**Report project 1: Healthcare**

Nils Moerman

20/03/2023

**Predict a lifespan!**

**Introduction:**

* **Assignment**
* Project
* Learning goals
* Personal goals
* **Gathering the data.**
* How I gathered the data.
* **Analysing and transforming the data**.
* Analysing the data
* Transforming the data
* Building a pipeline
* **Building and training the model**
* Testing different models
* Building and training two models
* **Creating the interface**
* Creating the final interface
* **Overall experience and thoughts**
* Positives
* Negatives
* What did I learn?
* Conclusion

**Assignment:**

Project

The assignment was to create a program for doctors that can predict the lifespan of a patient. The assignment is **not** realistically accurate, but we have to pretend it is.

Learning goals

The main focus of this assignment is analysing and transforming the data. We have to learn how to look into the data and transform it. This is so we can recognise faulty data and change or remove it. We also must be able to recognize the outliers in the data and make choice whether we need to do something with this data.

Personal goals

1. I want to do things right and work structured. In the past I found it hard to work structured, this was mostly because I wasn’t interested in the things I was learning or working on.
2. I want to be able to figure stuff out by myself. I’m not afraid to ask when I need help, but I want to try to do it myself and don’t immediately ask when it gets hard.
3. I want to do things efficiently. This goes hand in hand with working structured, I want to use my time well and not dwindle on stuff that won’t change the result much.

**Gathering the data**

How I gathered the data

In this assignment we were given multiple datasets, the first dataset we had to pull from an API. We got a local server which we could access with a local URL to pull data from.

The second dataset we got was inside a database and had get it from there. This is the data we ended up using for the program as instructed by the teachers.

1. API data

***for this I would like to reference the getting\_data\_from\_url.ipynb notebook.***

***Location: notebooks/getting\_data\_from\_url.ipynb***

As can be seen in the notebook file, I’ve created a function that extracts the data (as json) from an API using requests and puts this into a variable. After checking the API manually, I noticed it had more pages and after doing some test runs I found that the different page URL’s were located in the “*next*” header. With that I mind I built this function.

After that I convert it to a data frame using the library pandas. I’ve discovered that I really like working with pandas! It makes working with the data more manageable for me. I can save a pandas data frame as CSV, so I won’t have to pull it from the API in the future.

1. SQL Database data (sqlite3)

***While getting this data I was already further ahead with the old data and didn’t use it in my data\_collection.ipynb notebook. I did however use it in my pipeline, so for this I would like to reference the first 31 lines in the pipeline.py file***

***Location: build/pipeline.py***

To be able to work with this data I first had to establish a connection with the database. To do this I had to establish a path, I put the path in a variable for generalisation purposes (someone else can easily change the path if they want to use it). After establishing a connection, I used pandas to read the right SQL table and convert it to a pandas data frame. I put this data frame into a variable for further use.

**Analysing and transforming the data**

***For this I would like to reference the data\_collection.ipynb notebook and the pipeline.py script.***

***Location notebook: notebooks/getting\_data\_from\_url.ipynb***

***Location script: build/pipeline.py***

Analysing the data

*Please look at the data\_collection.ipynb notebook for the step by step explanation.*

Transforming the data

*Please look at the data\_collection.ipynb notebook for the step by step explanation.*

Building a pipeline

*Please look at pipeline.py for reference*

I built a pipeline to transform my data.

To build the pipeline, I implemented the code I use in the data\_collection.ipynbnotebook but I leave the analysing part out. Since this pipeline is supposed to be used after analysing the data, I don’t need the analysing part and can leave out things like: creating plots, double checking if what I did worked, seeing what changed etc.

While building this pipeline and creating the “bmi” section I realised I didn’t create a failsafe for when the value of length is 0 (it’s impossible to divide by 0 and the program could crash). To solve this I created a for-loop that only does the calculation if length is bigger than 0.

In this pipeline I load the data from the database as explained in “how I gathered the data”.

Then I transformed it to the 2 data frames I want, like I do in the data\_collection.ipynb notebook. After that I save both data frames as new tables in the database. After that I can start testing with both tables.

**Building and training the model**

Testing different models

**OPTIONAL: While testing different regression models I’ve created notebooks trying them out, these are extra.**

**Locations:**

**notebooks/extra\_notebooks/decisiontree\_regression\_test.ipynb**

**notebooks/extra\_notebooks/random\_forest\_regression\_test.ipynb**

The 3 different models I tested were:

* + Decision tree regression
  + Random forest regression
  + Linear regression

After testing all 3 I ended up choosing the linear regression model, all 3 models were pretty accurate but the linear model was slightly better.

Building and training 2 models

**For the step by step explanation, please use the build\_model\_stepbystep.ipynb notebook**

**Location: notebooks/build\_model\_stepbystep.ipynb**

**For the build model, pleas use build\_model.py script**

**Location: builds/build\_model.py**

While analysing and transforming the data I already realised I wanted to create 2 models. One with outliers and one without outliers. I wanted this so I can test both and see which one performs best. To explain how I built and trained the models I made a notebook explaining it step by step. Please check out my build\_model\_stepbystep.ipynb where I explain the steps (location above).

**Creating the interface**

Creating the final interface

**For reference please check the interface.py script**

**Location: run/interface.py**

Now that I have a trained model, I can start building an interface. I started with a simple input function where the program asks each value and puts it in a dictionary so I can later convert it to a pandas data frame. While testing this function I realised I needed to make it so when users put in the wrong thing (like letters instead of numbers or crazy values that are impossible) it recognizes it’s wrong and tells the user what it needs and makes them try again. Later I was notified that the users aren’t supposed to have unlimited tries. To do this I created a while loop that first checks if the input is a number, if it’s not it will add a try to a counter and then check that counter. If the counter is 3 the program will send a message and shut down, if it is less than 3 the program lets you try again, so 3 is the amount of tries.

Then the program will check if the value of your input is within a certain parameter, if not: same thing like before, +1 to the counter and check.

If the input is correct the program will put the input into a dictionary, puts the counter back to 0 and does the same thing with the next value.

After the program has received all the values it will then convert the dictionary to a pandas data frame and make its prediction.

**Overall experience and thoughts**

Positives

* I was able to work structured and efficiently.
* I was able to find a solution for most of my problems without needing help from a colleague or instructor.
* We got a lot of time to analyse and transform the data.

Negatives

* I had some confidence issues at the beginning, which led me to keep looking for confirmation to check if I did the right thing.
* It was a group project but didn’t feel like one, since we had to make our own programs and the educational level’s felt divided.

What did I learn?

* I learned that I prefer programming and creating something over analysing and transforming data.
* I learned how to properly search for a solution online.
* I learned from my mistakes.
* I learned a lot of muscle-memory things that will help me in the future (like keyboard shortcuts etc.).
* I learned how to analyse data.
* I learned how to transform data.
* I learned how to build and implement a regression model.
* I learned how to program in python and a lot that comes with it (libraries, scripts and notebooks, databases, visual studio code, etc).
* I learned to be confident in my work and that I can do it.
* AND SO MUCH MORE!

Conclusion

I’m happy with the work I’ve done and the things I’ve learned. I learned from my mistakes and made something I’m proud of.