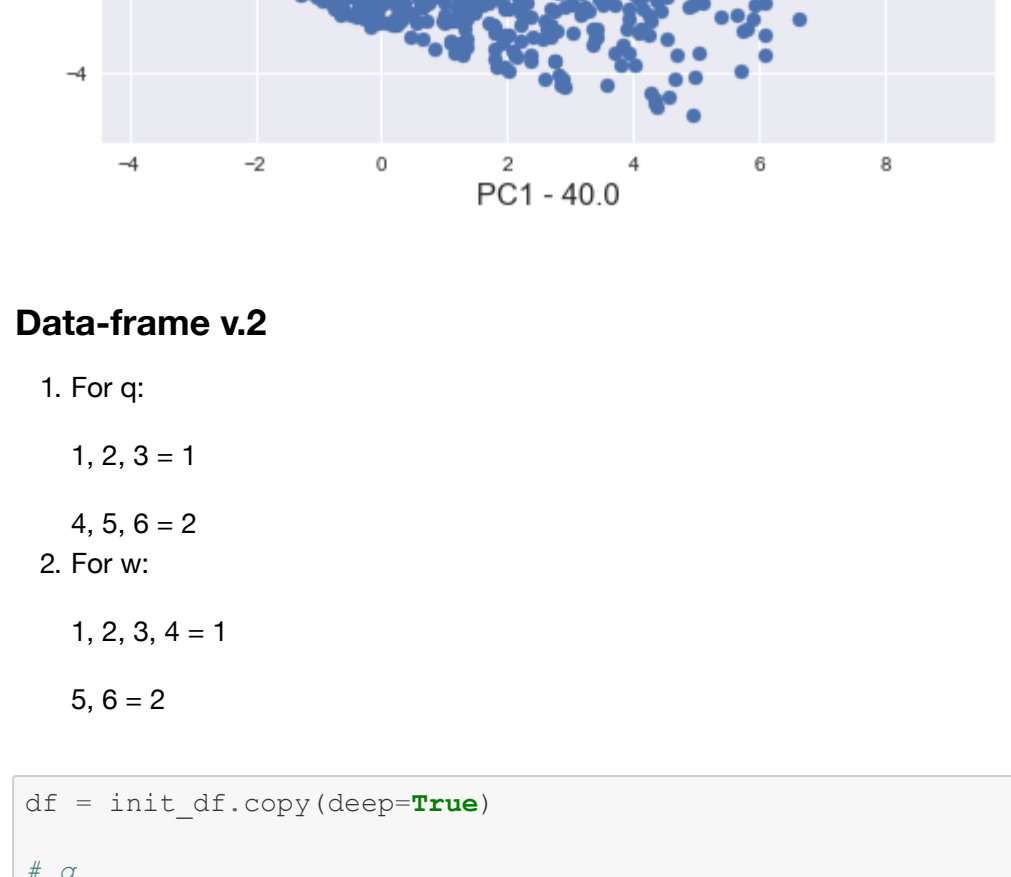
```python
df = init_df.copy(deep=True)
df.head()
```

Out[555]:

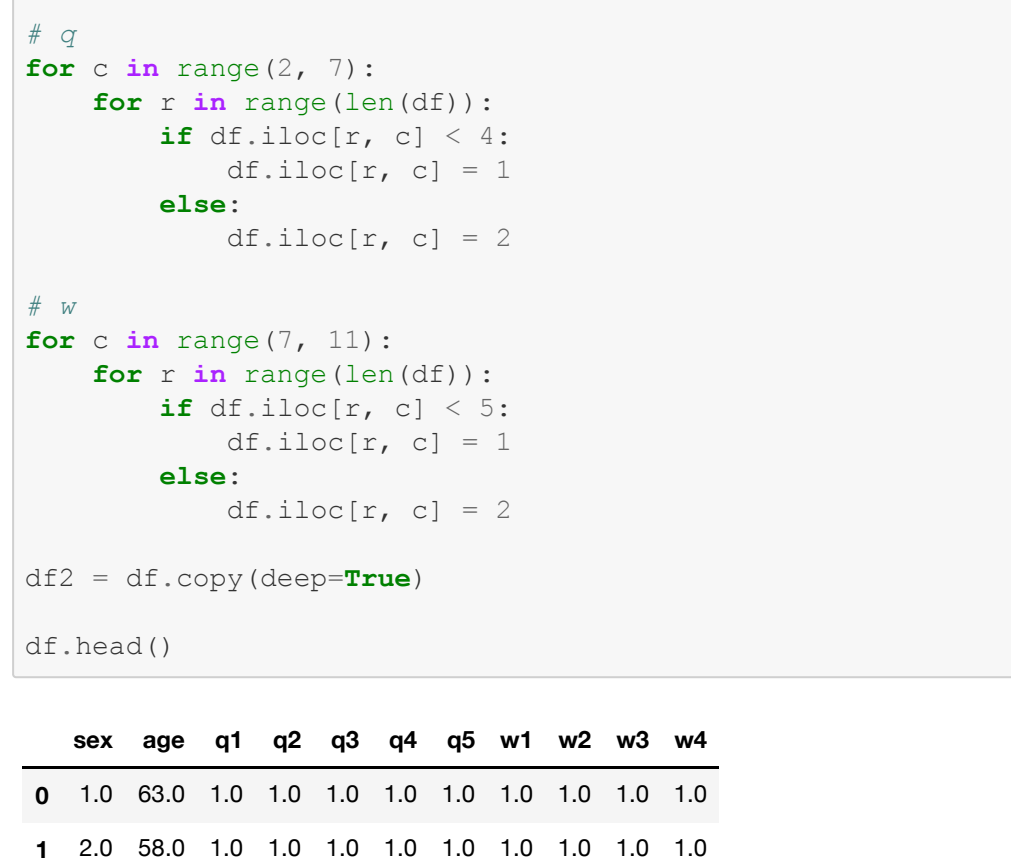
	sex	age	q1	q2	q3	q4	q5	w1	w2	w3	w4
0	1.0	63.0	3.0	3.0	3.0	3.0	3.0	2.0	2.0	2.0	2.0
1	2.0	58.0	2.0	3.0	2.0	3.0	2.0	2.0	3.0	2.0	2.0
2	2.0	32.0	2.0	2.0	3.0	2.0	3.0	2.0	2.0	2.0	2.0
3	1.0	35.0	3.0	2.0	2.0	2.0	3.0	2.0	2.0	2.0	2.0
4	2.0	27.0	2.0	2.0	3.0	3.0	2.0	2.0	4.0	2.0	2.0

In [556]:

```python
clusts = do_pca(df, delta=8)
```




```
Centers of clusters are represented by the foolowing samples: 
 0
```



Data-frame v.2

1. For q:

   1, 2, 3 = 1

   4, 5, 6 = 2
2. For w:

   1, 2, 3, 4 = 1

   5, 6 = 2

In [210]:

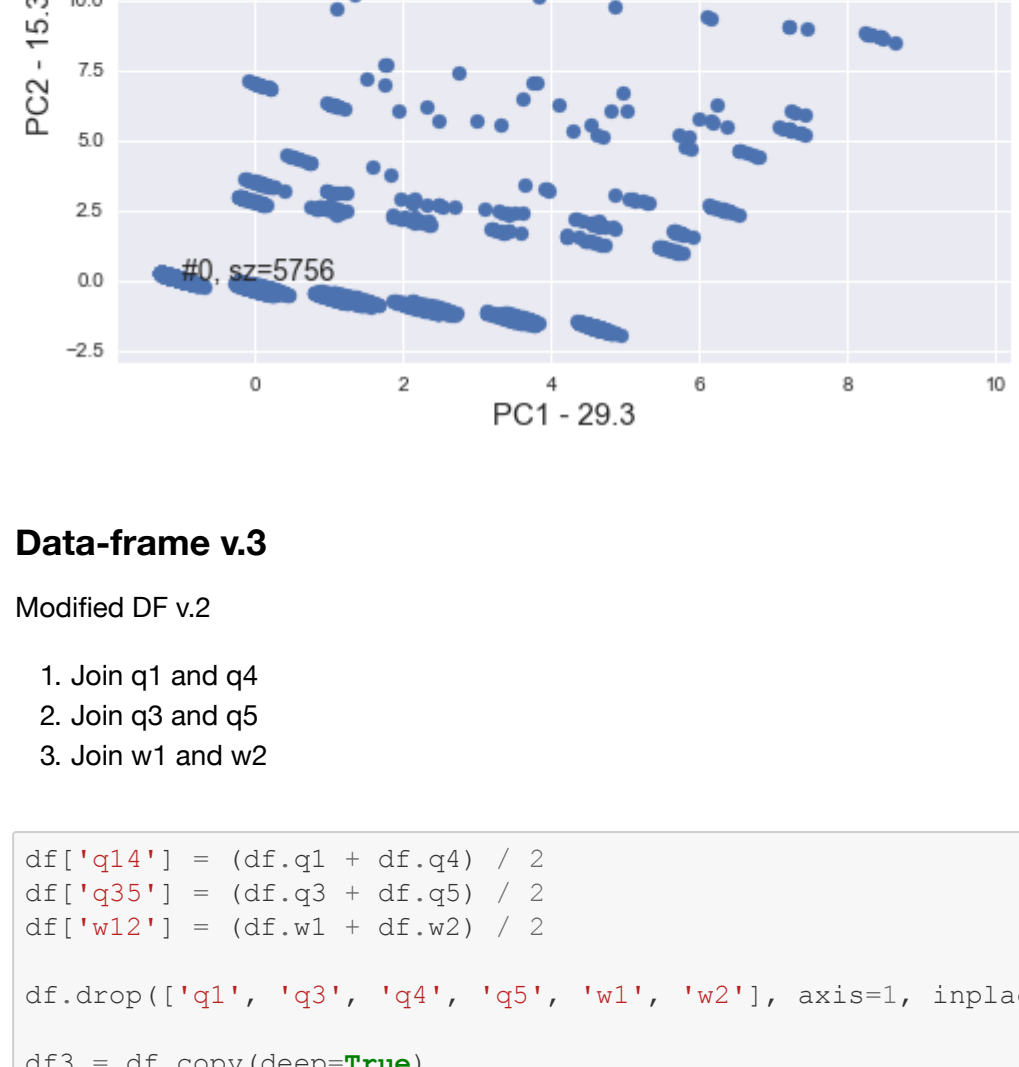
```python
df = init_df.copy(deep=True)

# q
for c in range(2, 7):
    for r in range(len(df)):
        if df.iloc[r, c] < 4:
            df.iloc[r, c] = 1
        else:
            df.iloc[r, c] = 2

# w
for c in range(7, 11):
    for r in range(len(df)):
        if df.iloc[r, c] < 5:
            df.iloc[r, c] = 1
        else:
            df.iloc[r, c] = 2

df2 = df.copy(deep=True)

df.head()
```

Out[210]:

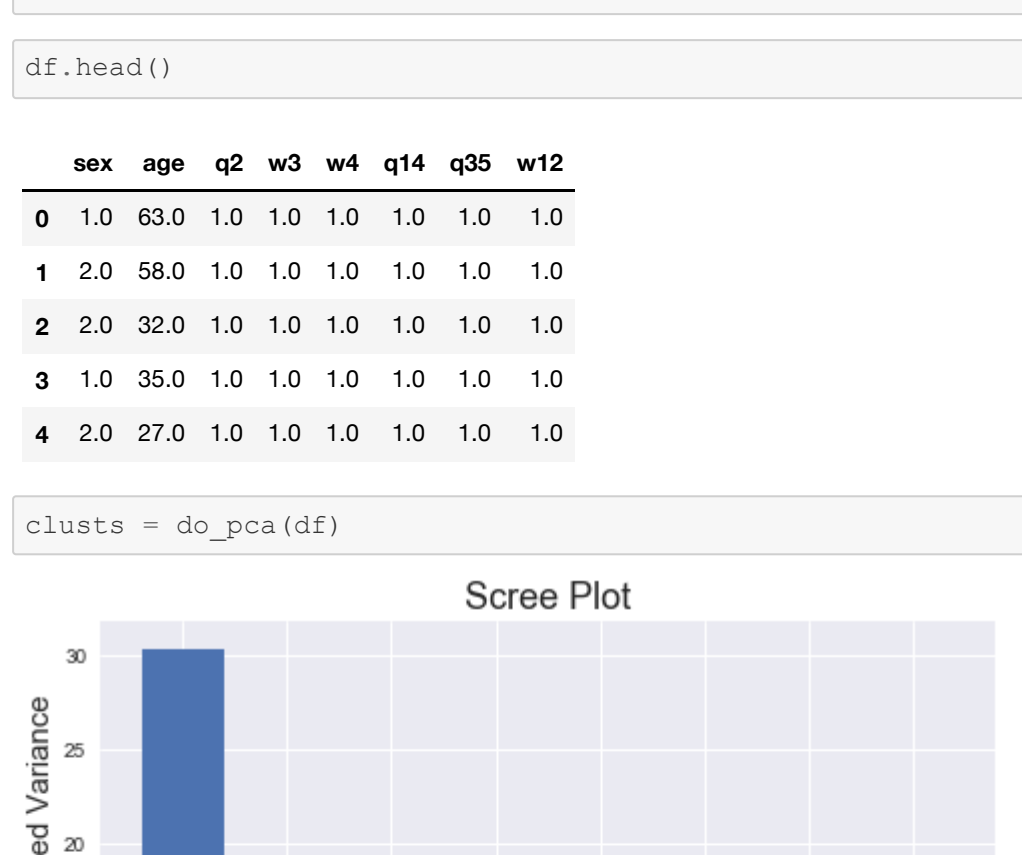
	sex	age	q1	q2	q3	q4	q5	w1	w2	w3	w4
0	1.0	63.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
1	2.0	58.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
2	2.0	32.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	35.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	2.0	27.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

In [208]:

```python
clusts = do_pca(df2)
```




```
Centers of clusters are represented by the foolowing samples: 
 0
```



Data-frame v.3

Modified DF v.2

1. Join q1 and q4
2. Join q3 and q5
3. Join w1 and w2

In [211]:

```python
df['q14'] = (df.q1 + df.q4) / 2
df['q35'] = (df.q3 + df.q5) / 2
df['w12'] = (df.w1 + df.w2) / 2

df.drop(['q1', 'q3', 'q4', 'q5', 'w1', 'w2'], axis=1, inplace=True)

df3 = df.copy(deep=True)
```

In [212]:

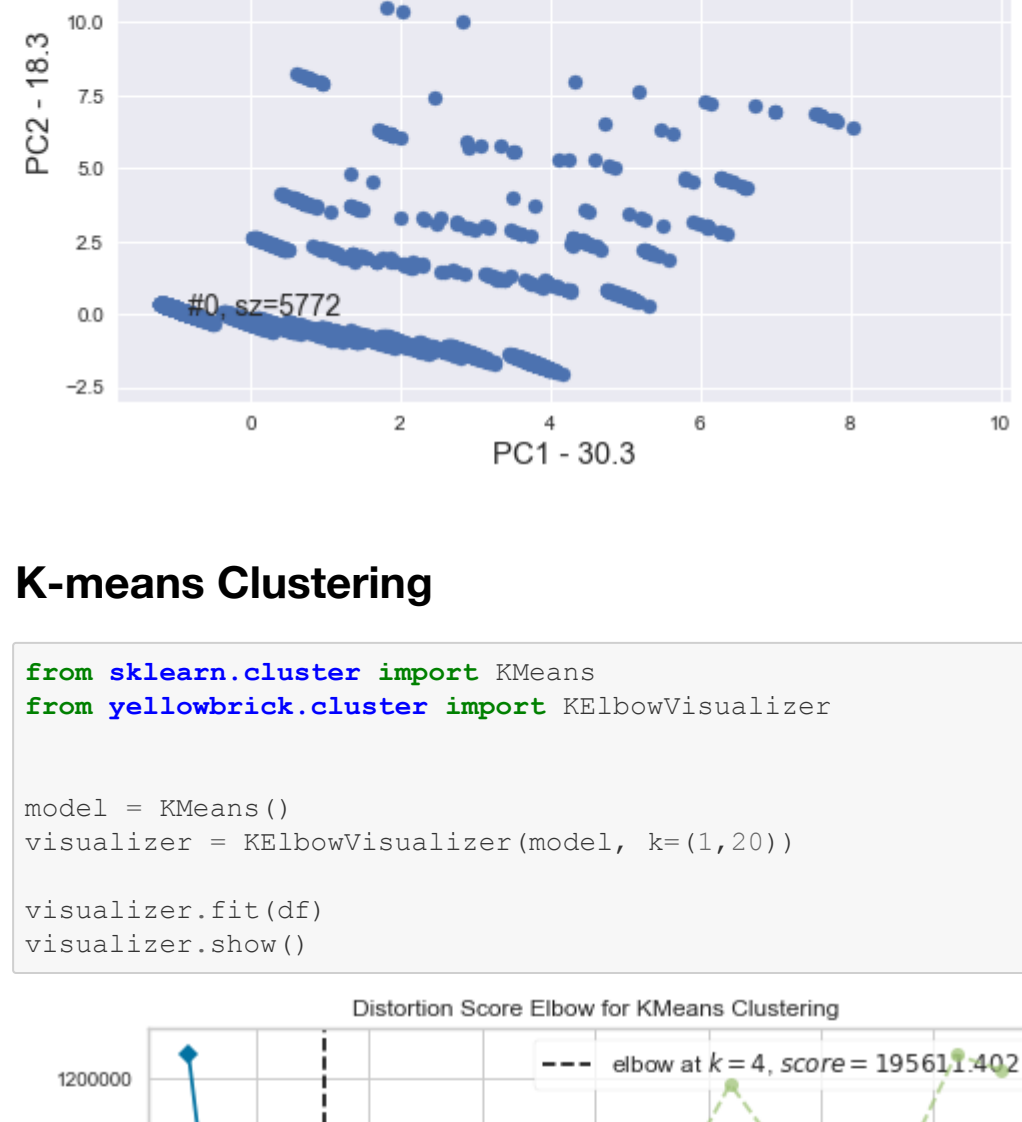
```python
df.head()
```

Out[212]:

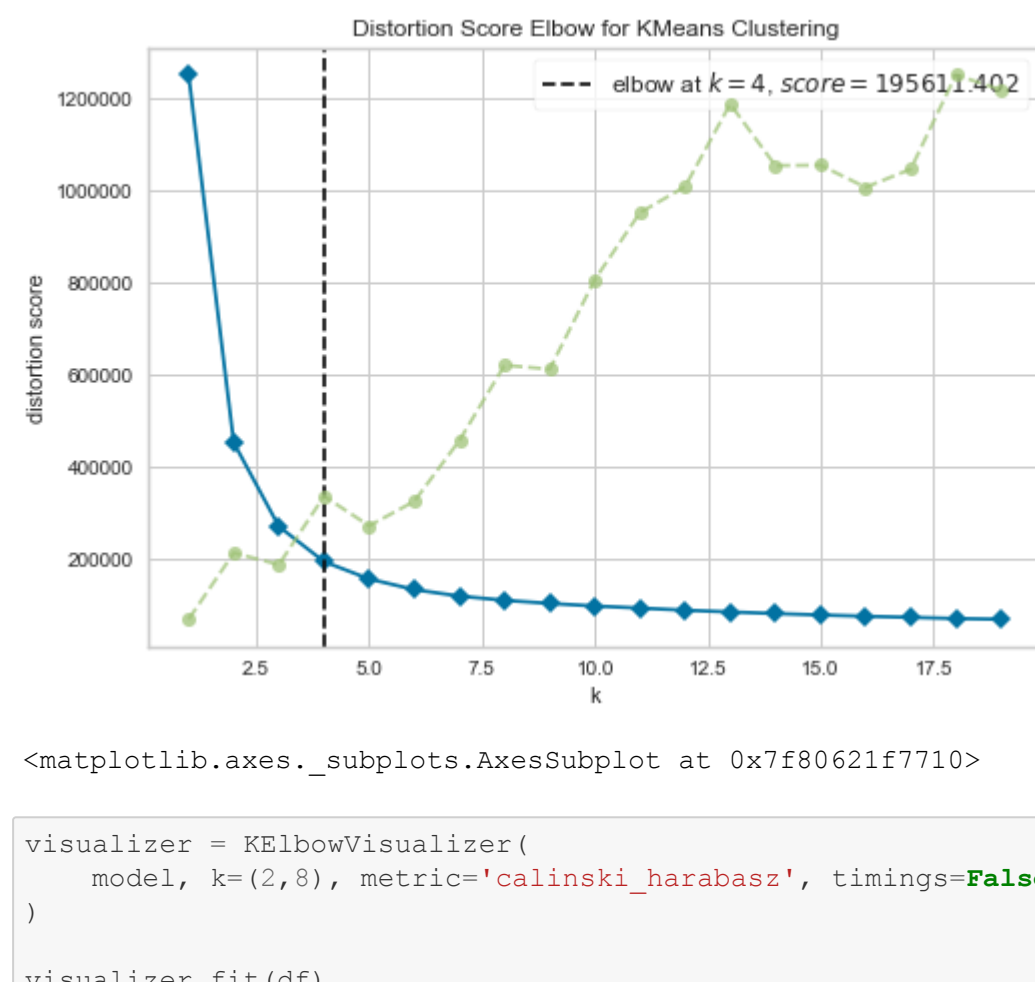
	sex	age	q2	w3	w4	q14	q35	w12
0	1.0	63.0	1.0	1.0	1.0	1.0	1.0	1.0
1	2.0	58.0	1.0	1.0	1.0	1.0	1.0	1.0
2	2.0	32.0	1.0	1.0	1.0	1.0	1.0	1.0
3	1.0	35.0	1.0	1.0	1.0	1.0	1.0	1.0
4	2.0	27.0	1.0	1.0	1.0	1.0	1.0	1.0

In [213]:

```python
clusts = do_pca(df)
```



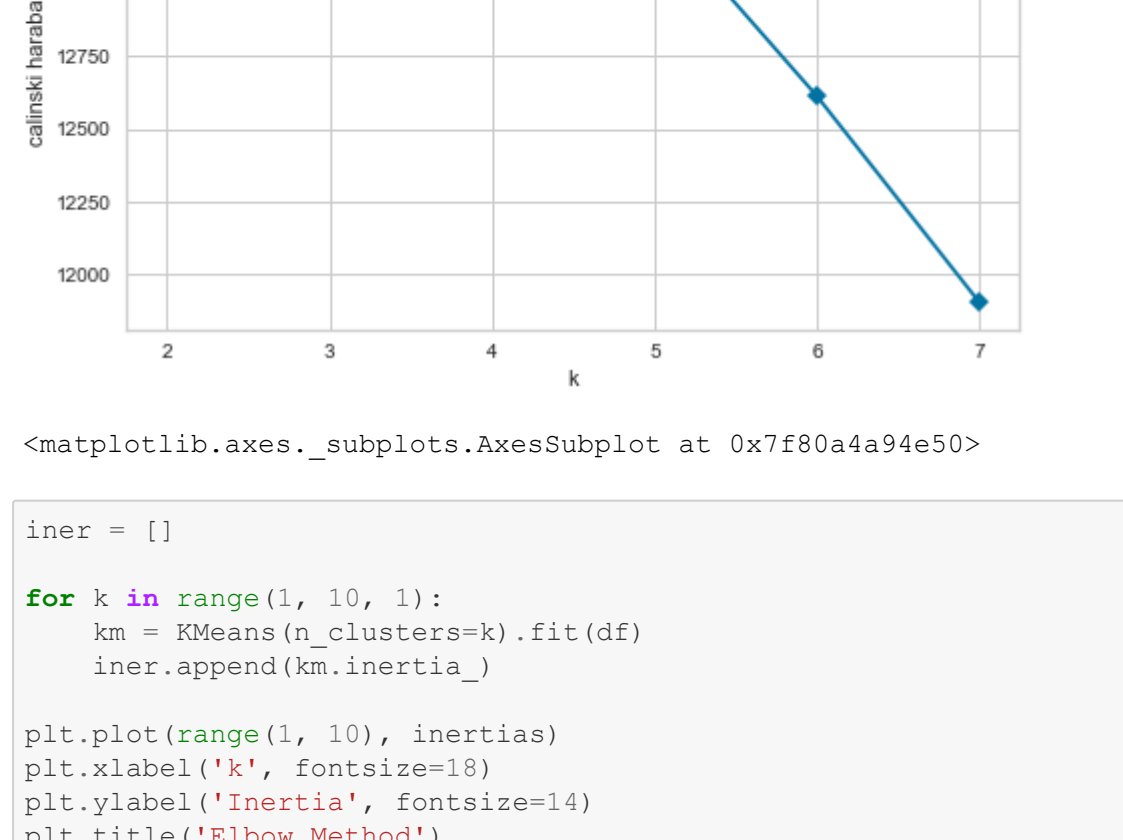

```
Centers of clusters are represented by the foolowing samples: 
 0
```



K-means Clustering

In [563]:

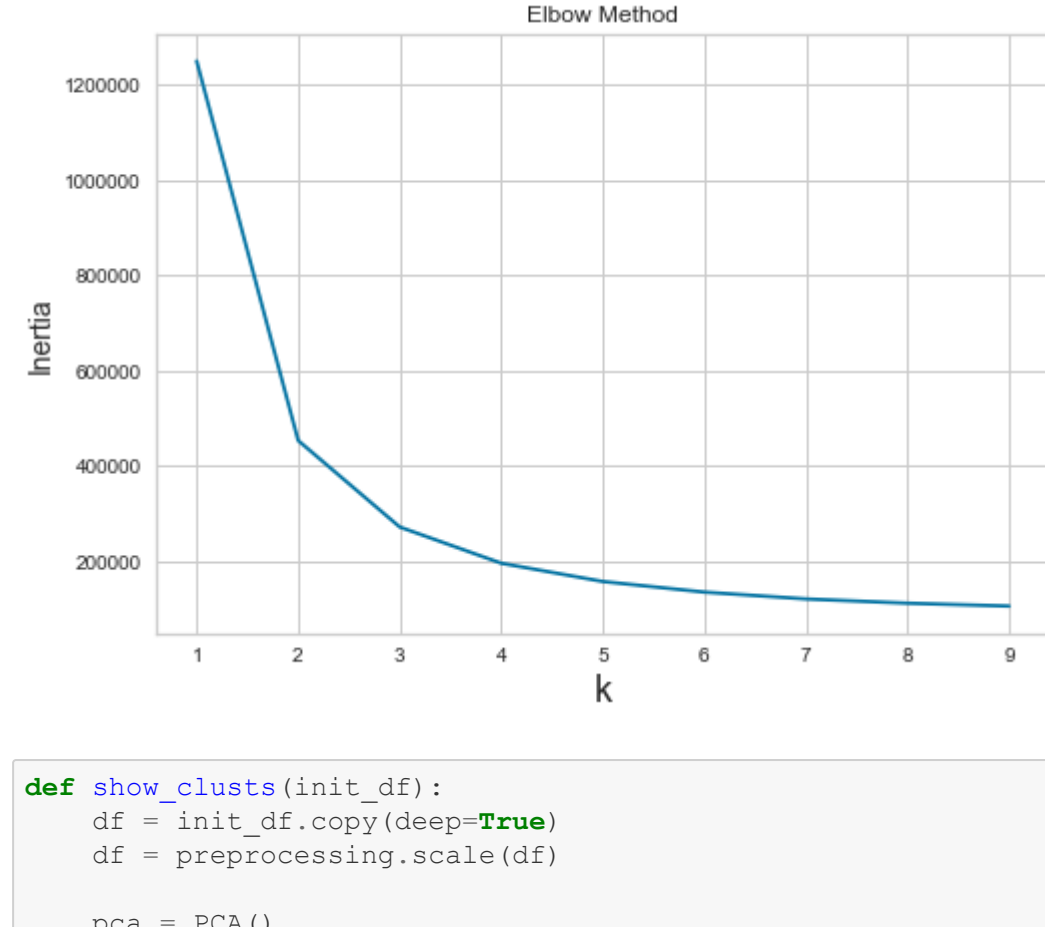
```python
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer


model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,20))

visualizer.fit(df)
visualizer.show()
```



Out[563]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f80621f7710>`

In [574]:

```python
visualizer = KElbowVisualizer(
    model, k=(2,8), metric='calinski_harabasz', timings=False, locate_elbow=False
)

visualizer.fit(df)
visualizer.show()
```

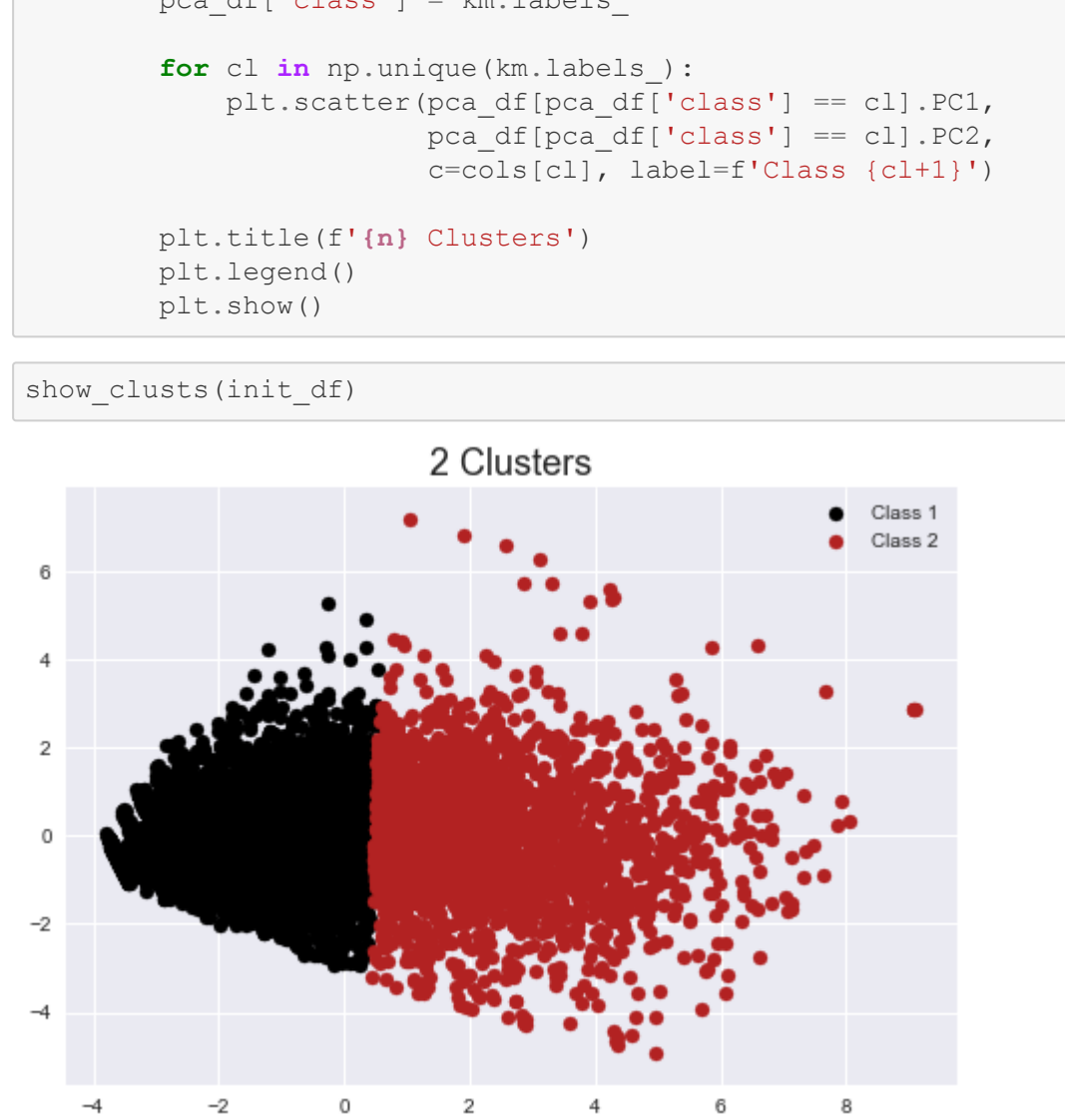


Out[574]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f80a4a94e50>`

In [570]:

```python
iner = []

for k in range(1, 10, 1):
    km = KMeans(n_clusters=k).fit(df)
    iner.append(km.inertia_)

plt.plot(range(1, 10), inertias)
plt.xlabel('k', fontsize=18)
plt.ylabel('Inertia', fontsize=14)
plt.title('Elbow Method')
```

Out[570]: `Text(0.5, 1.0, 'Elbow Method')`

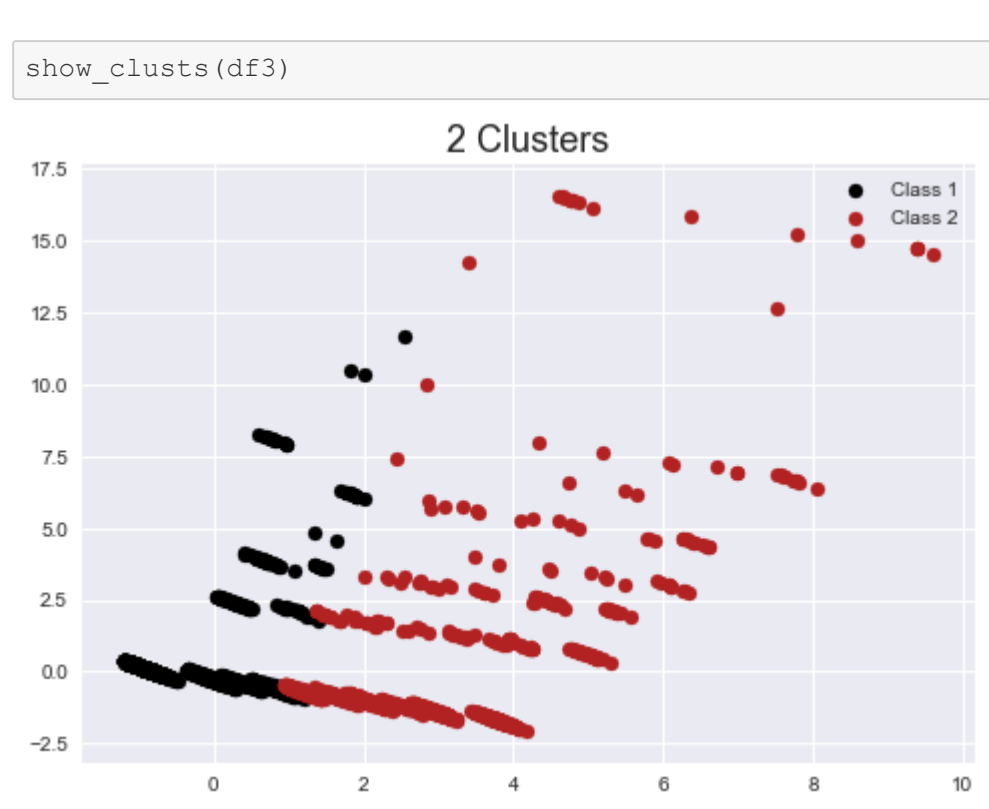


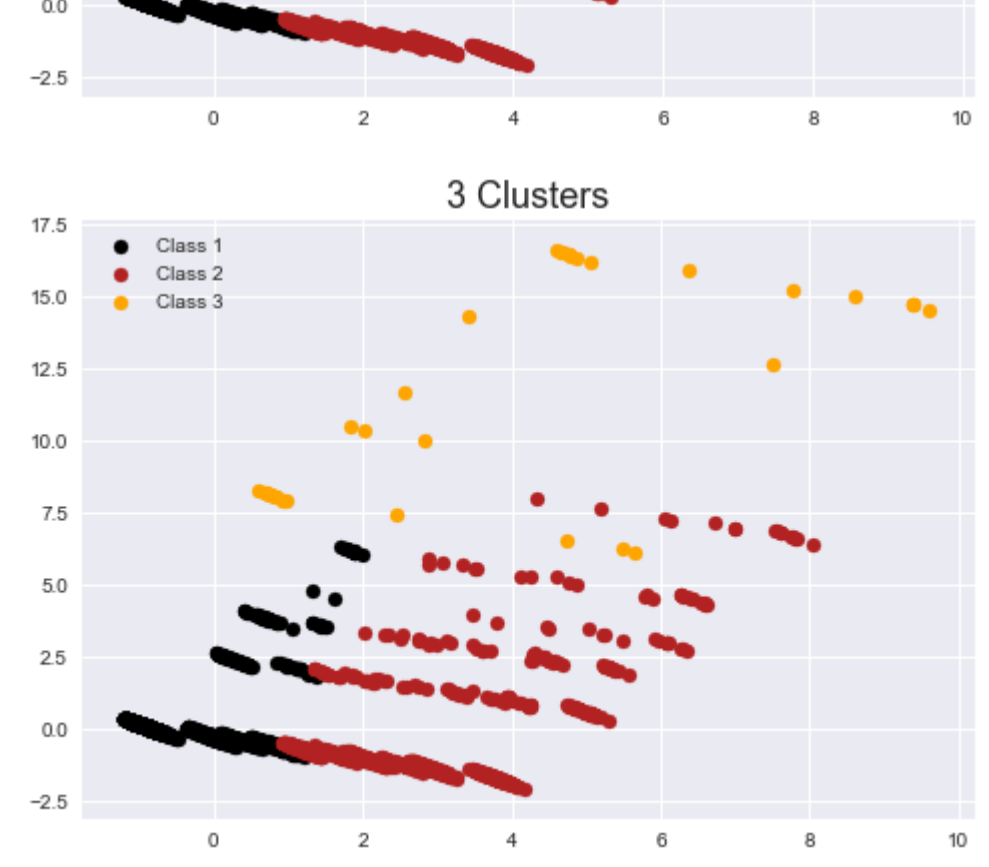
In [557]:

```python
def show_clusts(init_df):
    df = init_df.copy(deep=True)
    df = preprocessing.scale(df)

    pca = PCA()
    pca.fit(df)
    pca_data = pca.transform(df)
    per_var = np.round(pca.explained_variance_ratio_ * 100, 1)
    labels = ['PC' + str(i) for i in range(1, len(per_var) + 1)]

    pca_df = pd.DataFrame(pca_data, index=init_df.index, columns=labels)

    cols = ['black', 'firebrick', 'orange', 'lightgreen',
            'mediumturquoise', 'dodgerblue', 'stateblue', 'violet']

    for n in range(2, 5):
        km = KMeans(n_clusters=n, n_init=10, random_state=0).fit(df)
        pca_df['class'] = km.labels_

        for cl in np.unique(km.labels_):
            plt.scatter(pca_df[pca_df['class'] == cl].PC1,
                        pca_df[pca_df['class'] == cl].PC2,
                        c=cols[cl], label=f'Class {cl+1}')

        plt.title(f'{n} Clusters')
        plt.legend()
        plt.show()
```

In [558]:

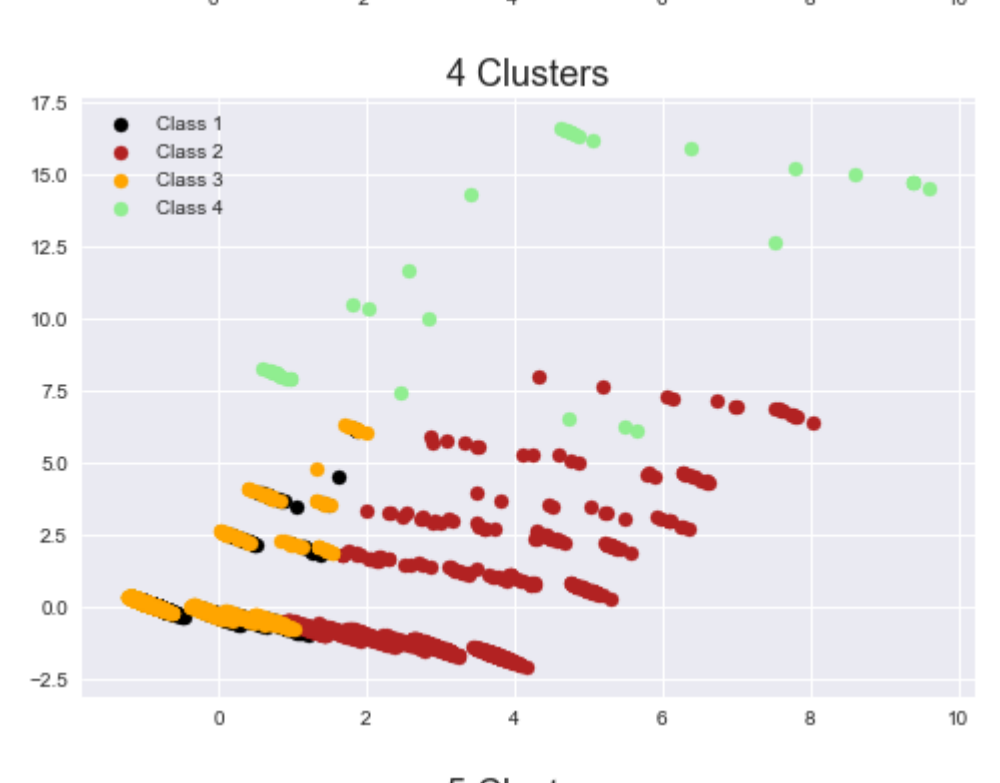
```python
show_clusts(init_df)
```

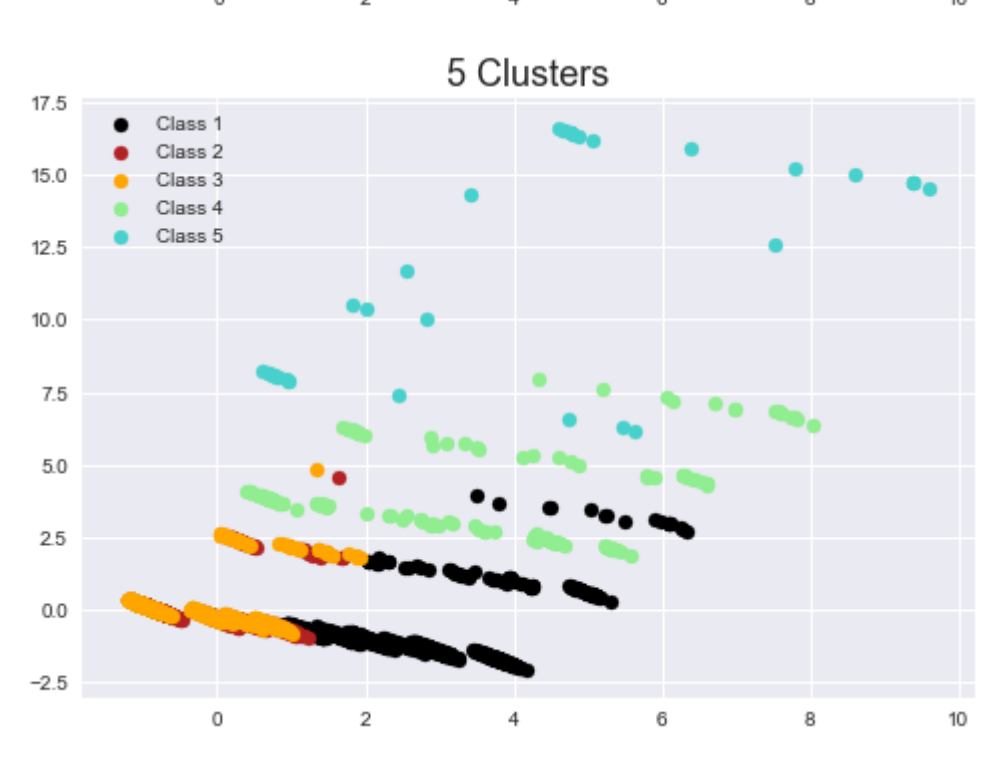


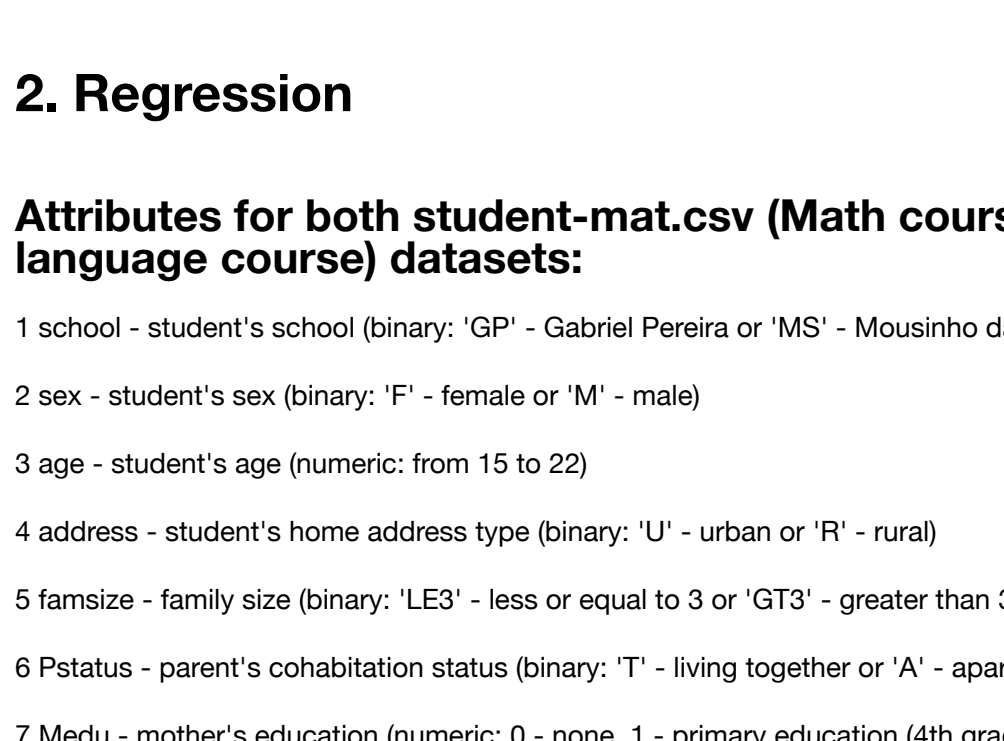


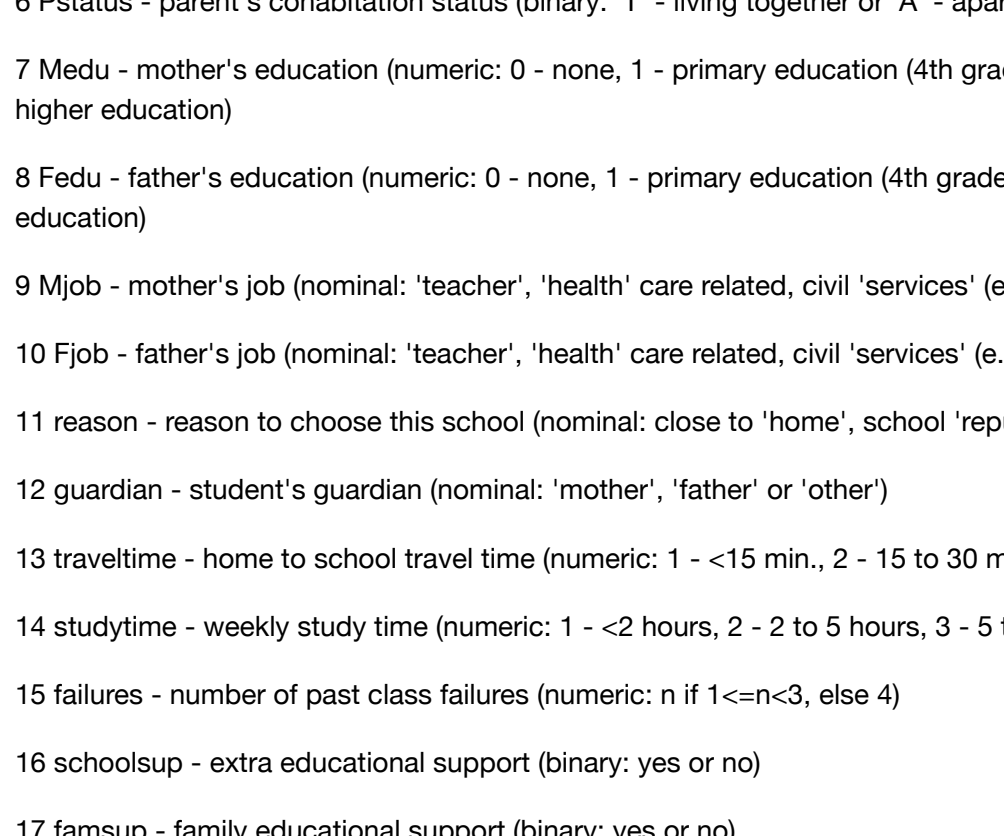
In [216]:

```python
show_clusts(df3)
```









2. Regression

Attributes for both student-mat.csv (Math course) and student-por.csv (Portuguese language course) datasets:

1 school - student's school (binary: 'GP' - Gabriel Pereira or 'MS' - Mousinho da Silveira)

2 sex - student's sex (binary: 'F' - female or 'M' - male)

3 age - student's age (numeric: from 15 to 22)

4 address - student's home address type (binary: 'U' - urban or 'R' - rural)

5 famsize - family size (binary: 'LE3' - less or equal to 3 or 'GT3' - greater than 3)

6 Pstatus - parent's cohabitation status (binary: 'T' - living together or 'A' - apart)

7 Medu - mother's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 -" secondary education or 4 - higher education)

8 Fedu - father's education (numeric: 0 - none, 1 - primary education (4th grade), 2 - 5th to 9th grade, 3 - secondary education or 4 - higher education)

9 Mjob - mother's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

10 Fjob - father's job (nominal: 'teacher', 'health' care related, civil 'services' (e.g. administrative or police), 'at_home' or 'other')

11 reason - reason to choose this school (nominal: close to 'home', school 'reputation', 'course' preference or 'other')

12 guardian - student's guardian (nominal: 'mother', 'father' or 'other')

13 traveltime - home to school travel time (numeric: 1 - <15 min., 2 - 15 to 30 min., 3 - 30 min. to 1 hour, or 4 - >1 hour)

14 studytime - weekly study time (numeric: 1 - <2 hours, 2 - 2 to 5 hours, 3 - 5 to 10 hours, or 4 - >10 hours)

15 failures - number of past class failures (numeric: n if 1<=n<3, else 4)

16 schoolsup - extra educational support (binary: yes or no)

17 famsup - family educational support (binary: yes or no)

18 paid - extra paid classes within the course subject (Math or Portuguese) (binary: yes or no)

19 activities - extra-curricular activities (binary: yes or no)

20 nursery - attended nursery school (binary: yes or no)

21 higher - wants to take higher education (binary: yes or no)

22 internet - Internet access at home (binary: yes or no)

23 romantic - with a romantic relationship (binary: yes or no)

24 famrel - quality of family relationships (numeric: from 1 - very bad to 5 - excellent)

25 freetime - free time after school (numeric: from 1 - very low to 5 - very high)

26 goout - going out with friends (numeric: from 1 - very low to 5 - very high)

27 Dalc - workday alcohol consumption (numeric: from 1 - very low to 5 - very high)

28 Walc - weekend alcohol consumption (numeric: from 1 - very low to 5 - very high)

29 health - current health status (numeric: from 1 - very bad to 5 - very good)

30 absences - number of school absences (numeric: from 0 to 93)

these grades are related with the course subject, Math or Portuguese:

31 G1 - first period grade (numeric: from 0 to 20)

31 G2 - second period grade (numeric: from 0 to 20)

32 G3 - final grade (numeric: from 0 to 20, output target)

Preprocessing

In [651]:

```python
def prep(df, dummy=True):
    df.dropna(inplace=True)

    # Binary to 0/1
    df['school'] = np.where(df['school'] == 'GP', 0, 1)
    df['sex'] = np.where(df['sex'] == 'F', 0, 1)
    df['address'] = np.where(df['address'] == 'U', 1, 0)
    df['famsize'] = np.where(df['famsize'] == 'LE3', 0, 1)
    df['Pstatus'] = np.where(df['Pstatus'] == 'T', 1, 0)

    # Binary with yes/no to 0/1
    features = ['schoolsup', 'famsup', 'paid', 'activities',
                'nursery', 'higher', 'internet', 'romantic']
    for f in features:
        df[f] = np.where(df[f] == 'yes', 1, 0)

    if dummy:
        # Dummy vars for nominal features
        features = ['Mjob', 'Fjob', 'reason', 'guardian']
        mf = pd.get_dummies(df[features])
        df.drop(features, axis=1, inplace=True)
        df = df.join(mf)

    return df
```

In [652]:

```python
init_df_mat = pd.read_csv('student-mat.csv', sep=';')
init_df_por = pd.read_csv('student-por.csv', sep=';')
```

In [683]:

```python
# Read and preprocess data1
df_mat = pd.read_csv('student-mat.csv', sep=';')
df_mat = prep(df_mat, dummy=True)

# Read and preprocess data2
df_por = pd.read_csv('student-por.csv', sep=';')
df_por = prep(df_por, dummy=True)

df_mat.head()
```

Out[683]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	traveltime	studytime	...	Fjob_other	Fjob_services	Fjob_teacher	reason_
0	0	0	18	1	1	0	4	4	2	2	...	0	0	1	
1	0	0	17	1	1	1	1	1	1	2	...	1	0	0	
2	0	0	15	1	0	1	1	1	1	2	...	1	0	0	
3	0	0	15	1	1	1	4	2	1	3	...	0	1	0	
4	0	0	16	1	1	1	3	3	1	2	...	1	0	0	

5 rows × 46 columns

In [684]:
```
df_por.head()
```

Out[684]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	traveltime	studytime	...	Fjob_other	Fjob_services	Fjob_teacher	reason_
0	0	0	18	1	1	0	4	4	2	2	...	0	0	1	
1	0	0	17	1	1	1	1	1	1	2	...	1	0	0	
2	0	0	15	1	0	1	1	1	1	2	...	1	0	0	
3	0	0	15	1	1	1	4	2	1	3	...	0	1	0	
4	0	0	16	1	1	1	3	3	1	2	...	1	0	0	

5 rows × 46 columns

Models

OLS

In [685]:
```
import statsmodels.api as sm
import statsmodels.formula.api as smf
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error as mse
```

In [686]:
```
df = df_por.copy(deep=True)
# df['const'] = np.ones(len(df))
df.corr().style.background_gradient(cmap='coolwarm').format('{0:,.1f}')
```

Out[686]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	traveltime	studytime	failures	schoolsup	famsup	paid
school	1.0	-0.1	0.1	-0.4	-0.0	0.0	-0.3	-0.2	0.3	-0.1	0.1	-0.1	-0.1	-0.0
sex	-0.1	1.0	-0.0	0.0	-0.1	0.1	0.1	0.1	0.0	-0.2	0.1	-0.1	-0.1	0.1
age	0.1	-0.0	1.0	-0.0	0.0	-0.0	-0.1	-0.1	0.0	-0.0	0.3	-0.2	-0.1	-0.0
address	-0.4	0.0	-0.0	1.0	-0.0	-0.1	0.2	0.1	-0.3	0.1	-0.1	0.0	0.0	-0.0
famsize	-0.0	-0.1	0.0	-0.0	1.0	0.2	0.0	0.0	-0.0	0.0	0.1	0.1	0.0	0.1
Pstatus	0.0	0.1	-0.0	-0.1	0.2	1.0	-0.1	-0.0	0.0	-0.0	-0.0	-0.0	0.0	0.0
Medu	-0.3	0.1	-0.1	0.2	0.0	-0.1	1.0	0.6	-0.3	0.1	-0.2	-0.0	0.1	0.1
Fedu	-0.2	0.1	-0.1	0.1	0.0	-0.0	0.6	1.0	-0.2	0.1	-0.2	0.0	0.1	0.1
traveltime	0.3	0.0	0.0	-0.3	-0.0	0.0	-0.3	-0.2	1.0	-0.1	0.1	-0.0	-0.0	-0.0
studytime	-0.1	-0.2	-0.0	0.1	0.0	-0.0	0.1	0.1	-0.1	1.0	-0.1	0.1	0.1	-0.0
failures	0.1	0.1	0.3	-0.1	0.1	-0.0	-0.2	-0.2	0.1	-0.1	1.0	-0.0	-0.0	0.1
schoolsup	-0.1	-0.1	-0.2	0.0	0.1	-0.0	-0.0	0.0	-0.0	0.1	-0.0	1.0	0.1	0.0
famsup	-0.1	-0.1	-0.1	0.0	0.0	0.0	0.1	0.1	-0.0	0.1	-0.0	0.1	1.0	0.1
paid	-0.0	0.1	-0.0	-0.0	0.1	0.0	0.1	0.1	-0.0	-0.0	0.1	0.0	0.1	1.0
activities	-0.1	0.1	-0.1	-0.0	0.0	0.1	0.1	0.1	-0.0	0.1	0.0	-0.0	-0.0	0.1
nursery	0.0	-0.0	-0.0	0.0	-0.1	-0.0	0.1	0.1	-0.0	0.0	-0.1	0.0	0.0	0.0
higher	-0.1	-0.1	-0.3	0.1	-0.0	0.0	0.2	0.2	-0.1	0.2	-0.3	0.1	0.1	0.0
internet	-0.2	0.1	0.0	0.2	-0.0	0.1	0.3	0.2	-0.2	0.0	-0.1	-0.0	0.1	0.0
romantic	0.1	-0.1	0.2	-0.0	0.0	-0.1	-0.0	-0.1	0.0	0.0	0.1	-0.1	-0.0	-0.0
famrel	-0.0	0.1	-0.0	-0.0	-0.0	0.1	0.0	0.0	-0.0	-0.0	-0.1	-0.0	0.0	0.0
freetime	0.0	0.1	-0.0	-0.0	0.0	0.0	-0.0	0.0	0.0	-0.1	0.1	-0.0	0.0	-0.0
goout	0.0	0.1	0.1	0.0	0.0	0.0	0.0	0.0	0.1	-0.1	0.0	-0.1	0.0	-0.0
Dalc	0.0	0.3	0.1	-0.0	-0.1	0.0	-0.0	0.0	0.1	-0.1	0.1	-0.0	-0.0	0.1
Walc	0.0	0.3	0.1	-0.0	-0.1	0.1	-0.0	0.0	0.1	-0.2	0.1	-0.1	-0.1	0.0
health	-0.1	0.1	-0.0	0.0	-0.0	0.0	0.0	0.0	-0.0	-0.1	0.0	0.0	0.0	0.1
absences	-0.2	0.0	0.1	0.1	-0.0	-0.1	-0.0	0.0	-0.0	-0.1	0.1	-0.1	0.0	-0.0
G1	-0.3	-0.1	-0.2	0.2	-0.0	0.0	0.3	0.2	-0.2	0.3	-0.4	-0.1	0.0	-0.1
G2	-0.3	-0.1	-0.1	0.2	-0.0	0.0	0.3	0.2	-0.2	0.2	-0.4	-0.1	0.0	-0.0
G3	-0.3	-0.1	-0.1	0.2	-0.0	-0.0	0.2	0.2	-0.1	0.2	-0.4	-0.1	0.1	-0.1
Mjob_at_home	0.2	-0.1	0.1	-0.2	-0.0	0.0	-0.4	-0.2	0.2	-0.0	0.1	-0.0	-0.0	-0.0
Mjob_health	-0.1	0.0	-0.1	0.1	-0.0	0.0	0.3	0.1	-0.1	-0.0	-0.0	-0.1	0.0	0.0
Mjob_other	0.0	-0.0	0.0	-0.0	0.1	-0.0	-0.2	-0.2	0.0	-0.0	-0.0	0.0	-0.1	-0.0
Mjob_services	-0.1	0.1	-0.0	0.1	-0.0	-0.0	0.1	0.1	-0.1	0.0	0.0	0.0	0.1	0.0
Mjob_teacher	-0.1	0.1	-0.0	0.1	-0.0	0.0	0.4	0.3	-0.1	0.0	-0.1	-0.1	0.0	-0.0
Fjob_at_home	0.1	-0.1	0.1	0.0	-0.0	-0.0	-0.1	-0.1	-0.0	-0.0	0.0	0.0	0.1	0.0
Fjob_health	-0.1	-0.0	-0.1	0.1	-0.0	-0.0	0.2	0.2	-0.1	0.1	-0.1	0.1	0.1	-0.0
Fjob_other	-0.1	-0.0	0.1	-0.1	-0.0	-0.1	-0.1	-0.2	0.1	-0.0	0.0	-0.0	-0.1	0.0
Fjob_services	0.1	0.0	-0.0	0.0	-0.0	0.1	-0.0	0.0	-0.0	0.0	-0.0	-0.1	-0.0	0.0
Fjob_teacher	-0.1	0.0	-0.1	0.0	0.1	-0.0	0.3	0.3	-0.0	-0.0	-0.1	0.1	0.0	-0.0
reason_course	0.1	-0.0	0.0	-0.1	0.0	0.0	-0.1	-0.1	0.1	-0.1	0.2	-0.1	-0.0	-0.0
reason_home	-0.1	0.1	-0.0	0.2	-0.0	-0.0	0.0	0.0	-0.1	-0.0	-0.1	0.1	0.0	0.1
reason_other	0.2	0.0	-0.0	-0.1	0.0	0.0	-0.0	-0.0	0.1	-0.1	0.0	0.0	-0.1	0.0
reason_reputation	-0.2	-0.0	-0.0	-0.0	-0.0	-0.0	0.1	0.1	-0.1	0.2	-0.1	0.0	0.0	-0.1
guardian_father	0.1	0.0	-0.1	0.0	0.0	0.1	-0.0	0.1	0.0	0.0	-0.1	0.0	0.0	-0.1
guardian_mother	-0.1	-0.0	-0.0	0.0	-0.0	-0.1	0.1	-0.0	-0.1	-0.0	-0.1	-0.0	-0.0	0.0
guardian_other	-0.0	-0.0	0.3	-0.0	0.0	-0.1	-0.1	-0.1	0.1	0.0	0.2	-0.0	0.1	0.0

In [687]:
```
# all columns except from G3
pred_cols_all = [i for i in df.columns if i != 'G3']

# all columns except from G1, G2, G3
pred_cols = [i for i in df.columns if i not in ['G1', 'G2', 'G3']]

# y/X
y = df['G3']
X = df[pred_cols]
X_all = df[pred_cols_all]

# train/test split
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=1)

x_train_all, x_test_all, y_train_all, y_test_all = train_test_split(
    X_all, y, test_size=0.25, random_state=1)
```

Model without G1 and G2

In [688]:
```
mod = sm.OLS(y_train, x_train).fit()
mod.summary()
```

Out[688]:

OLS Regression Results

Dep. Variable:	G3	R-squared:	0.402
Model:	OLS	Adj. R-squared:	0.350
Method:	Least Squares	F-statistic:	7.686
Date:	Thu, 01 Apr 2021	Prob (F-statistic):	1.95e-30
Time:	19:46:39	Log-Likelihood:	-1121.8
No. Observations:	486	AIC:	2324.
Df Residuals:	446	BIC:	2491.
Df Model:	39		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
school	-1.1764	0.304	-3.867	0.000	-1.774	-0.579
sex	-0.5898	0.284	-2.081	0.038	-1.147	-0.033
age	0.1281	0.117	1.093	0.275	-0.102	0.359
address	0.3605	0.302	1.195	0.233	-0.232	0.953
famsize	-0.4531	0.275	-1.650	0.100	-0.993	0.087
Pstatus	0.1327	0.379	0.350	0.726	-0.612	0.878
Medu	-0.1052	0.169	-0.622	0.534	-0.438	0.227
Fedu	0.2499	0.155	1.612	0.108	-0.055	0.555
traveltime	0.1808	0.182	0.993	0.321	-0.177	0.539
studytime	0.3780	0.154	2.462	0.014	0.076	0.680
failures	-1.3380	0.216	-6.193	0.000	-1.763	-0.913
schoolsup	-1.3227	0.395	-3.352	0.001	-2.098	-0.547
famsup	0.0912	0.255	0.357	0.721	-0.410	0.593
paid	-0.6197	0.521	-1.188	0.235	-1.645	0.405
activities	0.4781	0.248	1.928	0.055	-0.009	0.966
nursery	0.0308	0.302	0.102	0.919	-0.563	0.625
higher	1.8337	0.414	4.427	0.000	1.020	2.648
internet	0.5399	0.306	1.766	0.078	-0.061	1.141
romantic	-0.4005	0.256	-1.564	0.118	-0.904	0.103
famrel	0.2394	0.131	1.830	0.068	-0.018	0.496
freetime	-0.1631	0.124	-1.312	0.190	-0.407	0.081
goout	-0.0926	0.120	-0.771	0.441	-0.329	0.144
Dalc	-0.0588	0.165	-0.356	0.722	-0.383	0.265
Walc	-0.1649	0.131	-1.256	0.210	-0.423	0.093
health	-0.1549	0.086	-1.809	0.071	-0.323	0.013
absences	-0.0301	0.027	-1.101	0.272	-0.084	0.024
Mjob_at_home	1.4322	0.551	2.600	0.010	0.350	2.515
Mjob_health	1.9948	0.599	3.329	0.001	0.817	3.172
Mjob_other	1.5642	0.507	3.087	0.002	0.569	2.560
Mjob_services	1.6755	0.546	3.066	0.002	0.602	2.749
Mjob_teacher	1.9190	0.611	3.143	0.002	0.719	3.119
Fjob_at_home	2.0185	0.642	3.145	0.002	0.757	3.280
Fjob_health	1.3289	0.686	1.937	0.053	-0.020	2.678
Fjob_other	1.7755	0.512	3.471	0.001	0.770	2.781
Fjob_services	1.3929	0.519	2.684	0.008	0.373	2.413
Fjob_teacher	2.0698	0.683	3.032	0.003	0.728	3.411
reason_course	2.0593	0.598	3.446	0.001	0.885	3.234
reason_home	2.2429	0.620	3.618	0.000	1.025	3.461
reason_other	1.5938	0.649	2.455	0.014	0.318	2.870
reason_reputation	2.6897	0.645	4.169	0.000	1.422	3.958
guardian_father	2.7906	0.768	3.634	0.000	1.281	4.300
guardian_mother	2.7978	0.752	3.723	0.000	1.321	4.275
guardian_other	2.9974	0.951	3.153	0.002	1.129	4.866

Omnibus:	77.462	Durbin-Watson:	2.107
Prob(Omnibus):	0.000	Jarque-Bera (JB):	230.745
Skew:	-0.748	Prob(JB):	7.84e-51
Kurtosis:	6.026	Cond. No.	1.22e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.23e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [689]:
```
y_pred = mod.predict(x_test)
print(f'MSE: {mse(y_pred, y_test)}')
```

```
MSE: 9.529890246086358
```

Model with G1 and G2

In [690]:
```
mod_all = sm.OLS(y_train_all, x_train_all).fit()
mod_all.summary()
```

Out[690]:

OLS Regression Results

Dep. Variable:	G3	R-squared:	0.859
Model:	OLS	Adj. R-squared:	0.845
Method:	Least Squares	F-statistic:	65.72
Date:	Thu, 01 Apr 2021	Prob (F-statistic):	3.67e-162
Time:	19:46:55	Log-Likelihood:	-771.46
No. Observations:	486	AIC:	1627.
Df Residuals:	444	BIC:	1803.
Df Model:	41		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
school	-0.2543	0.152	-1.673	0.095	-0.553	0.044
sex	-0.1100	0.139	-0.792	0.429	-0.383	0.163
age	0.0069	0.058	0.120	0.905	-0.106	0.120
address	0.1373	0.147	0.933	0.352	-0.152	0.427
famsize	-0.0580	0.135	-0.431	0.667	-0.322	0.207
Pstatus	-0.0778	0.185	-0.421	0.674	-0.441	0.285
Medu	-0.1303	0.083	-1.578	0.115	-0.292	0.032
Fedu	0.0846	0.076	1.118	0.264	-0.064	0.233
traveltime	0.1021	0.089	1.150	0.251	-0.072	0.277
studytime	0.0654	0.076	0.864	0.388	-0.083	0.214
failures	-0.2051	0.110	-1.870	0.062	-0.421	0.010
schoolsup	-0.2280	0.195	-1.167	0.244	-0.612	0.156
famsup	0.1147	0.124	0.922	0.357	-0.130	0.359
paid	-0.3151	0.256	-1.231	0.219	-0.818	0.188
activities	0.0677	0.121	0.557	0.577	-0.171	0.306
nursery	-0.0274	0.147	-0.186	0.853	-0.317	0.262
higher	0.3829	0.206	1.856	0.064	-0.023	0.788
internet	0.1543	0.149	1.033	0.302	-0.139	0.448
romantic	-0.0491	0.125	-0.392	0.695	-0.295	0.197
famrel	-0.0543	0.065	-0.840	0.402	-0.181	0.073
freetime	-0.0547	0.061	-0.900	0.369	-0.174	0.065
goout	-0.0413	0.059	-0.703	0.482	-0.157	0.074
Dalc	0.0480	0.081	0.596	0.551	-0.110	0.206
Walc	-0.0587	0.064	-0.917	0.360	-0.185	0.067
health	-0.0524	0.042	-1.251	0.212	-0.135	0.030
absences	0.0124	0.013	0.923	0.356	-0.014	0.039
G1	0.1255	0.047	2.654	0.008	0.033	0.218
G2	0.8591	0.043	19.852	0.000	0.774	0.944
Mjob_at_home	0.0644	0.276	0.234	0.815	-0.477	0.606
Mjob_health	0.3355	0.297	1.129	0.260	-0.249	0.920
Mjob_other	0.0045	0.255	0.018	0.986	-0.496	0.505
Mjob_services	0.0679	0.273	0.249	0.804	-0.469	0.604
Mjob_teacher	0.2084	0.304	0.686	0.493	-0.388	0.805
Fjob_at_home	0.5388	0.317	1.700	0.090	-0.084	1.162
Fjob_health	-0.0467	0.339	-0.138	0.890	-0.712	0.619
Fjob_other	0.2034	0.256	0.794	0.427	-0.300	0.707
Fjob_services	0.1763	0.259	0.680	0.497	-0.333	0.685
Fjob_teacher	-0.1910	0.343	-0.557	0.578	-0.865	0.483
reason_course	0.3906	0.298	1.311	0.191	-0.195	0.976
reason_home	0.1955	0.312	0.626	0.531	-0.418	0.809
reason_other	-0.1890	0.325	-0.581	0.561	-0.828	0.450
reason_reputation	0.2835	0.325	0.872	0.384	-0.355	0.923
guardian_father	0.2204	0.388	0.568	0.570	-0.542	0.982
guardian_mother	0.1802	0.380	0.475	0.635	-0.566	0.926
guardian_other	0.2802	0.474	0.591	0.555	-0.651	1.211

Omnibus:	304.384	Durbin-Watson:	1.889
Prob(Omnibus):	0.000	Jarque-Bera (JB):	5495.624
Skew:	-2.372	Prob(JB):	0.00
Kurtosis:	18.776	Cond. No.	1.20e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 2.17e-27. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [691]:
```
y_pred_all = mod_all.predict(x_test_all)
print(f'MSE: {mse(y_pred_all, y_test_all)}')
```

```
MSE: 1.7672917612630112
```

Modified DF guess 1

There will not be G1 and G2 in the predictors from this point.

In [692]:
```
df.drop(['G1', 'G2'], axis=1, inplace=True)

# Take only predictors with low p-values
cols = [*mod.pvalues[mod.pvalues < 0.05].keys()]
mod1 = sm.OLS(y_train, x_train[cols]).fit()
mod1.summary()
```

Out[692]:

OLS Regression Results

Dep. Variable:	G3	R-squared:	0.357
Model:	OLS	Adj. R-squared:	0.331
Method:	Least Squares	F-statistic:	13.63
Date:	Thu, 01 Apr 2021	Prob (F-statistic):	4.30e-34
Time:	19:50:06	Log-Likelihood:	-1139.3
No. Observations:	486	AIC:	2319.
Df Residuals:	466	BIC:	2402.
Df Model:	19		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
school	-1.3195	0.277	-4.759	0.000	-1.864	-0.775
sex	-0.7345	0.254	-2.889	0.004	-1.234	-0.235
studytime	0.4248	0.151	2.806	0.005	0.127	0.722
failures	-1.4887	0.208	-7.146	0.000	-1.898	-1.079
schoolsup	-1.4765	0.385	-3.831	0.000	-2.234	-0.719
higher	1.9458	0.404	4.813	0.000	1.151	2.740
Mjob_at_home	2.3507	0.329	7.136	0.000	1.703	2.998
Mjob_health	2.8474	0.416	6.852	0.000	2.031	3.664
Mjob_other	2.4719	0.299	8.258	0.000	1.884	3.060
Mjob_services	2.6546	0.332	7.992	0.000	2.002	3.307
Mjob_teacher	3.0516	0.389	7.836	0.000	2.286	3.817
Fjob_at_home	0.5043	0.763	0.660	0.509	-0.996	2.005
Fjob_other	0.2798	0.619	0.452	0.652	-0.938	1.497
Fjob_services	-0.0090	0.631	-0.014	0.989	-1.248	1.230
Fjob_teacher	0.8185	0.783	1.046	0.296	-0.720	2.357
reason_course	3.2545	0.305	10.678	0.000	2.656	3.853
reason_home	3.4315	0.334	10.267	0.000	2.775	4.088
reason_other	2.6606	0.382	6.964	0.000	1.910	3.411
reason_reputation	4.0296	0.340	11.858	0.000	3.362	4.697
guardian_father	4.5409	0.391	11.624	0.000	3.773	5.309
guardian_mother	4.3327	0.366	11.838	0.000	3.614	5.052
guardian_other	4.5027	0.513	8.772	0.000	3.494	5.511

Omnibus:	71.319	Durbin-Watson:	2.114
Prob(Omnibus):	0.000	Jarque-Bera (JB):	191.446
Skew:	-0.721	Prob(JB):	2.68e-42
Kurtosis:	5.715	Cond. No.	1.72e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 1.1e-29. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [693]:
```
y_pred = mod1.predict(x_test[cols])
print(f'MSE: {mse(y_pred, y_test)}')
```

```
MSE: 9.347147341660435
```

Modified DF guess 2 - Correlation

In [699]:
```
# Choose by correlation
cor = df.corr()
target = abs(cor['G3'])
relevant = [i for i in [*target[target > 0.2].keys()] if i not in ['G1', 'G2', 'G3']]
relevant
```

Out[699]:
```
['school', 'Medu', 'Fedu', 'studytime', 'failures', 'higher', 'Dalc']
```

In [700]:
```
mod2 = sm.OLS(y_train, x_train[relevant]).fit()
mod2.summary()
```

Out[700]:

OLS Regression Results

Dep. Variable:	G3	R-squared (uncentered):	0.927
Model:	OLS	Adj. R-squared (uncentered):	0.926
Method:	Least Squares	F-statistic:	873.2
Date:	Thu, 01 Apr 2021	Prob (F-statistic):	4.41e-268
Time:	19:51:07	Log-Likelihood:	-1270.7
No. Observations:	486	AIC:	2555.
Df Residuals:	479	BIC:	2585.
Df Model:	7		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
school	0.1970	0.317	0.622	0.535	-0.426	0.820
Medu	0.5543	0.175	3.165	0.002	0.210	0.898
Fedu	0.6172	0.179	3.445	0.001	0.265	0.969
studytime	1.6425	0.166	9.880	0.000	1.316	1.969
failures	-0.4514	0.254	-1.776	0.076	-0.951	0.048
higher	5.3008	0.443	11.977	0.000	4.431	6.170
Dalc	0.6050	0.149	4.072	0.000	0.313	0.897

Omnibus:	51.865	Durbin-Watson:	2.071
Prob(Omnibus):	0.000	Jarque-Bera (JB):	98.390
Skew:	-0.637	Prob(JB):	4.31e-22
Kurtosis:	4.798	Cond. No.	13.6

Notes:
[1] R^2 is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [701]:
```
y_pred = mod2.predict(x_test[relevant])
print(f'MSE: {mse(y_pred, y_test)}')
```

```
MSE: 12.420520053311673
```

In [702]:
```
df[relevant].corr()
```

Out[702]:

	school	Medu	Fedu	studytime	failures	higher	Dalc
school	1.000000	-0.254787	-0.209806	-0.137857	0.113788	-0.136112	0.047169
Medu	-0.254787	1.000000	0.647477	0.097006	-0.172210	0.213896	-0.007018
Fedu	-0.209806	0.647477	1.000000	0.050400	-0.165915	0.191735	0.000061
studytime	-0.137857	0.097006	0.050400	1.000000	-0.147441	0.188256	-0.137585
failures	0.113788	-0.172210	-0.165915	-0.147441	1.000000	-0.309400	0.105949
higher	-0.136112	0.213896	0.191735	0.188256	-0.309400	1.000000	-0.131663
Dalc	0.047169	-0.007018	0.000061	-0.137585	0.105949	-0.131663	1.000000

Backward Elimination

In [703]:
```
# df.drop('const', axis=1, inplace=True)

# Backward Elimination
cols = list(X.columns)
pmax = 1

while (len(cols) > 0):
    p = []
    X_1 = X[cols]
    X_1 = sm.add_constant(X_1)
    model = sm.OLS(y,X_1).fit()
    p = pd.Series(model.pvalues.values[1:],index = cols)
    pmax = max(p)
    feature_with_p_max = p.idxmax()

    if pmax > 0.05:
        cols.remove(feature_with_p_max)
    else:
        break

selected_features = cols
print(selected_features)
```

```
['school', 'sex', 'studytime', 'failures', 'schoolsup', 'higher', 'Dalc', 'health', 'Mjob_at_home',
'Mjob_health', 'Mjob_other', 'Mjob_services', 'Mjob_teacher', 'Fjob_teacher', 'reason_course', 'reaso
n_home', 'reason_other', 'reason_reputation', 'guardian_father', 'guardian_mother', 'guardian_other']
```

In [704]:
```
mod3c = sm.OLS(y_train, sm.add_constant(x_train[selected_features])).fit()
mod3c.summary()
```

Out[704]:

OLS Regression Results

Dep. Variable:	G3	R-squared:	0.364
Model:	OLS	Adj. R-squared:	0.339
Method:	Least Squares	F-statistic:	14.83
Date:	Thu, 01 Apr 2021	Prob (F-statistic):	1.17e-35
Time:	19:54:20	Log-Likelihood:	-1136.8
No. Observations:	486	AIC:	2312.
Df Residuals:	467	BIC:	2391.
Df Model:	18		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	6.4895	0.357	18.179	0.000	5.788	7.191
school	-1.3597	0.273	-4.981	0.000	-1.896	-0.823
sex	-0.5666	0.263	-2.158	0.031	-1.083	-0.051
studytime	0.4016	0.150	2.676	0.008	0.107	0.697
failures	-1.4497	0.207	-6.994	0.000	-1.857	-1.042
schoolsup	-1.4153	0.383	-3.698	0.000	-2.167	-0.663
higher	1.8940	0.401	4.724	0.000	1.106	2.682
Dalc	-0.2428	0.135	-1.798	0.073	-0.508	0.023
health	-0.1411	0.082	-1.713	0.087	-0.303	0.021
Mjob_at_home	0.9970	0.259	3.848	0.000	0.488	1.506
Mjob_health	1.4165	0.391	3.627	0.000	0.649	2.184
Mjob_other	1.1296	0.204	5.539	0.000	0.729	1.530
Mjob_services	1.3067	0.261	4.999	0.000	0.793	1.820
Mjob_teacher	1.6398	0.337	4.868	0.000	0.978	2.302
Fjob_teacher	0.6282	0.538	1.167	0.244	-0.429	1.686
reason_course	1.5646	0.199	7.843	0.000	1.173	1.957
reason_home	1.7320	0.236	7.324	0.000	1.267	2.197
reason_other	0.9638	0.311	3.100	0.002	0.353	1.575
reason_reputation	2.2292	0.248	8.992	0.000	1.742	2.716
guardian_father	2.1878	0.264	8.294	0.000	1.669	2.706
guardian_mother	1.9952	0.217	9.205	0.000	1.569	2.421
guardian_other	2.3066	0.391	5.896	0.000	1.538	3.075

Omnibus:	65.831	Durbin-Watson:	2.141
Prob(Omnibus):	0.000	Jarque-Bera (JB):	167.413
Skew:	-0.684	Prob(JB):	4.43e-37
Kurtosis:	5.529	Cond. No.	4.97e+16

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The smallest eigenvalue is 4.57e-30. This might indicate that there are
strong multicollinearity problems or that the design matrix is singular.

In [705]:
```
y_pred = mod3c.predict(sm.add_constant(x_test[selected_features]))
print(f'MSE: {mse(y_pred, y_test)}')
```

```
MSE: 9.074017869962821
```

Best number of features

In [706]:
```
import itertools


def findsubsets(s, n):
    return [set(i) for i in itertools.combinations(s, n)]

best_mse = 16.05
best_subs = X.columns
for n in range(2, 13):
    combs = findsubsets(selected_features, n)
    for preds in combs:
        mod = sm.OLS(y_train, sm.add_constant(x_train[preds])).fit()
        y_pred = mod.predict(sm.add_constant(x_test[preds]))
        m = mse(y_pred, y_test)

        if m < best_mse:
            best_mse = m
            best_subs = preds

print(best_mse)
print(best_subs)
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-706-39e57b598b20> in <module>
     10     combs = findsubsets(selected_features, n)
     11     for preds in combs:
---> 12         mod = sm.OLS(y_train, sm.add_constant(x_train[preds])).fit()
     13         y_pred = mod.predict(sm.add_constant(x_test[preds]))
     14         m = mse(y_pred, y_test)

/opt/anaconda3/lib/python3.7/site-packages/statsmodels/tools/tools.py in add_constant(data, prepend,
 has_constant)
    326             warnings.warn(recarray_warning, FutureWarning)
    327         from statsmodels.tsa.tsatools import add_trend
--> 328         return add_trend(data, trend='c', prepend=prepend, has_constant=has_constant)
    329
    330     # Special case for NumPy

/opt/anaconda3/lib/python3.7/site-packages/statsmodels/tsa/tsatools.py in add_trend(x, trend, prepen
d, has_constant)
    108                 except:
    109                     return False
--> 110             col_const = x.apply(safe_is_const, 0)
    111         else:
    112             ptp0 = np.ptp(np.asanyarray(x), axis=0)

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/frame.py in apply(self, func, axis, raw, resul
t_type, args, **kwds)
   7763             kwds=kwds,
   7764         )
-> 7765         return op.get_result()
   7766
   7767     def applymap(self, func, na_action: Optional[str] = None) -> DataFrame:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/apply.py in get_result(self)
    183             return self.apply_raw()
    184
--> 185         return self.apply_standard()
    186
    187     def apply_empty_result(self):

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/apply.py in apply_standard(self)
    277
    278         # wrap results
--> 279         return self.wrap_results(results, res_index)
    280
    281     def apply_series_generator(self) -> Tuple[ResType, "Index"]:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/apply.py in wrap_results(self, results, res_in
dex)
    311         if constructor_sliced is Series:
    312             result = create_series_with_explicit_dtype(
--> 313                 results, dtype_if_empty=np.float64
    314             )
    315         else:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/construction.py in create_series_with_explicit
_dtype(data, index, dtype, name, copy, fastpath, dtype_if_empty)
    665         dtype = dtype_if_empty
    666     return Series(
--> 667         data=data, index=index, dtype=dtype, name=name, copy=copy, fastpath=fastpath
    668     )

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/series.py in __init__(self, data, index, dtyp
e, name, copy, fastpath)
    287                 data = data._mgr
    288             elif is_dict_like(data):
--> 289                 data, index = self._init_dict(data, index, dtype)
    290                 dtype = None
    291                 copy = False

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/series.py in _init_dict(self, data, index, dty
pe)
    381         # TODO: passing np.float64 to not break anything yet. See GH-17261
    382         s = create_series_with_explicit_dtype(
--> 383             values, index=keys, dtype=dtype, dtype_if_empty=np.float64
    384         )
    385

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/construction.py in create_series_with_explicit
_dtype(data, index, dtype, name, copy, fastpath, dtype_if_empty)
    665         dtype = dtype_if_empty
    666     return Series(
--> 667         data=data, index=index, dtype=dtype, name=name, copy=copy, fastpath=fastpath
    668     )

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/series.py in __init__(self, data, index, dtyp
e, name, copy, fastpath)
    250
    251             if index is not None:
--> 252                 index = ensure_index(index)
    253
    254             if data is None:

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in ensure_index(index_like, co
py)
   5915             index_like = copy_func(index_like)
   5916
-> 5917     return Index(index_like)
   5918
   5919

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/indexes/base.py in __new__(cls, data, dtype, c
opy, name, tupleize_cols, **kwargs)
    388                 )
    389             # other iterable of some kind
--> 390             subarr = com.asarray_tuplesafe(data, dtype=object)
    391             return Index(subarr, dtype=dtype, copy=copy, name=name, **kwargs)
    392

/opt/anaconda3/lib/python3.7/site-packages/pandas/core/common.py in asarray_tuplesafe(values, dtype)
    206         return construct_1d_object_array_from_listlike(values)
    207
--> 208     result = np.asarray(values, dtype=dtype)
    209
    210     if issubclass(result.dtype.type, str):

/opt/anaconda3/lib/python3.7/site-packages/numpy/core/_asarray.py in asarray(a, dtype, order, like)
    100         (Fortran-style) memory representation.  Defaults to 'C'.
    101
--> 102     Returns
    103     -------
    104     out : ndarray or an ndarray subclass

KeyboardInterrupt:
```

In [ ]:
```
mod = sm.OLS(y_train, sm.add_constant(x_train[best_subs])).fit()
mod.summary()
```

In [ ]:
```
y_pred = mod.predict(sm.add_constant(x_test[best_subs]))
mse(y_pred, y_test)
```

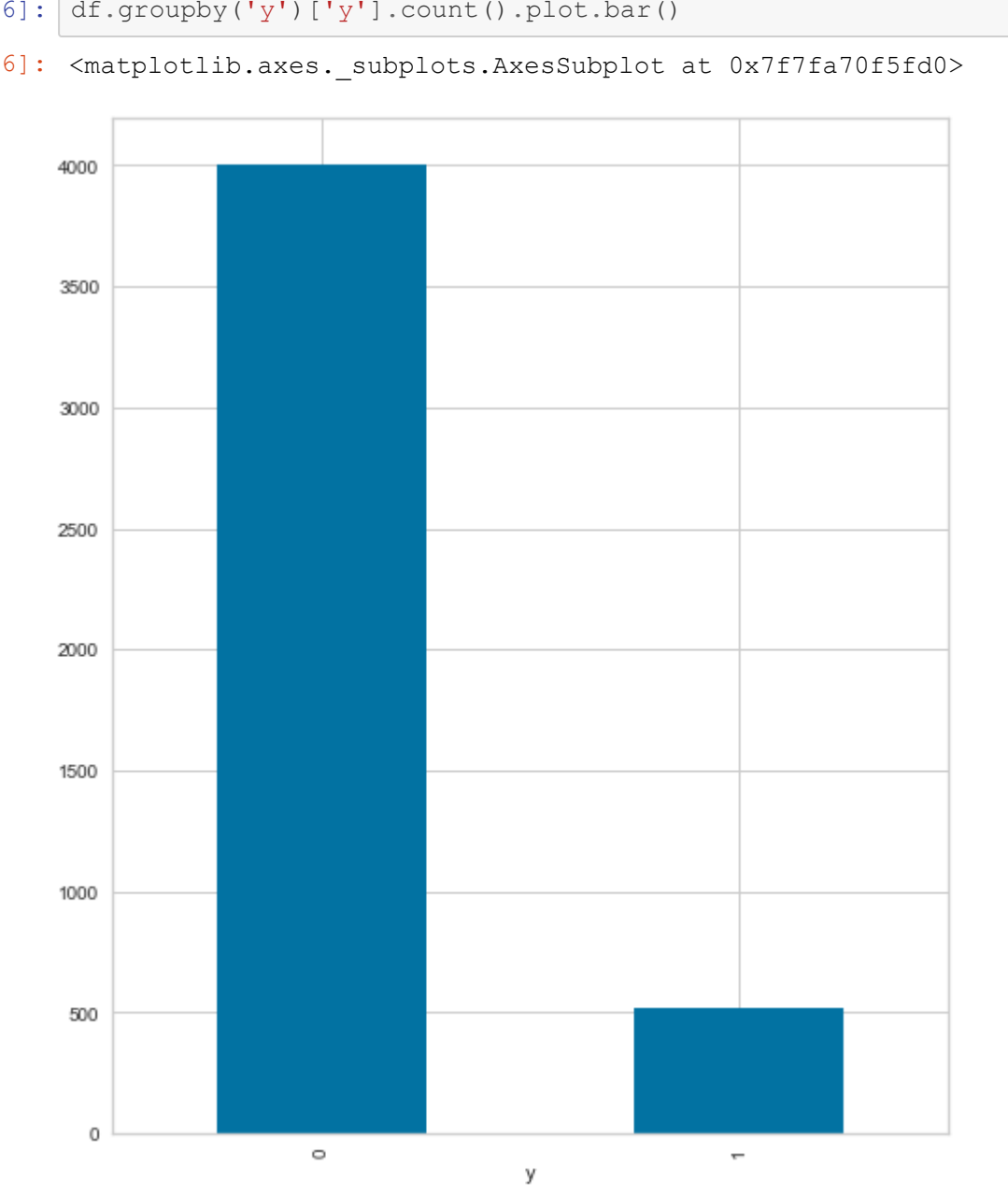
In [401]:
```
from sklearn.model_selection import KFold
from sklearn.utils import resample


def boot_test(df, mod, n_boots_=10, n_sam=100, cols=None):
    # prepare bootstrap sample

    if not cols:
        cols = [i for i in df.columns if i not in ['G1', 'G2', 'G3']]

    avg_mse = 0
    for n_boots in range(n_boots_):
        boot = resample(df, replace=True, n_samples=n_sam, random_state=1)
        y = boot['G3']
        y_pred = mod.predict(sm.add_constant(boot[cols]))
        avg_mse += mse(y_pred, y) / 10

    return avg_mse

print(boot_test(df, mod, cols=best_subs))
```

```
16.81219401618144
```

Combine Correlated Features

In [1016]:
```
df.groupby('y')['y'].count().plot.bar()
```

Out[1016]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7fa70f5fd0>



In [995]:
```
df.corr().style.background_gradient(cmap='coolwarm').format('{0:,.1f}')
```

```
/opt/anaconda3/lib/python3.7/site-packages/pandas/io/formats/style.py:1264: RuntimeWarning: All-NaN s
lice encountered
  smin = np.nanmin(s.to_numpy()) if vmin is None else vmin
/opt/anaconda3/lib/python3.7/site-packages/pandas/io/formats/style.py:1265: RuntimeWarning: All-NaN s
lice encountered
  smax = np.nanmax(s.to_numpy()) if vmax is None else vmax
```

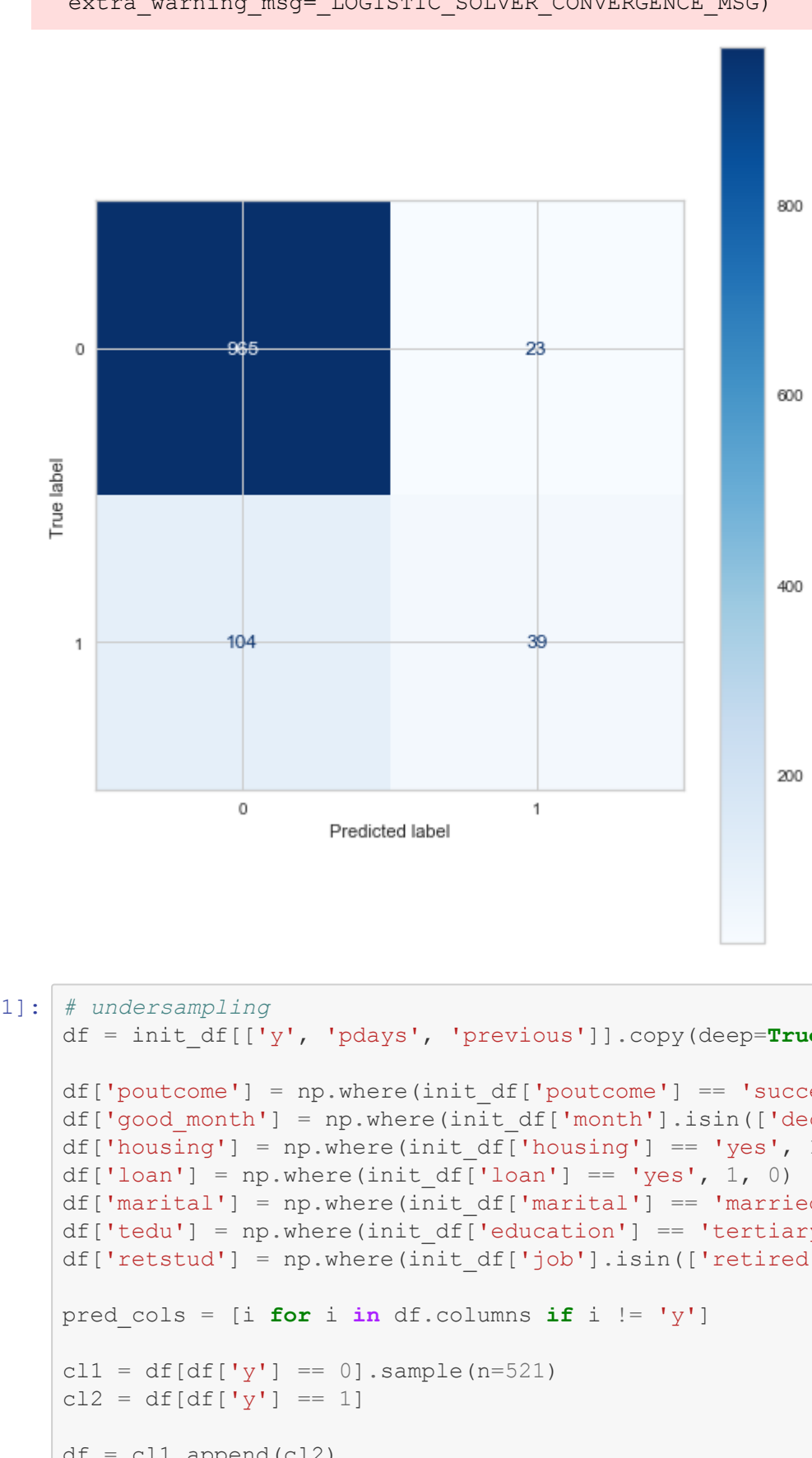
Out[995]:

	y	duration	pdays	previous	poutcome	good_month	housing	loan	marital	tedu	retstud
y	1.0	0.4	0.1	0.1	0.3	0.2	-0.1	-0.1	nan	0.1	0.1
duration	0.4	1.0	0.0	0.0	0.0	-0.0	0.0	-0.0	nan	-0.0	0.0
pdays	0.1	0.0	1.0	0.6	0.2	0.1	0.1	-0.0	nan	0.0	-0.0
previous	0.1	0.0	0.6	1.0	0.3	0.1	0.0	-0.0	nan	0.0	0.0
poutcome	0.3	0.0	0.2	0.3	1.0	0.2	-0.1	-0.1	nan	0.0	0.0
good_month	0.2	-0.0	0.1	0.1	0.2	1.0	-0.1	-0.1	nan	0.0	0.1
housing	-0.1	0.0	0.1	0.0	-0.1	-0.1	1.0	0.0	nan	-0.1	-0.2
loan	-0.1	-0.0	-0.0	-0.0	-0.1	-0.1	0.0	1.0	nan	-0.0	-0.0
marital	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan	nan
tedu	0.1	-0.0	0.0	0.0	0.0	0.0	-0.1	-0.0	nan	1.0	-0.1
retstud	0.1	0.0	-0.0	0.0	0.0	0.1	-0.2	-0.0	nan	-0.1	1.0

In [1047]:
```
pred_cols = [i for i in df.columns if i != 'y']
y = df['y']
X = df[pred_cols]

# train/test
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=1)


train_df = x_train.join(y_train)
test_df = x_test.join(y_test)

mod = LogisticRegression().fit(x_train, y_train)
plot_confusion_matrix(mod, x_test, y_test, cmap=plt.cm.Blues);

y_pred = mod.predict(x_test)
print('ROC: ', roc_auc_score(y_pred, y_test))
print('F1: ', f1_score(y_test, y_pred, zero_division=1))
```

```
ROC:  0.7658725368900692
F1:  0.3804878048780488
```

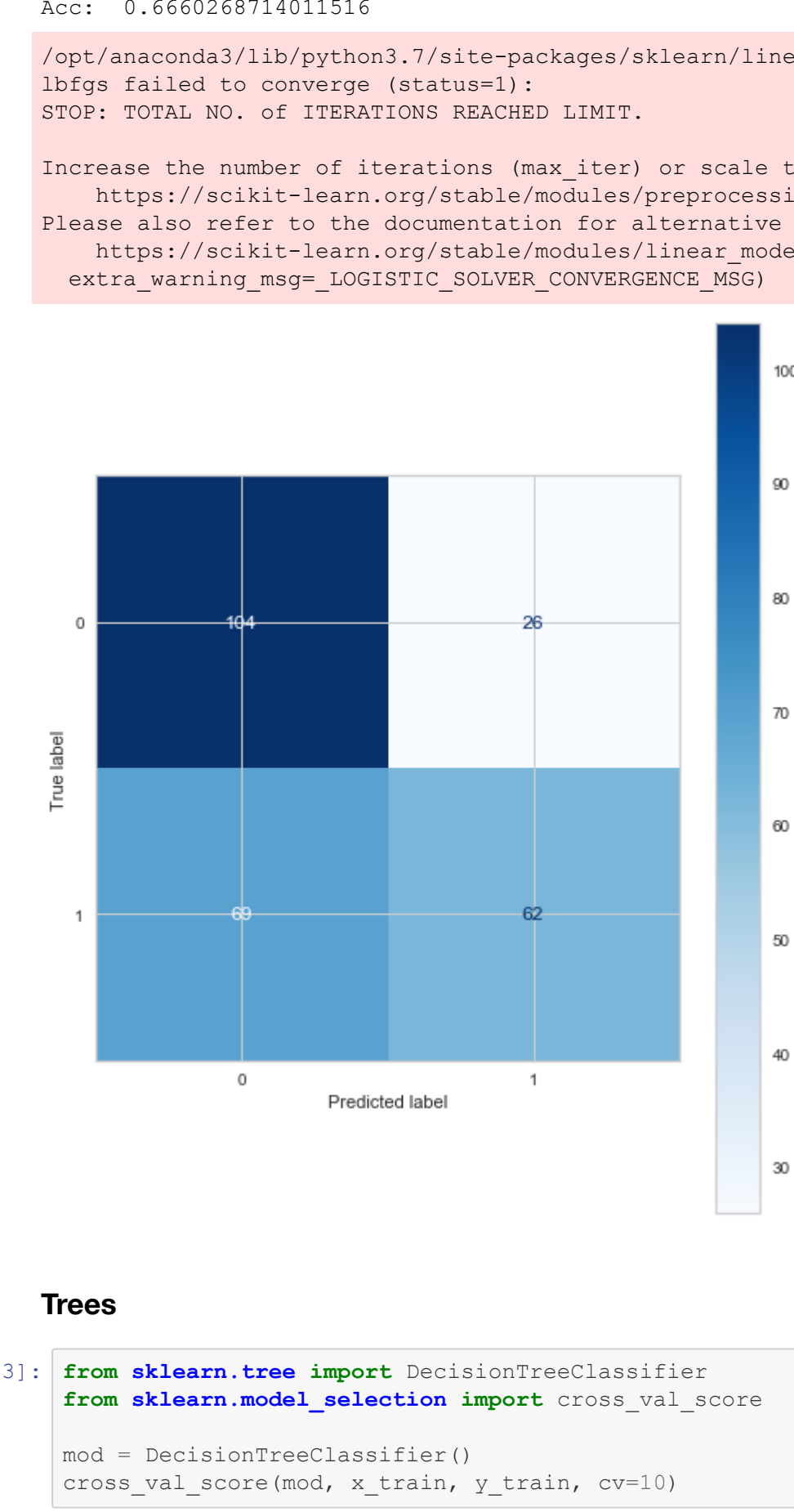
```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```



In [1051]:
```
# undersampling
df = init_df[['y', 'pdays', 'previous']].copy(deep=True)

df['poutcome'] = np.where(init_df['poutcome'] == 'success', 1, 0)
df['good_month'] = np.where(init_df['month'].isin(['dec', 'mar', 'oct', 'sep']), 1, 0)
df['housing'] = np.where(init_df['housing'] == 'yes', 1, 0)
df['loan'] = np.where(init_df['loan'] == 'yes', 1, 0)
df['marital'] = np.where(init_df['marital'] == 'married', 1, 0)
df['tedu'] = np.where(init_df['education'] == 'tertiary', 1, 0)
df['retstud'] = np.where(init_df['job'].isin(['retired', 'student']), 1, 0)

pred_cols = [i for i in df.columns if i != 'y']

cl1 = df[df['y'] == 0].sample(n=521)
cl2 = df[df['y'] == 1]

df = cl1.append(cl2)
y = df['y']
X = df[pred_cols]

# train/test
x_train, x_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=1)


train_df = x_train.join(y_train)
test_df = x_test.join(y_test)

mod = LogisticRegression().fit(x_train, y_train)
plot_confusion_matrix(mod, x_test, y_test, cmap=plt.cm.Blues);

y_pred = mod.predict(X)

print('ROC: ', roc_auc_score(y_pred, y))
print('F1: ', f1_score(y, y_pred, zero_division=1))
print('Acc: ', accuracy_score(y_pred, y))
```

```
ROC:  0.6810328551745193
F1:  0.6098654708520179
Acc:  0.6660268714011516
```

```
/opt/anaconda3/lib/python3.7/site-packages/sklearn/linear_model/_logistic.py:765: ConvergenceWarning:
lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  extra_warning_msg=_LOGISTIC_SOLVER_CONVERGENCE_MSG)
```



Trees

In [1043]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import cross_val_score

mod = DecisionTreeClassifier()
cross_val_score(mod, x_train, y_train, cv=10)
```

Out[1043]:
```
array([0.75949367, 0.75641026, 0.73076923, 0.73076923, 0.74358974,
       0.71794872, 0.82051282, 0.65384615, 0.69230769, 0.74358974])
```

In [ ]:
```
mod
```

In [ ]:

In [1041]:
```
from catboost import CatBoostClassifier
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-1041-a552d8edcebd> in <module>
----> 1 from catboost import CatBoostClassifier

ModuleNotFoundError: No module named 'catboost'
```

In [ ]: