

Predicting accidents

Personal challenge progress report

AI for society
Stefan Hobeijn

Table of contents

Table of contents	1
Linear regression on accident dataset	2
Finding first dataset	2
Preparing data for simple linear regression of total accidents per year	2
linear regression of total accidents per year	4
Preparing data for simple linear regression of total accidents per year for one province	5
Simple linear regression of total accidents per year for one province	7
Finding an additional dataset	8
Adding population and vehicle datasets	8
Preparing the population dataset	8
Compare the total amount of accidents to the population and the population density	10
Compare the total amount of accidents to the amount of personal vehicles	11
Selecting features	13
Using a correlation matrix to choose features	13
Applying regression, scoring and comparing	13
Applying multiple linear regression and scoring	13
Using different types of regression	16
Conclusion	18
Future recommendations	18

Linear regression on accident dataset

Finding first dataset

I have found a dataset that consists of all accidents that happened within a year. Each row is an instance of one accident. There is quite some information to be gotten here, but there are also a lot of columns that empty/almost empty.

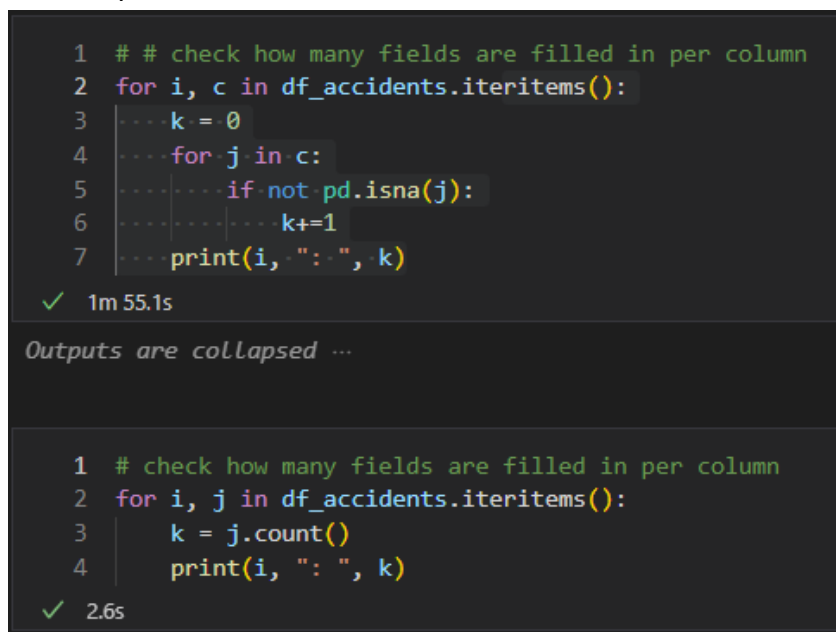
<https://data.overheid.nl/dataset/9841-verkeersongevallen---bestand-geregistreerde-ongevallen-nederland#panel-resources>

Preparing data for simple linear regression of total accidents per year

To analyse the “accidents” dataset, I have looked into the usable columns. First I wanted to see how many empty columns there were, so I printed the amount of filled in values for each column. This however took way too long so I looked to see if I could improve on it. I found a method in the pandas library which does the exact same for me. This sped up the code significantly for the exact same output.

Figure 1

The comparison between two methods.



```
1 # # check how many fields are filled in per column
2 for i, c in df_accidents.iteritems():
3     ... k = 0
4     ... for j in c:
5         ... if not pd.isna(j):
6             ... k += 1
7     ... print(i, ": ", k)
✓ 1m 55.1s

Outputs are collapsed ...

1 # check how many fields are filled in per column
2 for i, j in df_accidents.iteritems():
3     k = j.count()
4     print(i, ": ", k)
✓ 2.6s
```

After knowing what columns are empty/almost empty I wanted to drop these columns to make the dataset smaller. Doing this I had the same problem and solution as in Figure 1.

Next I wanted to plot the total amount of accidents each year. This didn't take too long, but I managed to make it faster anyway. Also with the same output.

Figure 2

A comparison between two methods

```
1 # plot yearly accidents
2
3 min_year = df_2.min(axis='rows', numeric_only=True)[1].astype(int)
4 max_year = df_2.max(axis='rows', numeric_only=True)[1].astype(int)
5 year = range(min_year, max_year + 1)
6 n_accidents = []
7 for i, j in enumerate(year):
8     n = 0
9     for k in df_2.iloc[:,2]:
10         if j == k:
11             n+=1
12     n_accidents.insert(i, n)
13
14 print(year)
15 print(n_accidents)
16 fig, ax = plt.subplots()
17 ax.plot(year, n_accidents)
18 plt.show()
```

✓ 5.4s

Outputs are collapsed ...

```
1 # plot yearly accidents
2
3 min_year = df_2.min(axis='rows', numeric_only=True)[1].astype(int)
4 max_year = df_2.max(axis='rows', numeric_only=True)[1].astype(int)
5 # get the range of the years
6 year = range(min_year, max_year + 1)
7 # get a dataframe of the amount of accidents for each year
8 df_temp = df_2.groupby("JAAR_VKL")["JAAR_VKL"].count()
9 fig, ax = plt.subplots()
10 ax.plot(year, df_temp.iloc[:])
11 plt.show()
```

✓ 0.4s

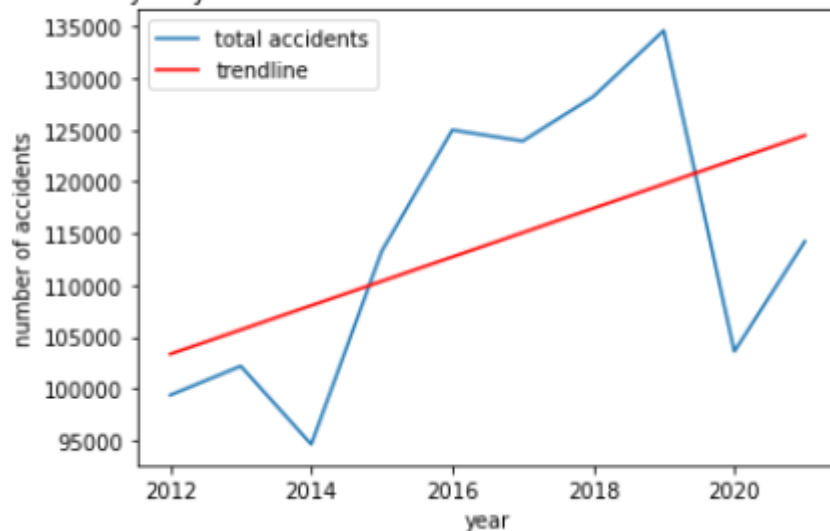
Figure 3
The output of the methods in Figure 2
Yearly total number of road accidents in the Netherlands



linear regression of total accidents per year

Having plotted the yearly accidents, I decided to apply linear regression on this plot. This is done without splitting the data between test and train data.

Figure 4
yearly total number of road accidents in the Netherlands



Preparing data for simple linear regression of total accidents per year for one province

To apply linear regression for one province I started to look which province I wanted to use as an example. To do this I wanted to plot the total amount of accidents each year for each province.

To do this I first decreased the amount of columns to three different columns. The columns I use are the year, the province code (a two letter abbreviation of the province name) and an enumerated version of the province code.

Figure 5

The method and the top 5 rows in the above mentioned dataset

```
1 df_5 = df_4[["JAAR_VKL", "PVE_CODE", "PVE_NUMR"]]
2 df_5.head(5)
```

✓ 0.6s

	JAAR_VKL	PVE_CODE	PVE_NUMR
0	2014	ZH	1
1	2014	ZH	1
2	2014	ZH	1
3	2014	ZH	1
4	2014	ZH	1

Next I made a new dataset which contains the amount of accidents for each province for each year. This code took way too long to execute so I decided to save it to a CSV file so that the code does not have to run each time the project is opened. Next I loaded the CSV file and checked it.

Figure 6

Time to run the code mentioned above

✓ 37m 54.1s

Figure 7

The dataframe loaded from the csv file.

	YEAR	PROVINCE	N_ACCIDENTS	PVE_CODE
0	2014	ZH	18208	1
1	2014	NB	15082	2
2	2014	LB	6436	3
3	2014	GL	12273	4
4	2014	FL	1595	5
...
115	2018	FR	3417	8
116	2018	GR	3566	9
117	2018	DR	2854	10
118	2018	ZL	2432	11
119	2018	OV	8636	12

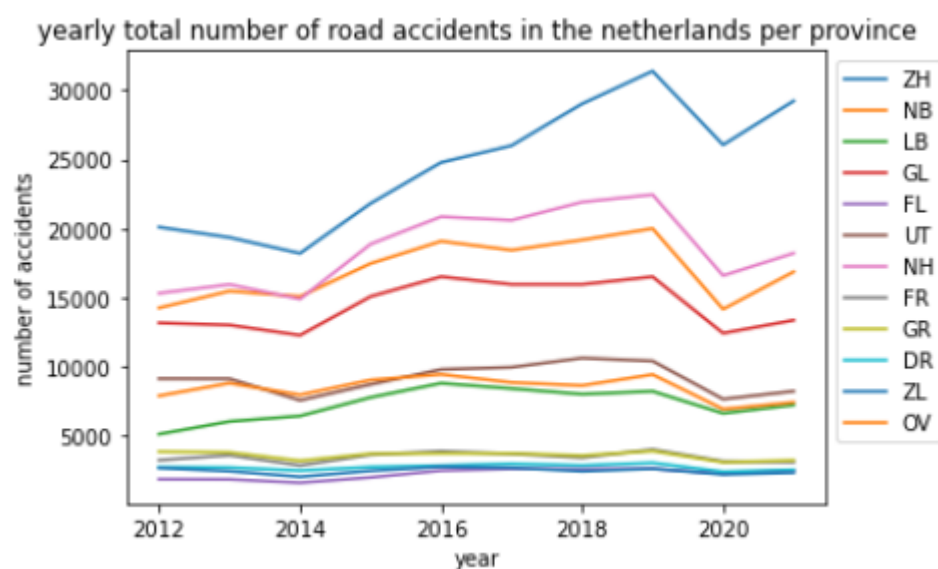
120 rows × 4 columns

The province codes are not saved to the file but instead are added again when loading.

This code could also be improved but since the results only change once a year and are already saved to a file I opted for continuing the project like this.

Next I plotted the total road accidents per province.

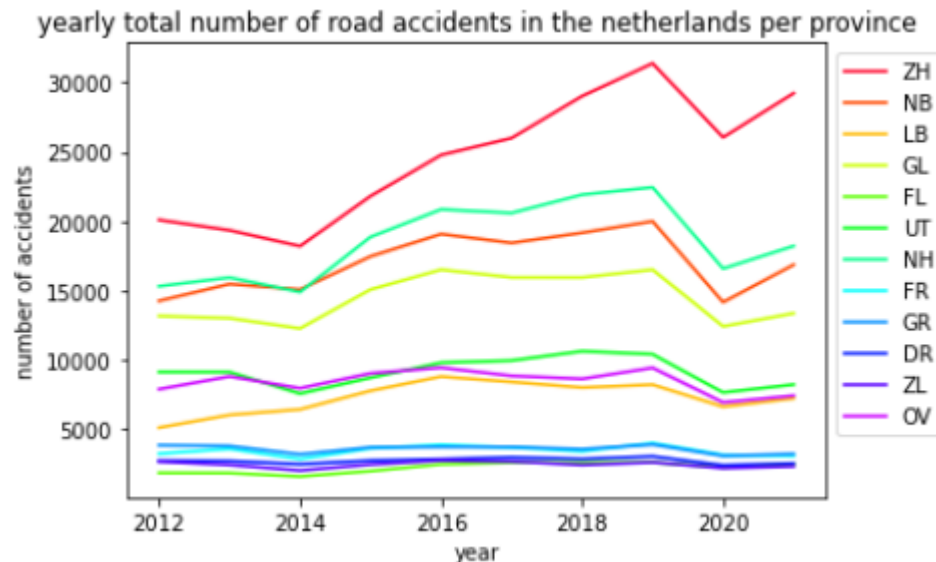
Figure 8



This plot has one problem. There are twelve lines and the matplotlib library automatically assigns up to ten different colours. So some lines have the same colours. To make the plot more clear I changed the colours. First I tried a spectrum from red to green. This made the

plot even less clear as most colours looked way too similar. Finally I found a rainbow spectrum that I now use. This way no colours are repeated and all of them are mostly distinguishable.

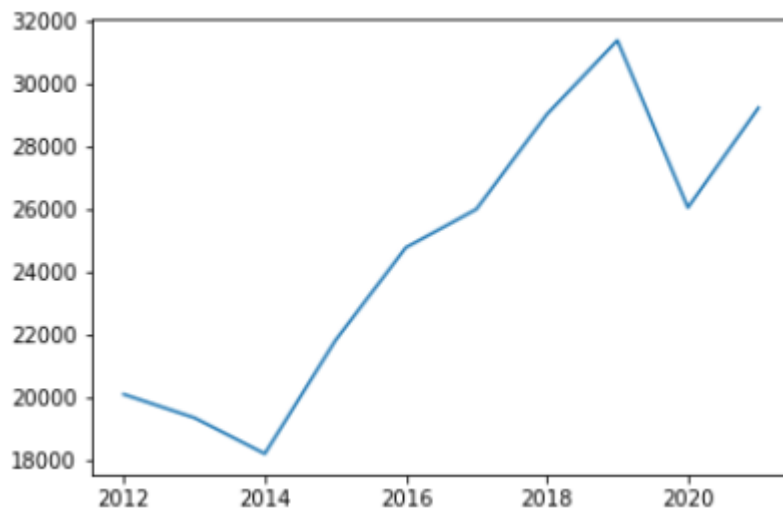
Figure 9



To single out a single province I created a dataset from the “accidents per province” dataset and only took the rows with the chosen province. I chose ZH (Zuid-Holland) to use as an example. I also made a plot for this province.

Figure 10

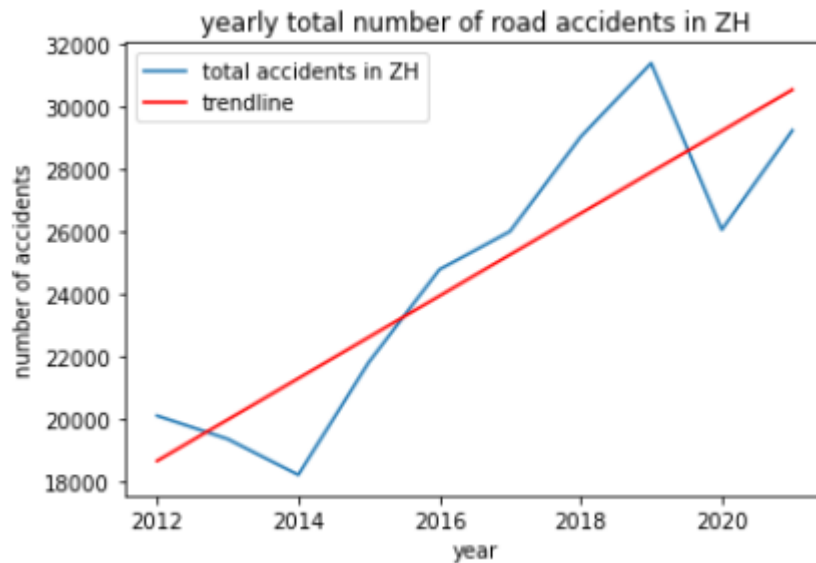
Yearly total number of accidents in Zuid-Holland



Simple linear regression of total accidents per year for one province

To apply linear regression, first I needed to sort the dataset of the single province by year. Next I applied linear regression the same way as with the total of the Netherlands.

Figure 11



Finding an additional dataset

To make the project more complex I found another dataset. This dataset contains data about subjects like population, vehicles and more. For this project only the two aforementioned are used. These subjects are, just like the accident dataset, on a yearly basis.

<https://opendata.cbs.nl/statline/#/CBS/nl/dataset/70072NED/table?ts=1669975216124&fromstatweb=true>

These subjects are also split between two different datasets. One population dataset and one vehicle dataset. This is to avoid an unnecessary big and unclear dataset.

Adding population and vehicle datasets

Preparing the population dataset

In order to use the dataset correctly I first made sure all dates and periods lined up with the accident dataset. This means all columns that are recorded on 1st of January are to be put back to the 31st of december of the previous year. All columns that are recorded on the 31st of december are not to be changed.

I did this by downloading a dataset with the columns of the 1st of January separately from a dataset with the columns of the 31st of december. Then I changed the years of the dataset of January and combined both. This way the whole dataset can be properly used together with the accidents dataset.

To load one of these two datasets I first needed to make sure I could use them in the code. The dataset had some unnecessary text that made pandas give some weird rows. There was some extra text in the CSV file that made it unusable. I could just delete it and that fixed the problem. (Figure 12 and Figure 13)

Also there were two rows explaining the columns, one for the name and one for the unit of measurement. To solve this without just deleting the row, I decided to add the unit of measurement in brackets to the column name. This way I won't lose the information and can use the dataset properly. (Figure 14)

Figure 12

Top of the population dataset

Regionale kerncijfers Nederland							
Onderwerp							
Bevolking							
Perioden	Regio's	aantal	aantal	aantal	aantal	aantal	aantal
2012	Groningen	580875	288754	292121	28662	29853	
2012	Fryslân (PV)	647214	323495	323719	35132	39058	

Figure 13

Bottom of the population dataset

2022	Gelderland	2110472	1048162
2022	Utrecht (PV)	1369873	674377
2022	Noord-Holland	2909827	1438477
2022	Zuid-Holland	3753944	1857303
2022	Zeeland (PV)	386767	192162
2022	Noord-Brabant	2592874	1301018
2022	Limburg (PV)	1118302	556421
Bron: CBS			

Figure 14

The new column names

	Regio's (naam)	Perioden (jaar)	Bevolking Bevolkingssamenstelling op 1 januari Totale bevolking (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Geslacht Mannen (aantal)
0	Groningen (PV)	2012	581705	289275
1	Groningen (PV)	2013	582728	289820
2	Groningen (PV)	2014	583942	290840

To make pandas see all the values correctly I saved the dataset and loaded it again. I also added a comma to be recognised as a decimal number because otherwise pandas sets it to a string and not a decimal datatype.

Compare the total amount of accidents to the population and the population density

To compare the amount of accidents to the “population” dataset I decided to only use the total population and the population density.

I plotted the total amount of accidents per province (Figure 15), the total population per province (Figure 16) and the population density per province (Figure 17). **Do keep in mind that the provinces don't have the same colour in every plot.**

Figure 15

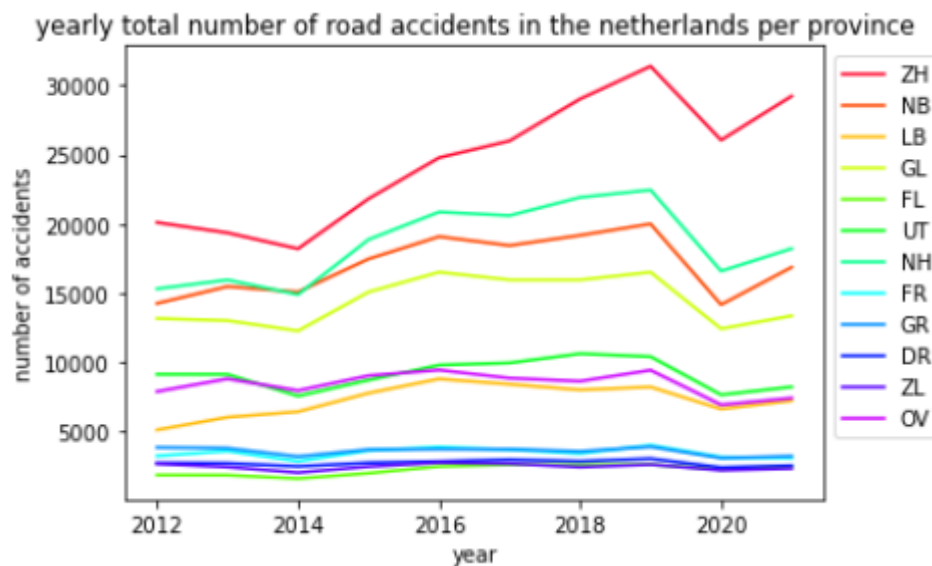


Figure 16

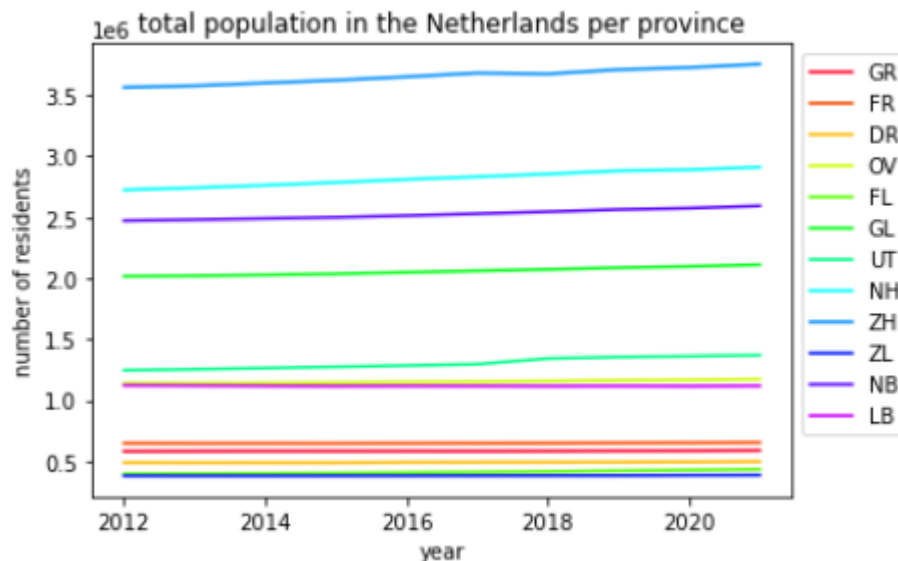
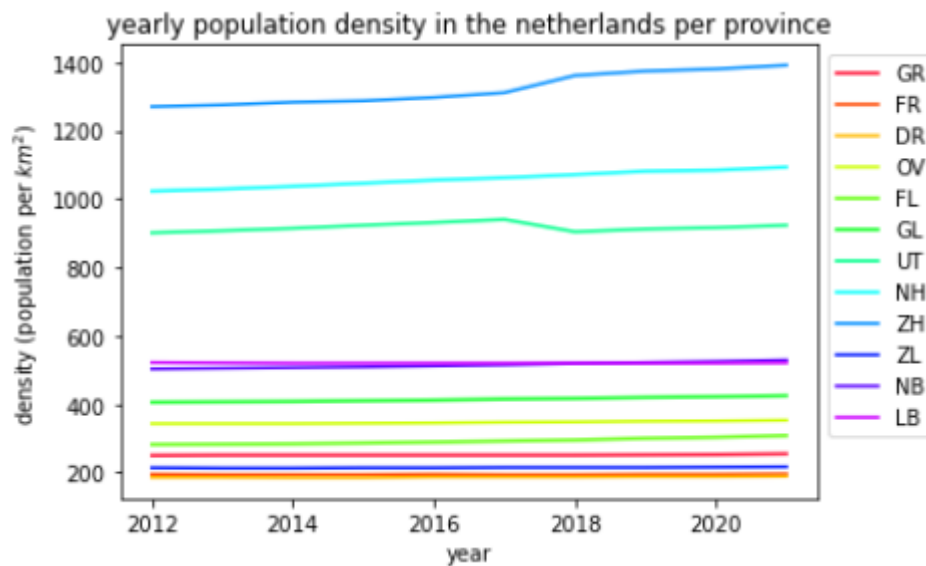
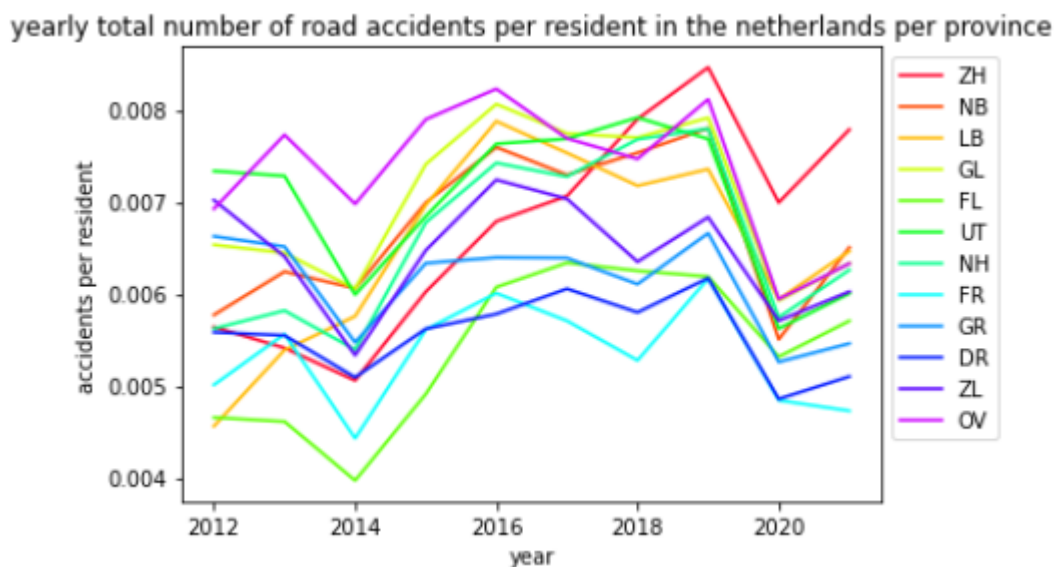


Figure 17



Seeing these plots I decided to make a comparison. I plotted the accidents per resident per province.

Figure 18



Preparing the vehicle dataset

To prepare the vehicle dataset I did not need to divide the dataset in two different datasets. I did however need to remove some extra text and combine the column names and unit of measurement.

Compare the total amount of accidents to the amount of personal vehicles

To compare the amount of accidents per province to the amount of personal vehicles per province I first plotted the amount of vehicles separately (Figure 19) and then I plotted the

amount of vehicles per accident per province (Figure 20). All columns of the dataset are recorded on the 1st of january, so the date is put back by one day. **Also here the provinces don't have the same colour in each plot.**

Figure 19

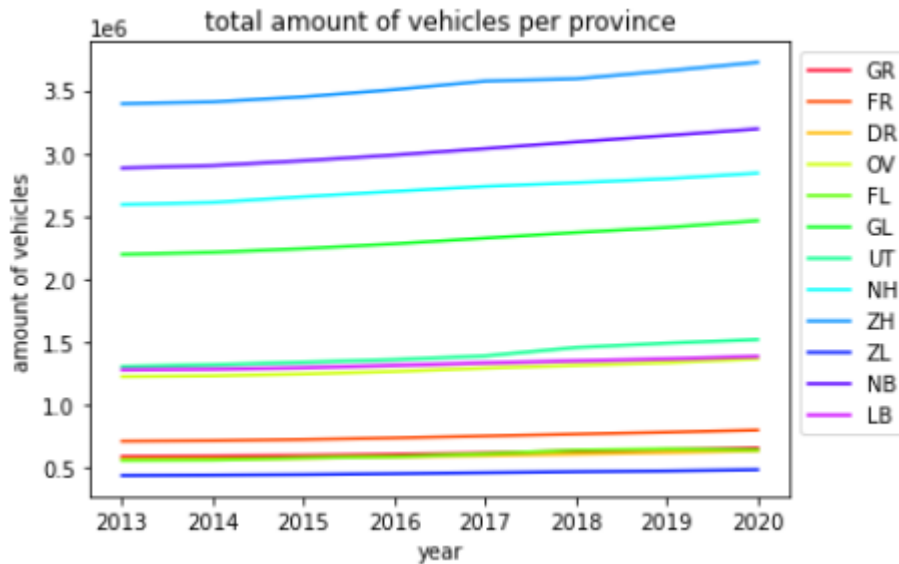
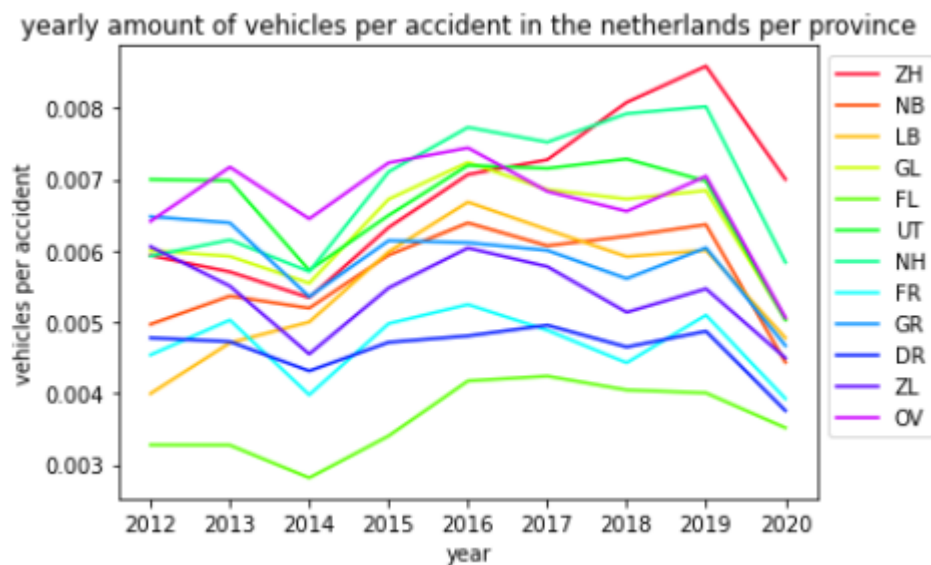


Figure 20



Because of the vehicle dataset missing a year, I decided to not use this in the project.

Selecting features

Using a correlation matrix to choose features

To find correlations I first tried to make a correlation matrix of a simple dataset. The dataset used is the dataset I made with the amount of accidents per province per year. Since the provinces were using names and not numbers this didn't work. This is why earlier in the project I gave the provinces a number. With that the correlation matrix works and I can make a correlation matrix without problem.

Figure 21

	YEAR	N_ACCIDENTS	PVE_CODE
YEAR			
N_ACCIDENTS	0.08		
PVE_CODE	0.00	-0.62	

I then did the same for the dataset of the population. I also added the amount of accidents since that is what I wanted to predict. This gave a huge correlation matrix.

Figure 22

	Perioden (jaar)	Bevolking Bevolkingssamenstelling op 1 januari Totale bevolking (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Geslacht Mannen (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Geslacht Vrouwen (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen Jonger dan 5 jaar (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 5 tot 10 jaar (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 10 tot 15 jaar (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 15 tot 20 jaar (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 20 tot 25 jaar (aantal)	Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 25 tot 30 jaar (aantal)
Perioden (jaar)										
Bevolking Bevolkingssamenstelling op 1 januari Totale bevolking (aantal)	0.02									
Bevolking Bevolkingssamenstelling op 1 januari Geslacht Mannen (aantal)	0.02	1.00								
Bevolking Bevolkingssamenstelling op 1 januari Geslacht Vrouwen (aantal)	0.02	1.00	1.00							
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen Jonger dan 5 jaar (aantal)	-0.02	0.99	0.99	0.99						
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 5 tot 10 jaar (aantal)	-0.02	0.99	0.99	1.00	1.00					
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 10 tot 15 jaar (aantal)	-0.03	1.00	0.99	1.00	1.00	1.00				
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 15 tot 20 jaar (aantal)	0.02	1.00	1.00	1.00	0.99	0.99	1.00			
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 20 tot 25 jaar (aantal)	0.03	1.00	0.99	1.00	0.99	0.99	0.99	1.00		
Bevolking Bevolkingssamenstelling op 1 januari Leeftijd Leeftijdsgroepen 25 tot 30 jaar (aantal)	0.03	1.00	0.99	1.00	0.99	0.99	0.99	0.99	1.00	

The matrix is too big to fit on one screen

For simplicity I took some groups of features and applied linear regression using these. I also took those groups and made another correlation matrix which is smaller. Using this matrix I chose two sets of features where the correlation was close to 1 or -1 with the amount of accidents but closer to 0 with each other.

Applying regression, scoring and comparing

Applying multiple linear regression and scoring

I applied simple linear regression on the following groups:

- Year and population

- All features
- All features with a correlation higher than a certain number with the amount of accidents
- Front he correlation matrix: Mediocre urbanity (the middle one of urbanity) and population density
- From the correlation matrix: Total population and population density
- All features below this point together
- Men and women
- Features based on age groups that can legally drive
- Total population and population of 15 years and older
- Features based on migration background
- Feathers about urbanity
- Total population and population density

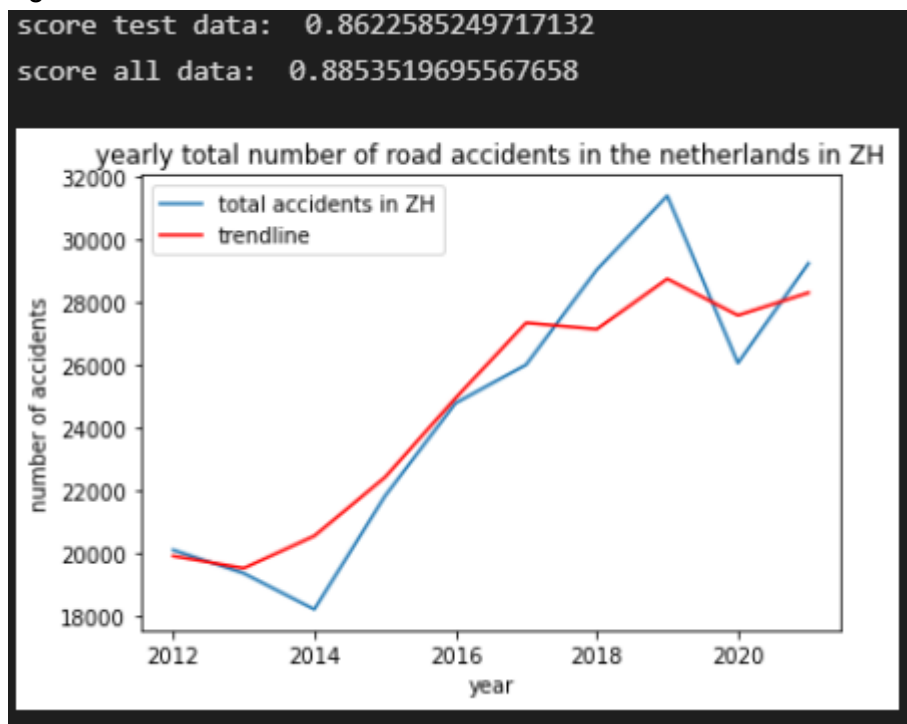
To apply linear regression I used the province Zuid-Holland.

However, until this point I haven't split the data into test and train data yet. To properly apply linear regression I split my data into test and train data. Because I only have 10 data points per province I split the data into 40% test data and 60% train data so that I do have some more test data. I trained my model with the train data and tested it with both just the test data and all the data. To score my models I used R^2 . For this metric the score needs to be as close to 1 as possible.

The features sets of features i chose to go forward with are:

- Feature set 1: Ages 15-20, age 65-80 and 80 and older

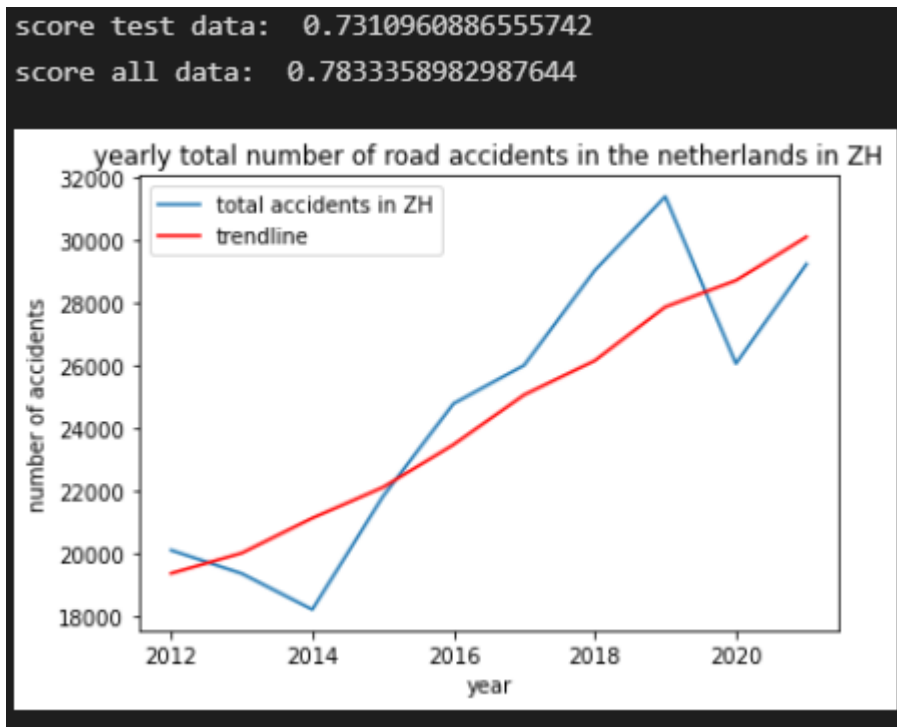
Figure 23



-

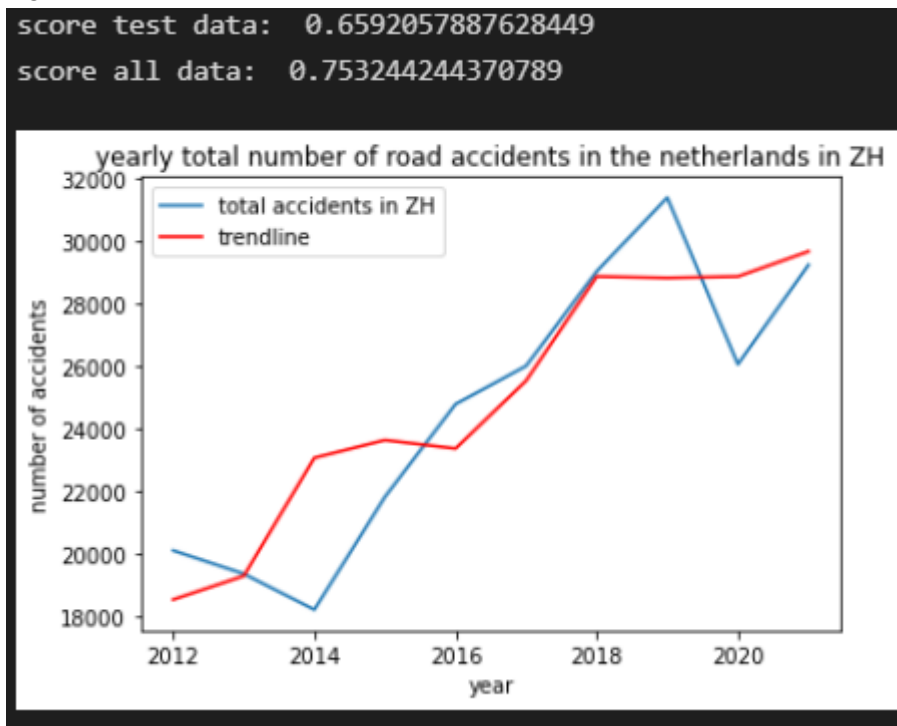
- Feature set 2: From the correlation matrix: Mediocre urbanity (the middle one of urbanity) and population density

Figure 24



- Feature set 3: From the correlation matrix: Total population and population density

Figure 25



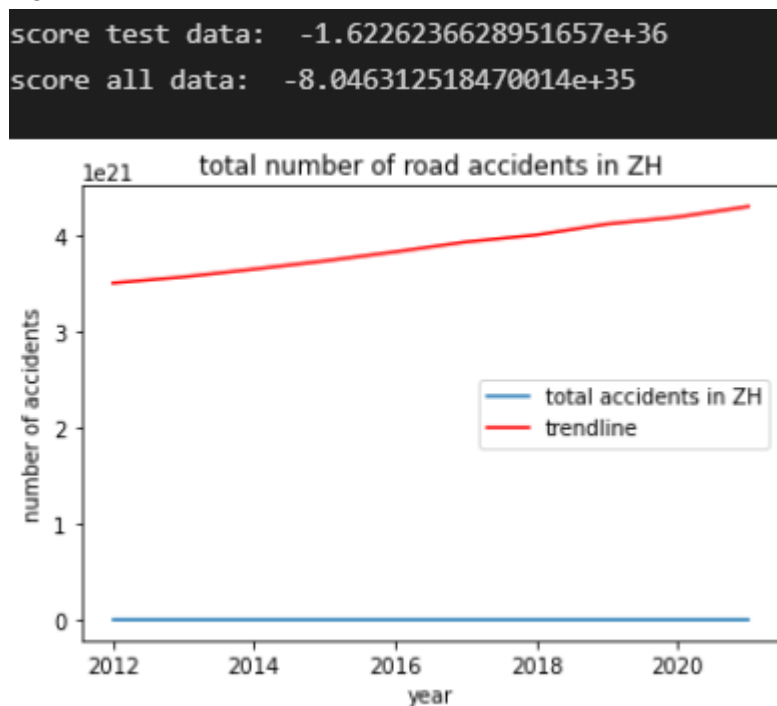
The score with all data stays stable for the most part, but with just the test data the score keeps changing significantly. This could be because the test data is only 4 data points. This is why I also test it with all the data.

Using different types of regression

To compare different types of regression I decided to try two more regression models. SGD regression and ridge regression.

Sgd Regression is a regression model which uses an iterative method called stochastic gradient descent to get close to the correct answer.

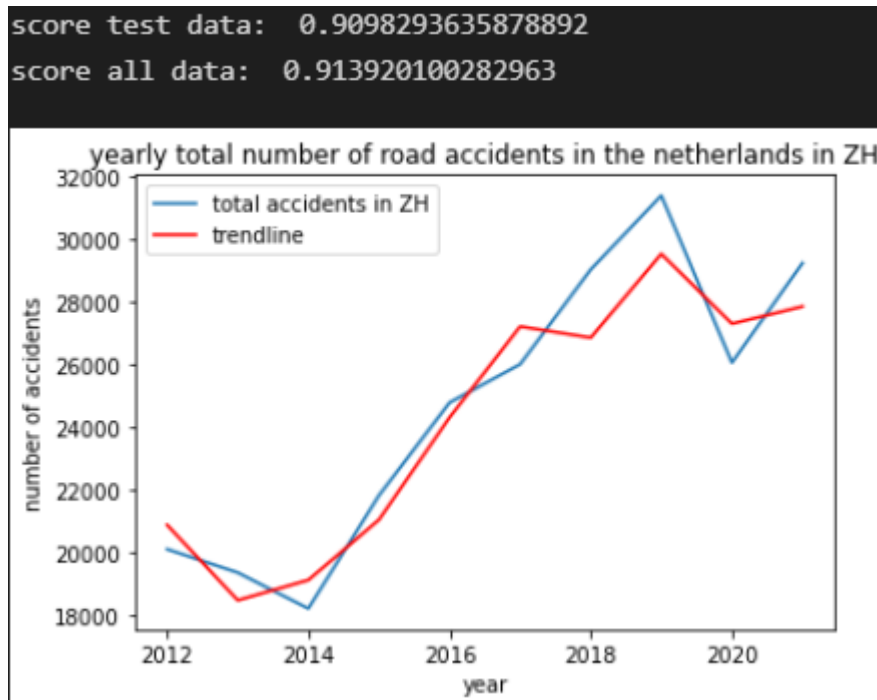
Figure 26



This however didn't bring in any results so I decided to look for another model. The model I found was Ridge regression. Ridge regression is a regression model which focuses on situations where the features are highly correlated with each other, which is the case in this project.

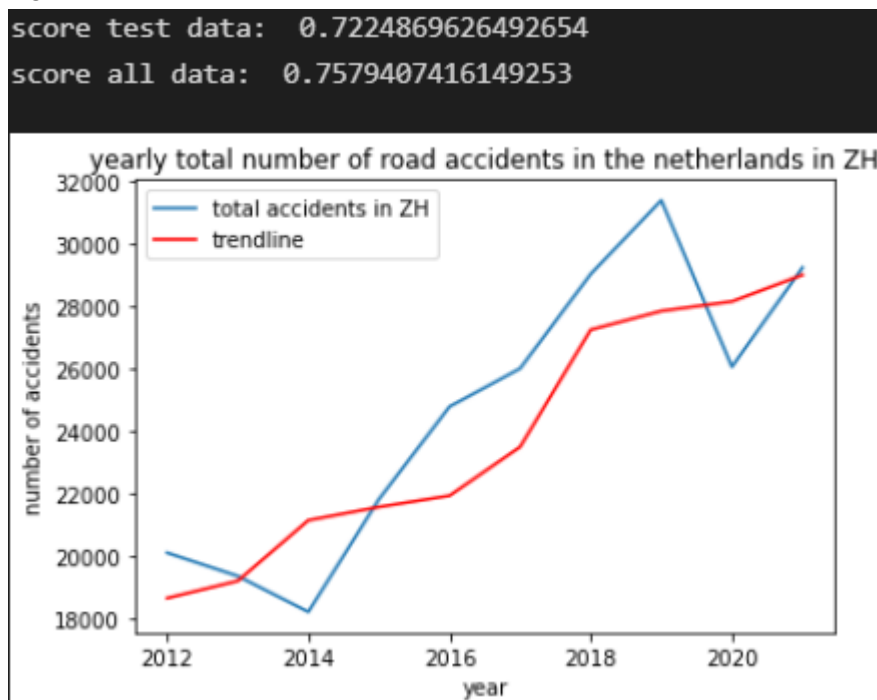
- Feature set 1: age 15-20, age 65-80 and age 80 and up

Figure 27



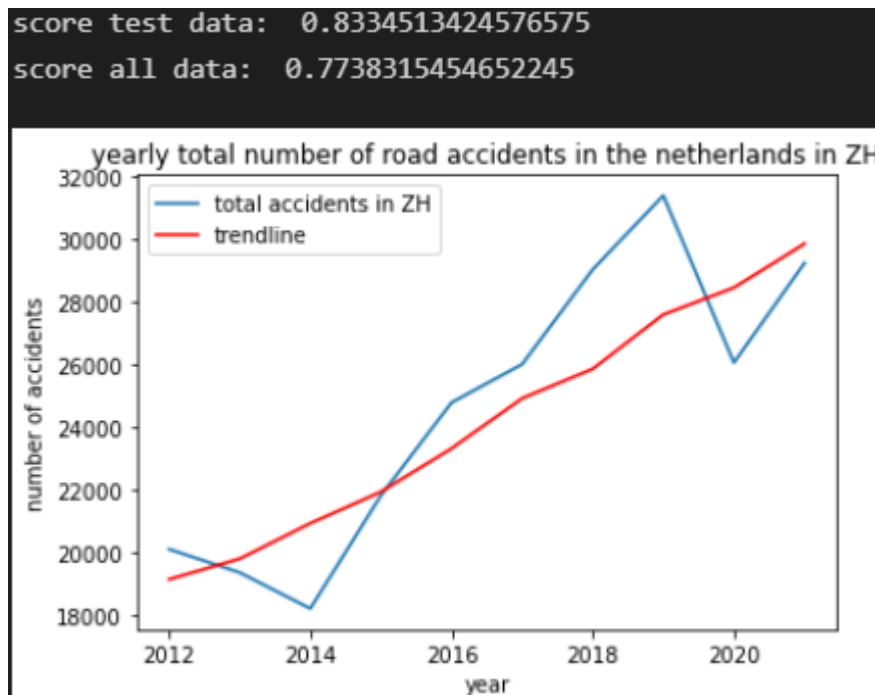
- Feature set 2: Mediocre urbanity and population density

Figure 28



- Feature set 3: Total population and population density

Figure 29



With ridge regression. Although the results of the scores of the test data are slightly more stable, they are still not consistent enough. This could mean I need more data.

Conclusion

I prepared two datasets and combined them to use them in the project. I also plotted graphs along the way to visualise the results. Using the data I applied linear regression, SGD regression and ridge regression. I compared these results.

While I have tried multiple linear regression. The R^2 scores are less than ideal. I have also tried different types of regression, namely SGD regression and ridge regression. SGD regression did not work in this project. Ridge regression has slightly more stable results than linear regression. However they are not stable enough that they could be used confidently. This could be because the data used is too small to train a model on.

Future recommendations

- Instead of splitting the data between test and train data of one province. I could use whole provinces to test and train.
- Adding 2022 to the datasets since the end of the project is in 2023.
- The code that makes a dataset which contains the amount of accidents for each province for each year can be improved on, which makes it way more future proof.