FIT1043 Assignment2

student ID:32068611

student name: Lianzheng Xie

Task A: Data Wrangling and Analysis on ARD Dataset

A1. Dataset size

How many rows and columns exist in this dataset?

In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import sklearn
import seaborn as sns
from scipy.stats import linregress
from sklearn.linear_model import LinearRegression
data = pd.read_csv('Australian_Road_Deaths.csv',encoding='utf-8')
# read  'monthly_smartcard_replacements.csv'
#A1
data.shape
#.shape function returns the rows and columns of the data
print('The data has rows number:',data.shape[0])
print('The data has columns number:',data.shape[1])
```

```
The data has rows number: 9140
The data has columns number: 22
```

A2. The number of unique values in some columns

Count the number of unique values for National Remoteness Areas, SA4 Name 2016, National LGA Name 2017, and National Road Type in this dataset.

In [2]:
```
data[['National Remoteness Areas','SA4 Name 2016','National LGA Name 2017','National Road Type']].nunique()
```

Out[2]:
```
National Remoteness Areas      5
SA4 Name 2016                 88
National LGA Name 2017       500
National Road Type             9
dtype: int64
```

The number of unique values of National Remoteness Areas are 5,SA4 Name 2016 are 88, National LGA Name 2017 are 500, National Road Type are 9.

A3. Missing values and duplicates

There are some missing values: Unspecified, Undetermined, and blank (NaN) represent missing values.

1. How many rows contain missing values (Unspecified or Undetermined or blank) in this dataset?

In [3]:
```
data['YYYYMM']=pd.to_datetime(data['YYYYMM'],format='%Y%m') #to_datetime Convert the type of the "YYYYMM" column to date-time forma
data['month']=data['YYYYMM'].dt.month # dt.month get the month of date
data['year']=data['YYYYMM'].dt.year # dt.year get the year of date
```

In [4]:
```
data = data.replace('Unspecified',np.nan) # Use Replace to replace 'Unspecified'  with the null value 'NaN'
data = data.replace('Undetermined',np.nan)# Use Replace to replace 'Undetermined' with the null value 'NaN'
df = data.loc[data.isnull().T.any()]
# Isnull () determines whether an element in the data isnull; T is the transpose; Any () determines whether the row has a null valu
df
```

Out[4]:

	Crash ID	State	YYYYMM	Day of week	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	Road User	...	National Remoteness Areas	SA4 Name 2016	Na LGA
0	20212133	Vic	2021-09-01	Sunday	0:30:00	Single	NaN	NaN	NaN	Motorcycle rider	...	Inner Regional Australia	Melbourne - Outer East	Ran
1	20214022	SA	2021-09-01	Saturday	23:31:00	Multiple	No	No	No	Pedestrian	...	Major Cities of Australia	Adelaide - North	Playf
2	20212096	Vic	2021-09-01	Saturday	23:00:00	Single	NaN	NaN	NaN	Car passenger	...	Inner Regional Australia	Hume	Wan
3	20212145	Vic	2021-09-01	Saturday	22:25:00	Single	NaN	NaN	NaN	Car driver	...	Outer Regional Australia	Hume	Wan
4	20212075	Vic	2021-09-01	Saturday	5:15:00	Single	NaN	NaN	NaN	Motorcycle rider	...	Major Cities of Australia	Melbourne - South East	Ca
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9135	20142068	Vic	2014-01-01	Monday	18:20:00	Single	No	No	No	Car passenger	...	NaN	NaN	
9136	20141285	NSW	2014-01-01	Tuesday	20:50:00	Single	No	No	No	Car driver	...	NaN	NaN	
9137	20143125	Qld	2014-01-01	Friday	1:00:00	Single	No	No	No	Motorcycle pillion Car passenger	...	NaN	NaN	
9138	20143065	Qld	2014-01-01	Friday	10:00:00	Multiple	No	No	Yes	Car driver	...	NaN	NaN	
9139	20141099	NSW	2014-01-01	Wednesday	13:24:00	Single	No	No	No	Car driver	...	NaN	NaN	

2302 rows × 24 columns

There are 2302 rows contain missing values in this dataset

2. List the months with no missing values in them.

In [5]:
```
df['month'].unique() # The month is obtained according to the table containing missing value
```

Out[5]:
```
array([ 9,  8,  7,  6,  5,  4,  3,  2,  1, 12, 11, 10], dtype=int64)
```

There are missing values in the data every month

3. Remove the records with missing values.

In [6]:
```
# data.dropna(inplace = True)
data = data.dropna(how ='any',axis =0)
# How ='any 'to delete rows (columns) that contain missing values;Axis =0 or axis='index 'deletes rows with missing values
data
```

Out[6]:

	Crash ID	State	YYYYMM	Day of week	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	Road User	...	National Remoteness Areas	SA4 Name 2016	Nati N
5	20213034	Qld	2021-09-01	Saturday	4:00:00	Multiple	No	No	No	Motorcycle rider	...	Major Cities of Australia	Brisbane - South	Bris
8	20213026	Qld	2021-09-01	Wednesday	23:00:00	Multiple	No	No	No	Car passenger	...	Major Cities of Australia	Ipswich	Ips
9	20213092	Qld	2021-09-01	Saturday	2:00:00	Single	No	No	No	Car driver	...	Major Cities of Australia	Logan - Beaudesert	Loga
10	20214053	SA	2021-09-01	Thursday	21:00:00	Single	No	No	No	Car driver	...	Inner Regional Australia	Adelaide - Central and Hills	Ade Hills
11	20213178	Qld	2021-09-01	Sunday	21:00:00	Multiple	No	No	No	Motorcycle rider	...	Major Cities of Australia	Gold Coast	Coa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9106	20144083	SA	2014-01-01	Friday	11:10:00	Multiple	No	Yes	No	Car passenger	...	Outer Regional Australia	South Australia - South East	Coc
9112	20145108	WA	2014-01-01	Wednesday	11:47:00	Single	No	No	No	Motorcycle rider	...	Major Cities of Australia	Perth - South East	Bel
9121	20144022	SA	2014-01-01	Monday	9:35:00	Single	No	No	No	Pedestrian	...	Major Cities of Australia	Adelaide - North	Tea Gul
9129	20145072	WA	2014-01-01	Tuesday	21:30:00	Single	No	No	No	Car driver	...	Remote Australia	Western Australia - Outback (South)	Esper
9131	20144007	SA	2014-01-01	Tuesday	20:00:00	Single	No	No	No	Pedestrian	...	Major Cities of Australia	Adelaide - North	Pla

6838 rows × 24 columns

4. Remove duplicates as well after removing the missing values

In [7]:

```
data = data.drop_duplicates()
data.reset_index()
```

Out[7]:

	index	Crash ID	State	YYYYMM	Day of week	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	...	National Remoteness Areas	SA4 Name 2016	National LGA Name 2017
0	5	20213034	Qld	2021-09-01	Saturday	4:00:00	Multiple	No	No	No	...	Major Cities of Australia	Brisbane - South	Brisbane (C)
1	8	20213026	Qld	2021-09-01	Wednesday	23:00:00	Multiple	No	No	No	...	Major Cities of Australia	Ipswich	Ipswich (C)
2	9	20213092	Qld	2021-09-01	Saturday	2:00:00	Single	No	No	No	...	Major Cities of Australia	Logan - Beaudesert	Logan (C)
3	10	20214053	SA	2021-09-01	Thursday	21:00:00	Single	No	No	No	...	Inner Regional Australia	Adelaide - Central and Hills	Adelaide Hills (DC)
4	11	20213178	Qld	2021-09-01	Sunday	21:00:00	Multiple	No	No	No	...	Major Cities of Australia	Gold Coast	Gold Coast (C)
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
6817	9106	20144083	SA	2014-01-01	Friday	11:10:00	Multiple	No	Yes	No	...	Outer Regional Australia	South Australia - South East	The Coorong (DC)
6818	9112	20145108	WA	2014-01-01	Wednesday	11:47:00	Single	No	No	No	...	Major Cities of Australia	Perth - South East	Belmont (C)
6819	9121	20144022	SA	2014-01-01	Monday	9:35:00	Single	No	No	No	...	Major Cities of Australia	Adelaide - North	Tea Tree Gully (C)
6820	9129	20145072	WA	2014-01-01	Tuesday	21:30:00	Single	No	No	No	...	Remote Australia	Western Australia - Outback (South)	Esperance (S)
6821	9131	20144007	SA	2014-01-01	Tuesday	20:00:00	Single	No	No	No	...	Major Cities of Australia	Adelaide - North	Playford (C)

6822 rows × 25 columns

A4. Number of crashes in each month

List the number of crashes in each month. In which two months are the number of crashes at their largest?

In [8]:
```
pd.value_counts(data['month'],sort = True)
```

Out[8]:
```
3     654
8     637
7     596
1     593
4     575
12    565
6     556
5     554
9     531
10    530
11    517
2     514
Name: month, dtype: int64
```

March and August had the highest number of crashes

A5. Investigating crashes over different months for specific road user

Now look at the Road User and YYYYMM columns and answer the following questions

1. Compute the average number of crashes against Month for car drivers. To do this,

a. Extract Year and Month as separate columns

In [9]:
```
data
```

Out[9]:

	Crash ID	State	YYYYMM	Day of week	Time	Crash Type	Bus Involvement	Heavy Rigid Truck Involvement	Articulated Truck Involvement	Road User	...	National Remoteness Areas	SA4 Name 2016	Nati N
5	20213034	Qld	2021-09-01	Saturday	4:00:00	Multiple	No	No	No	Motorcycle rider	...	Major Cities of Australia	Brisbane - South	Bris
8	20213026	Qld	2021-09-01	Wednesday	23:00:00	Multiple	No	No	No	Car passenger	...	Major Cities of Australia	Ipswich	Ips
9	20213092	Qld	2021-09-01	Saturday	2:00:00	Single	No	No	No	Car driver	...	Major Cities of Australia	Logan - Beaudesert	Loga
10	20214053	SA	2021-09-01	Thursday	21:00:00	Single	No	No	No	Car driver	...	Inner Regional Australia	Adelaide - Central and Hills	Ade Hills
11	20213178	Qld	2021-09-01	Sunday	21:00:00	Multiple	No	No	No	Motorcycle rider	...	Major Cities of Australia	Gold Coast	Coa
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
9106	20144083	SA	2014-01-01	Friday	11:10:00	Multiple	No	Yes	No	Car passenger	...	Outer Regional Australia	South Australia - South East	Coc
9112	20145108	WA	2014-01-01	Wednesday	11:47:00	Single	No	No	No	Motorcycle rider	...	Major Cities of Australia	Perth - South East	Bel
9121	20144022	SA	2014-01-01	Monday	9:35:00	Single	No	No	No	Pedestrian	...	Major Cities of Australia	Adelaide - North	Tea Gul
9129	20145072	WA	2014-01-01	Tuesday	21:30:00	Single	No	No	No	Car driver	...	Remote Australia	Western Australia - Outback (South)	Esper
9131	20144007	SA	2014-01-01	Tuesday	20:00:00	Single	No	No	No	Pedestrian	...	Major Cities of Australia	Adelaide - North	Pla

6822 rows × 24 columns

This was done in A3.1

b. Compute the number of crashes by both Year and Month for car drivers

In [10]:
```
temp = data[['year','Road User','month']]
temp = temp[temp['Road User']=='Car driver']
pd.DataFrame(temp.groupby(['year','month']).size().reset_index(name='crashes num'))
```

Out[10]:

	year	month	crashes num
0	2014	1	7
1	2014	2	10
2	2014	3	12
3	2014	4	9
4	2014	5	6
...	...	...	...
88	2021	5	33
89	2021	6	41
90	2021	7	43
91	2021	8	39
92	2021	9	21

93 rows × 3 columns

c. Based on task A5-1-b result, compute again the average number of crashes against Month. For each month, the average number of crashes is calculated over different years for which we have collected data for.

In [11]:
```
pd.DataFrame(temp.groupby(['month']).size().div(8).reset_index(name='Average number of crashes'))
```

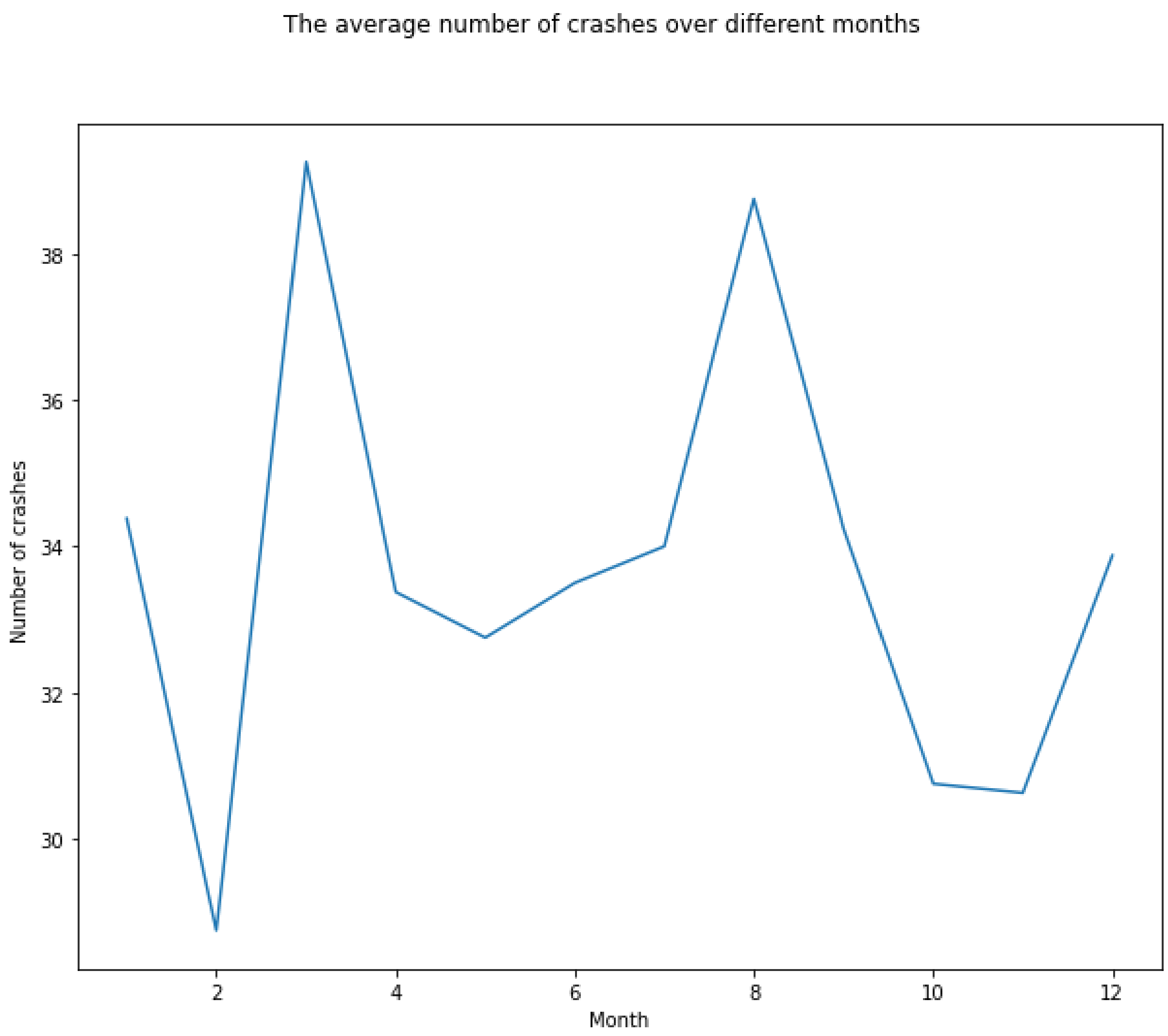
Out[11]:

	month	Average number of crashes
0	1	34.375
1	2	28.750
2	3	39.250
3	4	33.375
4	5	32.750
5	6	33.500
6	7	34.000
7	8	38.750
8	9	34.250
9	10	30.750
10	11	30.625
11	12	33.875

2. Draw a chart showing the average number of crashes over different months computed in task A5-1.

In [12]:

```
temp.groupby(['month']).size().div(8).plot(figsize=(10,8))
plt.xlabel('Month')
plt.ylabel('Number of crashes ')
plt.suptitle('The average number of crashes over different months')
plt.show()
```



3. Discuss any interesting point in the chart

The number of car accidents is highest in March and August, and lowest in February. The whole figure is shaped like a cat's head.

A6. Exploring Speed, National Road Type, and Age

Now look at the Speed, National Road Type, and Age columns and answer the following questions

1. Draw a chart showing the average speed against National Road Type for car drivers

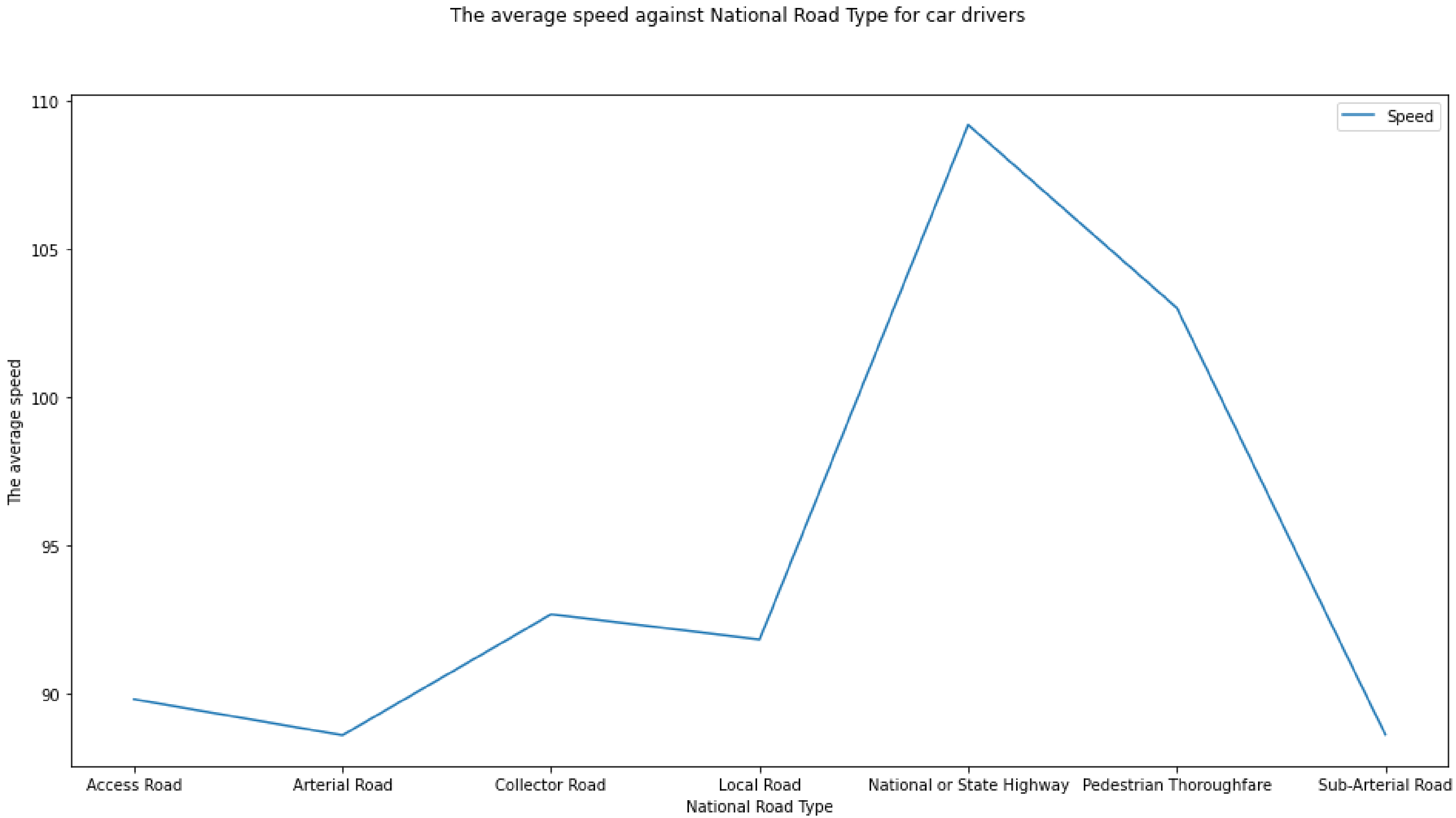
In [13]:
```
A6 = data[['Speed','Road User','National Road Type','Age']]
A61 = A6[A6['Road User']=='Car driver']
A61 = pd.DataFrame(A61.groupby(['National Road Type'])['Speed'].mean())
A61
```

Out[13]:

	Speed
National Road Type	
Access Road	89.785714
Arterial Road	88.571262
Collector Road	92.652439
Local Road	91.803957
National or State Highway	109.177102
Pedestrian Thoroughfare	103.000000
Sub-Arterial Road	88.595573

In [14]:
```
A61.plot(figsize=(16,8))
plt.xlabel('National Road Type')
plt.ylabel('The average speed')
plt.suptitle('The average speed against National Road Type for car drivers')
```

Out[14]:
```
Text(0.5, 0.98, 'The average speed against National Road Type for car drivers')
```



2. Due to measurement error, there are some counter-intuitive values in Age column.Identify those values and replace them with zero.

In [15]:
```
data.loc[data.Age<0,'Age'] = 0
```

Set the measurement error to the age range of less than 0 years, and replace these data with zero.

A7. Relationship between Age, Speed, and Driving Experiences

1. Compute pairwise correlation of columns, Age, Speed, and Driving Experiences for vehicle drivers (such as Motorcycle rider). Which two features have the highest linear association?

In [16]:
```
data['Road User'].unique()
```

Out[16]:
```
array(['Motorcycle rider', 'Car passenger', 'Car driver', 'Pedal cyclist',
       'Pedestrian', 'Other vehicle driver',
       'Motorcycle pillion Car passenger'], dtype=object)
```

In [17]:
```
A7 = data[['Age','Speed','Driving experience','Road User']]
A7_1 = A7[A7['Road User'].isin(['Motorcycle rider','Car driver','Pedal cyclist','Other vehicle driver'])]
# Collect all qualified road users
A7_1
```

Out[17]:

	Age	Speed	Driving experience	Road User
5	19	41	3	Motorcycle rider
9	47	53	12	Car driver
10	24	140	7	Car driver
11	52	71	29	Motorcycle rider
13	32	97	11	Car driver
...	...	...	...	...
9093	34	130	14	Motorcycle rider
9094	26	21	6	Car driver
9105	45	125	14	Car driver
9112	46	142	15	Motorcycle rider
9129	84	74	43	Car driver

4626 rows × 4 columns

In [18]:
```
A7_1.corr()
```

Out[18]:

	Age	Speed	Driving experience
Age	1.000000	-0.005108	0.935406
Speed	-0.005108	1.000000	-0.007659
Driving experience	0.935406	-0.007659	1.000000

The correlation between a variable and itself is one， Age and Driving experience have the strongest linear correlation.

2. Now let's look at the relationship between the number of crashes and Driving Experiences.To do this, first compute the number of crashes against Driving Experiences for vehicle drivers and plot the values of these two features against each other. Is there any relationship between these two features? Describe it.

In [19]:

```
A7_2 = pd.DataFrame(A7_1.groupby(['Driving experience']).size().reset_index(name='Number of crashes'))
A7_2
```

Out[19]:

	Driving experience	Number of crashes
0	1	43
1	2	125
2	3	127
3	4	236
4	5	311
5	6	244
6	7	234
7	8	110
8	11	164
9	12	380
10	13	359
11	14	354
12	15	206
13	26	133
14	27	284
15	28	256
16	29	282
17	30	142
18	40	2
19	41	91
20	42	130
21	43	134
22	44	153
23	45	83

	Driving experience	Number of crashes
24	61	4
25	62	9
26	63	7
27	64	16
28	65	7

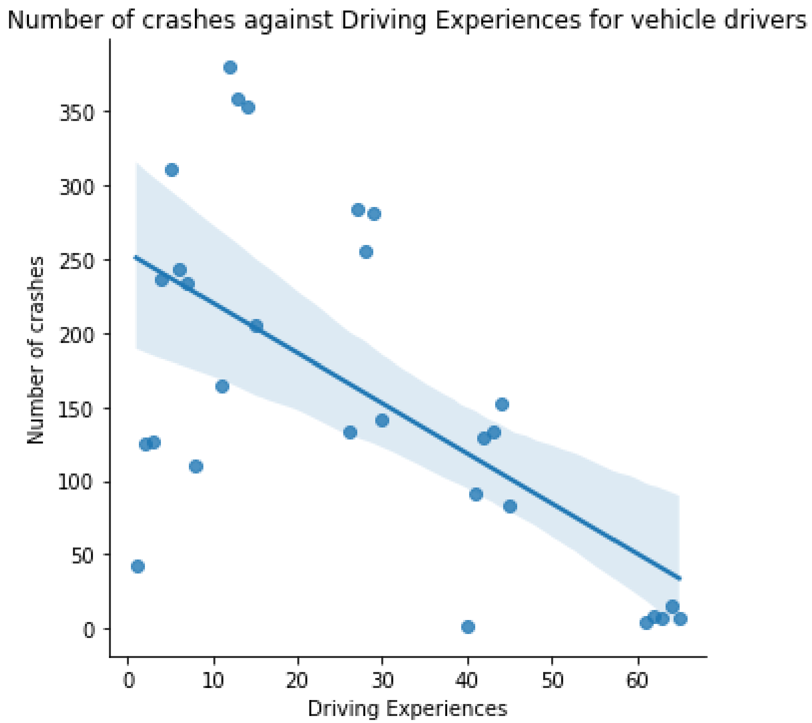
In [20]:
```
A7_2.corr()
```

Out[20]:

	Driving experience	Number of crashes
Driving experience	1.000000	-0.628265
Number of crashes	-0.628265	1.000000

In [21]:
```
# plt.scatter(A7_2['Driving experience'],A7_2['Number of crashes'])
sns.lmplot(data = A7_2,x='Driving experience',y = 'Number of crashes')
plt.xlabel('Driving Experiences')
plt.ylabel('Number of crashes')
plt.title('Number of crashes against Driving Experiences for vehicle drivers')
```

Out[21]:
```
Text(0.5, 1.0, 'Number of crashes against Driving Experiences for vehicle drivers')
```



As can be seen from the above figure, the more experience you have, the fewer collisions you have. So get an experienced driver for safety.

A8. Investigating yearly trend of crash

We will now investigate the trend in the crash over years. For this, you will need to compute the number of crashes by year.

In [22]:
```
A8 = data.groupby('year').size().reset_index(name='number of crashes')
A8
```

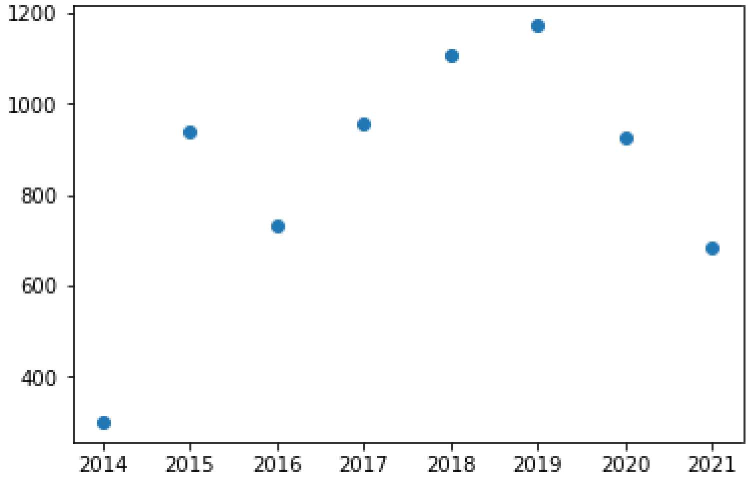
Out[22]:

	year	number of crashes
0	2014	301
1	2015	939
2	2016	733
3	2017	957
4	2018	1108
5	2019	1173
6	2020	926
7	2021	685

1. Fit a linear regression using Python to this data (The number of crashes over different years) and plot the linear fit.

In [23]:

```
plt.scatter(A8['year'],A8['number of crashes'])
plt.show()
```



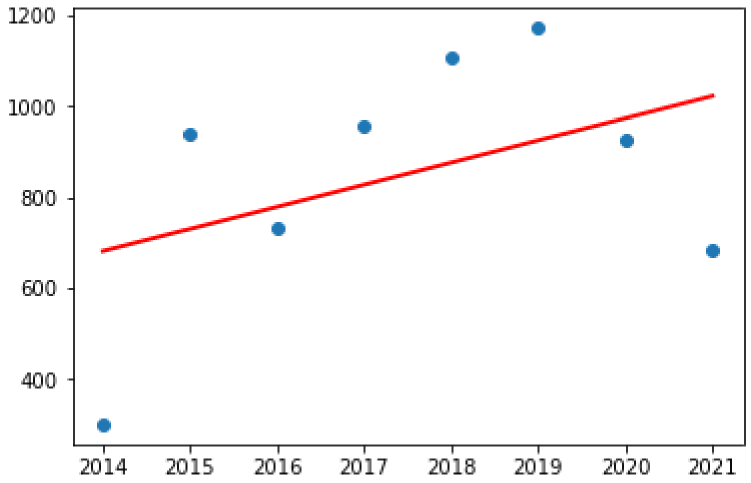
In [24]:

```
slope, intercept, r_value, p_value, std_err = linregress(A8['year'],A8['number of crashes'])
```

In [25]:
```
print("slope: %f     intercept: %f" % (slope, intercept))
print("r-value: %f" % r_value)
print("p-value: %f" % p_value)
print("std-err: %f" % std_err)
```

```
slope: 48.738095     intercept: -97476.357143
r-value: 0.430510
p-value: 0.286985
std-err: 41.715525
```

In [26]:
```
line = [slope*xi + intercept for xi in A8['year']]
# We can then plot the 'line':

plt.plot(A8['year'],line,'r-', linewidth=2)
# And add the original data points to the same plot:

plt.scatter(A8['year'], A8['number of crashes'])
plt.show()
```



2. Use the linear fit to predict the number of crashes in 2022.

In [27]:
```
slope*2022+intercept
```

Out[27]:
```
1072.0714285714348
```

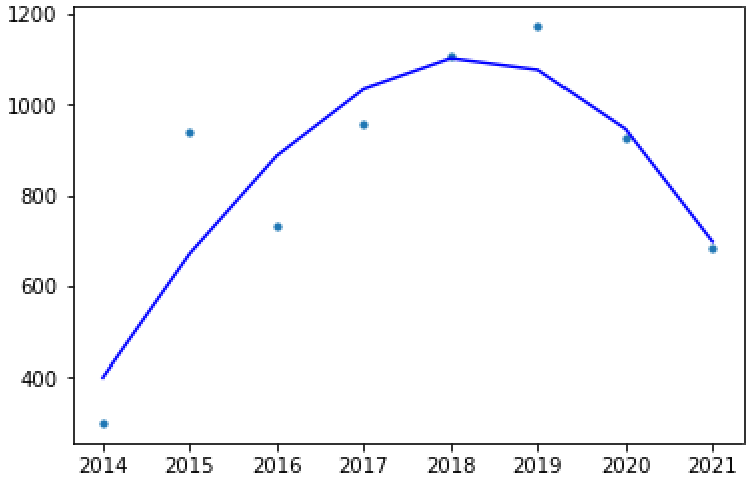
The predicted number of collisions in 2022 is 1072

3. Can you think of a better model that well captures the trend of yearly crash? Develop a new model and explain why it is better suited for this task.

In [28]:

```
#import required packages
import operator
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.preprocessing import PolynomialFeatures

#provide data
np.random.seed(0)
x = A8['year'].to_numpy()
y = A8['number of crashes'].to_numpy()

# transforming the data to include another axis
x = x[:, np.newaxis]
y = y[:, np.newaxis]

#create polynomial regression
polynomial_features= PolynomialFeatures(degree= 20)
x_poly = polynomial_features.fit_transform(x)


model = LinearRegression()
model.fit(x_poly, y)
y_poly_pred = model.predict(x_poly)

rmse = np.sqrt(mean_squared_error(y,y_poly_pred))
r2 = r2_score(y,y_poly_pred)
print(rmse)
print(r2)

plt.scatter(x, y, s=10)
# sort the values of x before line plot
sort_axis = operator.itemgetter(0)
sorted_zip = sorted(zip(x,y_poly_pred), key=sort_axis)
x, y_poly_pred = zip(*sorted_zip)
plt.plot(x, y_poly_pred, color='b')

plt.show()
```

```
122.94365591029292
0.7753620650654187
```



This new model is better than the original model because the trend in the data is up and down, while the original model can only see a large overall trend, and cannot accurately represent the short-term changes in quantity. It can also be seen from the initial scatter plot that the trend of the data is not static, which may also be because the year and number of collisions are not highly correlated and not closely correlated. Therefore, the polynomial model fits the data better than the linear model.

4. Use your new model to predict the number of crashes in 2022.

In [29]:

```
model.predict(polynomial_features.fit_transform([[2022]]))
```

Out[29]:

```
array([[318.85863018]])
```

A9. Filling in missing values

Rather than replacing some counter-intuitive values with zero in task A6, use a better (e.g., model-based) approach to fill in the counter-intuitive values

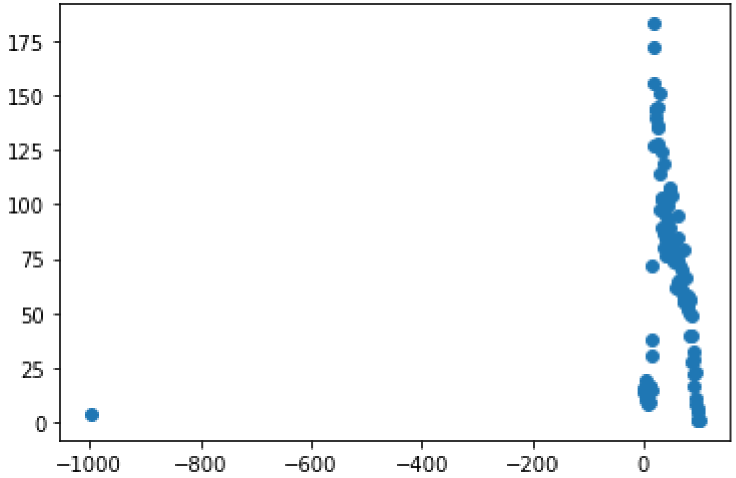
In [30]:

```
data = pd.read_csv('Australian_Road_Deaths.csv',encoding='utf-8')
data['YYYYMM']=pd.to_datetime(data['YYYYMM'],format='%Y%m') #to_datetime Convert the type of the "YYYYMM" column to date-time forma
data['month']=data['YYYYMM'].dt.month # dt.month get the month of date
```

```
data['year']=data['YYYYMM'].dt.year # dt.year get the year of date
data = data.replace('Unspecified',np.nan) # Use Replace to replace 'Unspecified'  with the null value 'NaN'
data = data.replace('Undetermined',np.nan)# Use Replace to replace 'Undetermined'  with the null value 'NaN'
df = data.loc[data.isnull().T.any()]
data = data.dropna(how ='any',axis =0)
data = data.drop_duplicates()
```

In [31]:
```
A9 = data.groupby(['Age']).size().reset_index(name = 'size')
plt.scatter(A9['Age'],A9['size'])
```

Out[31]: <matplotlib.collections.PathCollection at 0x291e9f44220>



Task B: Decision Tree Classification on Song Popularity Dataset and K-means Clustering on Other Data

B1. Classification

In [32]:
```
data = pd.read_csv('song_data.csv',encoding='utf-8')
```

In [33]:
```
data.head()
```

Out[33]:

	song_name	song_popularity	song_duration_ms	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	audio_mode	speechin
0	Boulevard of Broken Dreams	4	262333	0.005520	0.496	0.682	0.000029	8	0.0589	-4.095	1	0.0
1	In The End	4	216933	0.010300	0.542	0.853	0.000000	3	0.1080	-6.407	0	0.0
2	Seven Nation Army	4	231733	0.008170	0.737	0.463	0.447000	0	0.2550	-7.828	1	0.0
3	By The Way	4	216933	0.026400	0.451	0.970	0.003550	0	0.1020	-4.938	1	0.1
4	How You Remind Me	3	223826	0.000954	0.447	0.766	0.000000	10	0.1130	-5.065	1	0.0

In [34]:

```
df = data.iloc[:,1:]
# Just take the column of numbers
```

In [35]:

```
X = df.iloc[:,1:]
y = df.iloc[:,:1]
# y take the song_popularity column, X take the other column
```

In [36]:

```
X
```

Out[36]:

	song_duration_ms	acousticness	danceability	energy	instrumentalness	key	liveness	loudness	audio_mode	speechiness	tempo	time_signat
0	262333	0.005520	0.496	0.682	0.000029	8	0.0589	-4.095	1	0.0294	167.060	
1	216933	0.010300	0.542	0.853	0.000000	3	0.1080	-6.407	0	0.0498	105.256	
2	231733	0.008170	0.737	0.463	0.447000	0	0.2550	-7.828	1	0.0792	123.881	
3	216933	0.026400	0.451	0.970	0.003550	0	0.1020	-4.938	1	0.1070	122.444	
4	223826	0.000954	0.447	0.766	0.000000	10	0.1130	-5.065	1	0.0313	172.011	
...	...	...	...	...	...	...	...	...	...	...	...	
18830	159645	0.893000	0.500	0.151	0.000065	11	0.1110	-16.107	1	0.0348	113.969	
18831	205666	0.765000	0.495	0.161	0.000001	11	0.1050	-14.078	0	0.0301	94.286	
18832	182211	0.847000	0.719	0.325	0.000000	0	0.1250	-12.222	1	0.0355	130.534	
18833	352280	0.945000	0.488	0.326	0.015700	3	0.1190	-12.020	1	0.0328	106.063	
18834	193533	0.911000	0.640	0.381	0.000254	4	0.1040	-11.790	1	0.0302	91.490	

18835 rows × 13 columns

In [37]:

```
y
```

Out[37]:

	song_popularity
0	4
1	4
2	4
3	4
4	3
...	...
18830	3
18831	3
18832	2
18833	3
18834	3

18835 rows × 1 columns

1. Divide the data set into a 75% training set and a 25% testing set using only the features relevant for classification.

In [38]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size = 0.25, random_state = 42)
```

2. Use feature scaling and train a decision tree model.

In [39]:
```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

In [40]:
```
from sklearn.tree import DecisionTreeClassifier
classifier = DecisionTreeClassifier()
classifier.fit(X_train, y_train)
```

Out[40]:

```
DecisionTreeClassifier()
```

3. Using the test set, predict using the decision tree and compute the confusion matrix and the accuracy of classification.

In [41]:

```
print('accuracy:',classifier.score(X_test,y_test))
```

```
accuracy: 0.4890634954342748
```

In [42]:

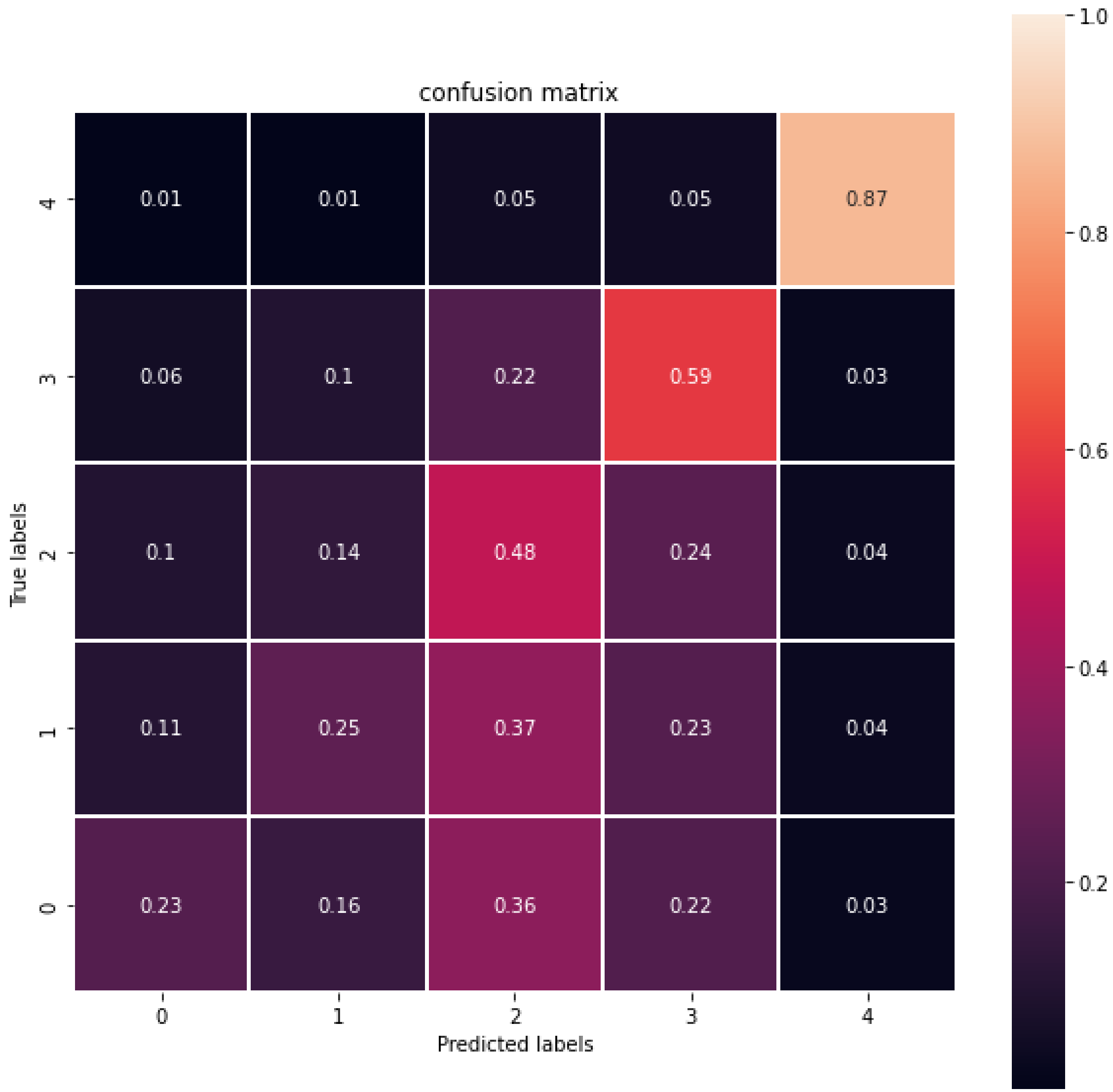
```
# Predicting the Test set results
y_pred = classifier.predict(X_test)
```

In [43]:

```
# Making the Confusion Matrix
from sklearn.metrics import confusion_matrix
con_mat = confusion_matrix(y_test, y_pred)
```

In [44]:

```
con_mat_norm = con_mat.astype('float') / con_mat.sum(axis=1)[:, np.newaxis]
con_mat_norm = np.around(con_mat_norm, decimals=2)

plt.figure(figsize=(10, 10))
sns.heatmap(con_mat_norm,linewidths=0.1,vmax=1.0, square=True,linecolor='white', annot=True)
plt.ylim(0,5)
plt.xlabel('Predicted labels')
plt.ylabel('True labels')
plt.title(' confusion matrix')
plt.show()
```



4. Discuss your findings from the confusion matrix and accuracy. You should consider other performance metrics you learnt in lecture 7 to answer this question.

In [45]:

```
from sklearn.metrics import f1_score
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
```

```
f1_score(y_test, y_pred, average='micro')
```

Out[45]: 0.4890634954342748

In [46]:
```
precision_score(y_test, y_pred, average='weighted')
```

Out[46]: 0.47834121759148124

In [47]:
```
recall_score(y_test, y_pred, average='macro')
```

Out[47]: 0.4866103883014675

The best classification accuracy of class 5 is 88%, while the accuracy of class 1 and Class 2 is low

B2. Clustering

dataset： https://www.kaggle.com/datasets/madhurpant/world-population-data

In [48]:
```
df2=pd.read_csv('height_weight_data.csv')
```

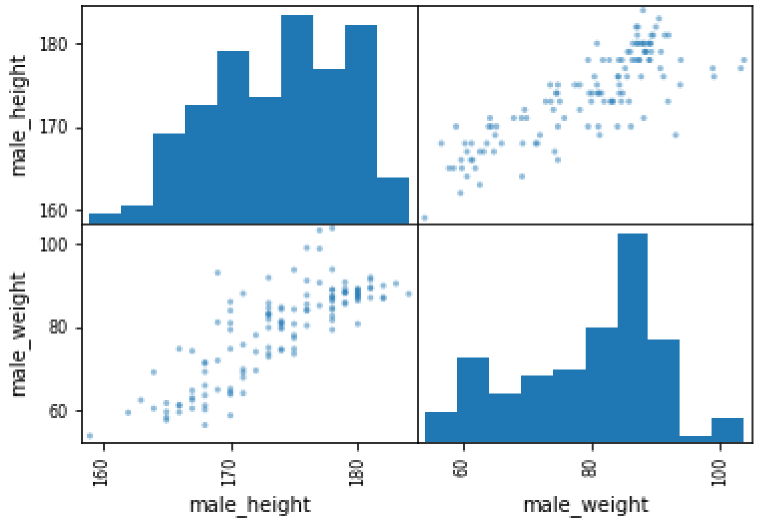
In [49]:
```
df2.head()
```

Out[49]:

	country	male_height	female_height	male_weight	female_weight	male_bmi	female_bmi
0	Netherlands	184	170	87.9	73.2	26.1	25.3
1	Montenegro	183	170	90.4	75.3	27.0	26.2
2	Estonia	182	168	89.9	73.7	27.0	26.0
3	Denmark	182	169	86.8	70.2	26.3	24.6
4	Bosnia and Herzegovina	182	167	87.1	70.6	26.4	25.3

In [50]:
```
df2 = df2.iloc[:,1:]
# Just take the column of numbers
```

In [51]:
```
from pandas.plotting import scatter_matrix
scatter_matrix(df2.iloc[:,[0,2]])
plt.show()
```



In [52]:
```
X = df2[["male_height", "male_weight"]]
X
```

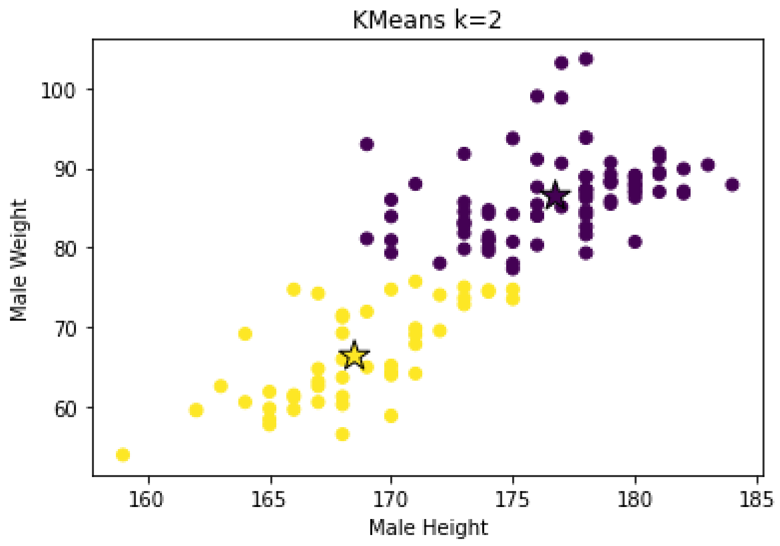
Out[52]:

	male_height	male_weight
0	184	87.9
1	183	90.4
2	182	89.9
3	182	86.8
4	182	87.1
...	...	...
121	164	60.5
122	164	69.1
123	163	62.5
124	162	59.5
125	159	53.9

126 rows × 2 columns

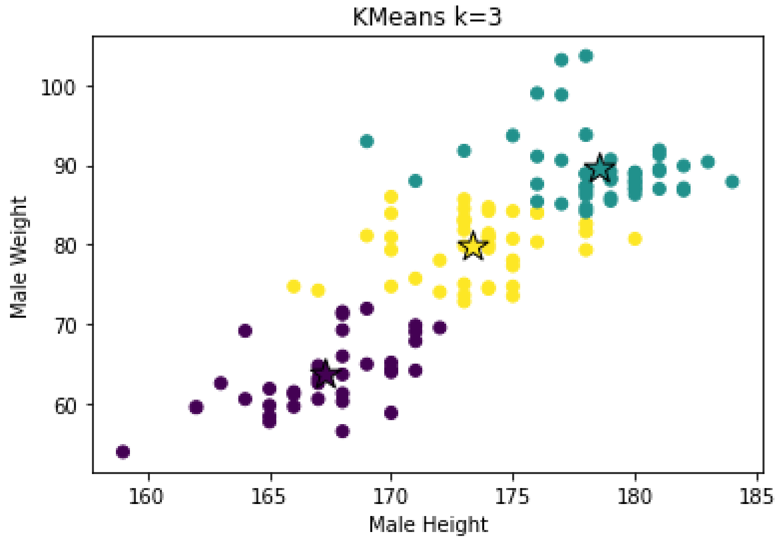
In [53]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
y_pred = kmeans.predict(X)

Xx=X.values

plt.scatter(Xx[:,0], Xx[:,1], c=y_pred)

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], marker="*", s=250, c=[0,1], edgecolors="k")

plt.xlabel("Male Height")
plt.ylabel("Male Weight")
plt.title("KMeans k=2")
plt.show()
```



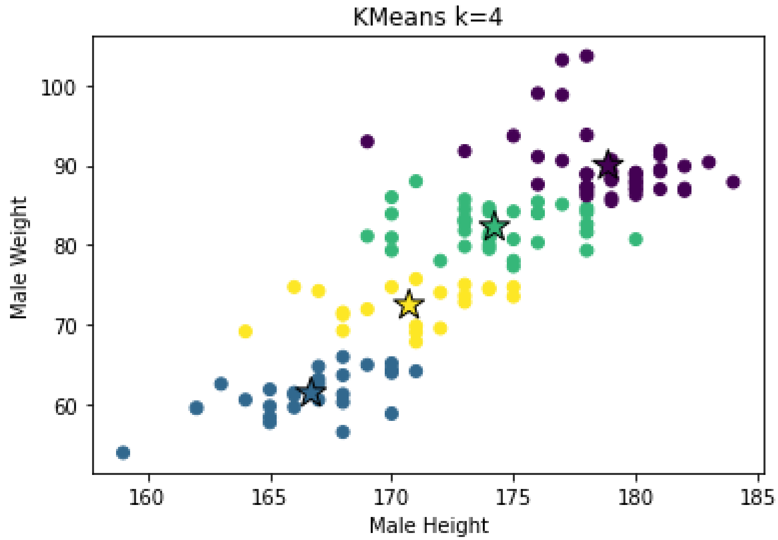
In [54]:

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
y_pred = kmeans.predict(X)

Xx=X.values

plt.scatter(Xx[:,0], Xx[:,1], c=y_pred)

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], marker="*", s=250, c=[0,1,2], edgecolors="k")

plt.xlabel("Male Height")
plt.ylabel("Male Weight")
plt.title("KMeans k=3")
plt.show()
```

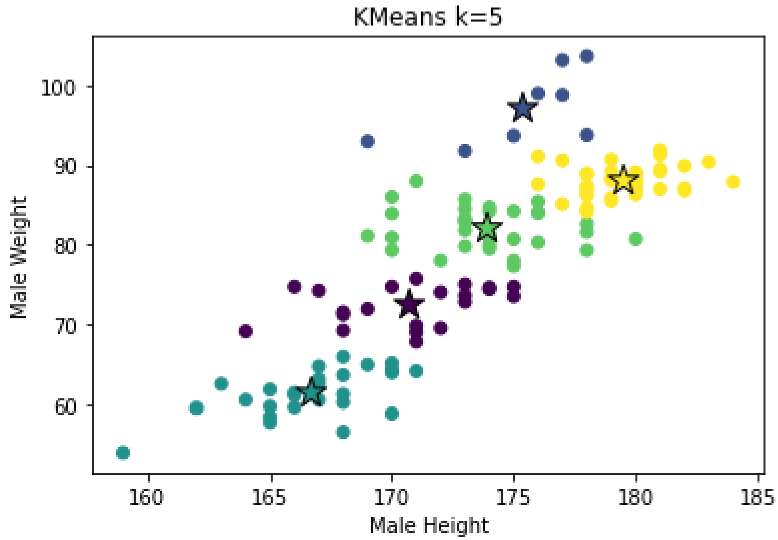


In [55]:

```
kmeans = KMeans(n_clusters=4)
kmeans.fit(X)
y_pred = kmeans.predict(X)

Xx=X.values

plt.scatter(Xx[:,0], Xx[:,1], c=y_pred)

plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], marker="*", s=250, c=[0,1,2,3], edgecolors="k")

plt.xlabel("Male Height")
plt.ylabel("Male Weight")
plt.title("KMeans k=4")
plt.show()
```



In [56]:
```
kmeans = KMeans(n_clusters=5)
kmeans.fit(X)
y_pred = kmeans.predict(X)

Xx=X.values

plt.scatter(Xx[:,0], Xx[:,1], c=y_pred)


plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], marker="*", s=250, c=[0,1,2,3,4], edgecolors="k")

plt.xlabel("Male Height")
plt.ylabel("Male Weight")
plt.title("KMeans k=5")
plt.show()
```



Kmeans clustered the data distribution into 5 classes. When the number of clusters was 2, the intra-group distance was minimized and the inter-group distance was maximized, so the Kmeans clustering effect was the best when the number of clusters was 2