INDIVIDUAL ASSIGNMENT

BIG DATA

Shifaz Amier

BD3C

INHOUDS
OPGAVE

1. SAMENVATTING

In dit verslag word er een programma gemaakt die door middel van verschillende classifiers bepaald of een hotel review positief of negatief is. Er worden hotel reviews van het internet afgehaald en er word gewerkt met een grote data set waarin positieve en negatieve reviews staan. Na het filteren van de dataset werden er drie classifiers gebruikt. Naive Bayes Multinomial, Logistic Regression en Linear SVM. Het programma werkt succesvol en Logistic Regression had de hoogste accuracy.

2. SCRAPING

Voor het scrapen heb ik gebruik gemaakt van de website tripadvisor. Ik heb gekozen om zoveel mogelijk dynamische code te maken zodat ik dit script bij elk hotel kan draaien. De loop gaat door de pagina's voor het Hotel Blyss. Ik heb voor dit hotel gekozen omdat ik de reviews erg gemixt zijn. Voor het scrapen maar ik gebruik van Beautiful Soup.

```
for x in range(scrapePages):
    url = f'https://www.tripadvisor.com/Hotel_Review-g188590-d232457-
Reviews-or{pageNummer}-Hotel_Blyss-
Amsterdam_North_Holland_Province.html#REVIEWS'

    r = requests.get(url)
    soup = BeautifulSoup(r.content, 'lxml')
```

Op tripadvisor worden er 5 reviews per pagina gegeven.

```
for i in range(5):
        try:

            print(i + 1)

            hotelName = soup.select('._1mTlpMC3')[0].text
            review = soup.select('.IRsGHoPm')[i].text
            reviewTitle = soup.select('.glasR4aX')[i].text
            reviewerName = soup.select('.ui_header_link')[i].text
            reviewScore = scoreCheck(str(soup.select('.nf9vGX55')[i]))
            isBadReview = badReview(reviewScore)
```

Het programma loopt door elke van de 5 reviews die er zo uit zien.

De pagina loop zit in een try catch. Als het programma alle pagina's is af geweest zal het stoppen.

```
except:
            print("page not found")
            break
```

Door dit is het zelfs mogelijk om de volledige tripadvisor website af te scrapen als er een extra loop in het script word gezet om alle hotels af te gaan.

De score word aangegeven door een aantal bolletjes en niet door een bepaald getal. Dit is opgelost door een aparte functie te maken

```
def scoreCheck(reviewScoreData):
    if 'ui_bubble_rating bubble_50' in str(reviewScoreData):
        return 5

    elif 'ui_bubble_rating bubble_40' in str(reviewScoreData):
        return 4

    elif 'ui_bubble_rating bubble_30' in str(reviewScoreData):
        return 3

    elif 'ui_bubble_rating bubble_20' in str(reviewScoreData):
        return 2

    elif 'ui_bubble_rating bubble_10' in str(reviewScoreData):
        return 1
```

Er word gekeken of het een bad review is aan de hand van de score. 1 en 2 zijn slecht, 3 is neutraal en 4 en 5 zijn goed. De score word bijgehouden in een kolom genaamd "is_bad_review" waarin scores van een 1 en 0 staan. Om de neutrale te onderscheiden van de rest zijn die gelabeld met een 2.

Alle reviews worden vervolgens in en dataframe gezet. Deze df word dan in een csv gezet met de pandas functie "to_csv".

```
df.to_csv(r'hotelReviewscrape.csv')
```

	column 1	column 2	column 3	
1		hotel Name	is bad review	review
2	1	Hotel Blyss	1	Room dated, TV not working. Very noisy, all night banging, crashing, people talking lou
3	2	Hotel Blyss	1	This hotel has to be the worst hotel that I have ever stayed in! Firstly, the rooms that ar
4	3	Hotel Blyss	1	Where to start. Staff all very nice, no complaints. The room was terrible. Window woul
5	4	Hotel Blyss	1	Here are some facts and you will understand my reaction: (1) Room hygiene is below th
6	5	Hotel Blyss	1	The room was nothing close to what it looked like on the pictures. Located on the 3rd f
7	6	Hotel Blyss	1	Before anyone jumps on me ... I have travelled extensively around Europe and currently
8	7	Hotel Blyss	1	We recently stay at this hotel for 3 night for my birthday. It's safe to say we won't be st
9	8	Hotel Blyss	1	Arrived with my partner in Amsterdam on 14/03 as a gift for his birthday for three nigh
10	9	Hotel Blyss	0	The hotel is ok, clean and has all basic amenities. The rooms are cleaned daily which is
11	10	Hotel Blyss	0	Staff are super friendly so 100% good impression on arrival. Downstairs single room ha
12	11	Hotel Blyss	2	This hotel is ideal for a city break but don't expect anything more than basic. No fridge
13	12	Hotel Blyss	1	My reservation was cancelled for no reason on the day of my arrival while I was waiting
14	13	Hotel Blyss	1	If I could rate this a 0, I would. Not quite sure where to begin. We checked in just fine a
15	14	Hotel Blyss	2	We stayed here on a couples holiday from the 2nd - 6th November 2018. It is a very ni
16	15	Hotel Blyss	2	We stayed in the suite, which on first appearance looked luxurious, but after unpacking
17	16	Hotel Blyss	1	Don't think I have ever paid so much for so little. Tiny room, shower awful. Mattress tha
18	17	Hotel Blyss	0	The atmosphere was nice. It was few minutes walk to the tram stops. I stayed for a nigl
19	18	Hotel Blyss	1	First do not believe anything you see on their add. Check for facebook reviews mostly.
20	19	Hotel Blyss	1	We stayed at hotel Blyss for 3 days. When you get in to the hotel it looks nice, but, whe
21	20	Hotel Blyss	1	Three nights in this " hotel" was quite sth. First of all the location is genial for my intere
22	21	Hotel Blyss	1	A very bad hotel, everything smelled in the room, the staff was disgusting, the hotel ch
23	22	Hotel Blyss	0	This hotel in quiet area. 5/10 minute walk to all the action Room is plain and simple C
24	23	Hotel Blyss	2	Having spent a couple of days at The Hotel Blyss and in line with many other reviews o
25	24	Hotel Blyss	1	Bath was not clean. Hair stuck on the wall, mold on bath s door and on room's sealing.
26	25	Hotel Blyss	1	Check in was late by 20 minutes which made us late for our meeting. It is not possible t
27	26	Hotel Blyss	1	We (a group of 4) stayed in this hotel for a weekend and it seemed a long time for such

3. DATA PREPARATION

Ik had de Kaggle dataset gedownload en geopend in pandas. De dataset had vele onnodige kolommen.

```
list(df.columns)


['Hotel_Address',
 'Additional_Number_of_Scoring',
 'Review_Date',
 'Average_Score',
 'Hotel_Name',
 'Reviewer_Nationality',
 'Negative_Review',
 'Review_Total_Negative_Word_Counts',
 'Total_Number_of_Reviews',
 'Positive_Review',
 'Review_Total_Positive_Word_Counts',
 'Total_Number_of_Reviews_Reviewer_Has_Given',
 'Reviewer_Score',
 'Tags',
 'days_since_review',
 'lat',
 'lng']
```

Ik had eigenlijk alleen maar de positieve en negatieve reviews nodig. Ik besloot om 2 aparte dataframes te maken. Een positieve en een negatieve. De positieve labelen we met 0 en de negatieve met 1.

```
negative_df = df[['Negative_Review']]
negative_df["is bad review"] = 1


positive_df = df[['Positive_Review']]
positive_df["is bad review"] = 0

negative_df = negative_df.rename(columns={"Negative_Review": "review"})
positive_df = positive_df.rename(columns={"Positive_Review": "review"})

frames = [positive_df, negative_df]

df = pd.concat(frames)
```

	review	is bad review
0	Only the park outside of the hotel was beauti...	0
1	No real complaints the hotel was great great ...	0
2	Location was good and staff were ok It is cut...	0
3	Great location in nice surroundings the bar a...	0
4	Amazing location and building Romantic setting	0
...	...	...
515733	no trolly or staff to help you take the lugga...	1
515734	The hotel looks like 3 but surely not 4	1
515735	The ac was useless It was a hot week in vienn...	1
515736	No Negative	1
515737	I was in 3rd floor It didn t work Free Wife	1

1031476 rows x 2 columns

Het nadeel hiervan is dus dat de eerste helft alleen maar positieve reviews zijn en de tweede helft negatieve reviews. Als ik volledig gebruik maak van alle rows is dit geen probleem maar zodra ik maar 10.000 wil word dit wat moeilijker.

Het beste was dus om de df te shufflen. Ik kwam erachter dat sklearn hier een functie voor had. Shuffle(df) shuffled alle rows en daarna reset ik de index.

```
from sklearn.utils import shuffle
df = shuffle(df)
df.reset_index(inplace=True, drop=True)

test = df.head(100_000)
```

output:

	review	is bad review
0	They are replacing the carpets which are need...	1
1	No Negative	1
2	The hotel had a nice environment was clean an...	0
3	The position of the bed meant that it was so ...	1
4	No Negative	1
...	...	...
1031471	Staff was very helpful	0
1031472	Smallish characturful hotel very good locatio...	0
1031473	Lovely hotel	0
1031474	No Negative	1
1031475	Staff not friendly staff also struggled with ...	1

1031476 rows × 2 columns

Nu ik alle rows heb kan ik beginnen met het cleanen van de reviews.

```
def clean_text(text):
    text = text.lower()
    text = ''.join(i for i in text if not i.isdigit())
    tokenizer = nltk.RegexpTokenizer(r"\w+")
    new_words = tokenizer.tokenize(text)
    sentence = " ".join(new_words)
    lem =  lemmatize_sentence(sentence)
    text = [t for t in lem if len(t) > 1]
    text = " ".join(text)
    return(text)
```

de cleaning functie zorgt voor het volgende:

- *Alle tekst lowercase maken*
- *Alle nummers uit de tekst halen*
- *Alle punten, emoji's en ongewoonlijke spaties etc weghalen*
- *Lemmetizen van de tekst*
- *String returnen*

Ik had hiervoor de .apply functie gebruikt van pandas. Het grote nadeel van deze functie is dat deze geen progress laat zien.

```
df["review_clean"] = df["review"].apply(lambda x: clean_text(x))
```

Na onderzoek te doen kwam ik er achter dat tqdm gebruikt kan worden met pandas en dat .progress_apply gebruikt kan worden.

```
tqdm.pandas()
df["review_clean"] = df["review"].progress_apply(lambda x: clean_text(x
))
```

Hiermee word de progressie laten zien van de functie. (dit voorbeeld hieronder is op een later moment gedaan daarom zijn er 867k rows in plaats van 1 miljoen)

```
df["review_clean"] = df["review"].progress_apply(lambda x: clean_text
(x))

  4%|▌          | 32740/867640 [00:20<06:18, 2203.00it/s]
```

De dataframe ziet er nu zo uit

	review	is bad review	review_clean
0	The floors in the rooms were very creaky and ...	1	the floor in the room be very creaky and noisy...
1	Restaurant could be better	1	restaurant could be good
2	Very cozy well located hotel and with an amaz...	0	very cozy well locate hotel and with an amazin...
3	building and design	0	building and design
4	The breakfast was excellent and the staff wer...	0	the breakfast be excellent and the staff be mo...
...	...	...	...
1031471	Just dont like shower and toilets cleaning it...	1	just dont like shower and toilet clean it not ...
1031472	No Negative	1	no negative
1031473	Helpful staff	0	helpful staff
1031474	The small room is very small It s enough for ...	1	the small room be very small it enough for sin...
1031475	I liked the size of the room it was pretty bi...	0	like the size of the room it be pretty big the...

1031476 rows × 3 columns

De cleaning functie uitvoeren op 1 miljoen rows duurt erg lang daarom heb ik ervoor gekozen om deze df in een csv te zetten met de pandas to_csv functie.

```
df.to_csv(r'newHotelReviews.csv')
```

De dataframe zit vol met “No Positive” en “No Negative”’s deze moeten weggehaald worden.

```
to_drop = ['no negative','no positive']
df = df[~df['review_clean'].isin(to_drop)]
```

867551 rows × 3 columns

Daarna worden alle strings die bestaan uit 1 woord weggehaald.

```
count = df['review'].str.split().str.len()
df = df[~(count==1)].copy()
```

814988 rows × 3 columns

Er zijn ook rows die alleen een spatie erin hebben.

```
df = df[~(df['review'] == " ")]
813956 rows x 3 columns
```

De twee kolommen worden omgezet in een string

```
df["review_clean"] = df["review_clean"].astype(str)
df["review"] = df["review"].astype(str)
```

	review	is bad review	review_clean
622	n a	1	nan
980	N a	1	nan
1348	n a	1	nan
1649	N A	1	nan
2153	N A	1	nan
...	...	...	...
812067	N a	1	nan
812138	N a	1	nan
812203	N A	1	nan
813167	N A	1	nan
813371	N A	1	nan

1890 rows x 3 columns

Er zijn ook rows waarin N A staat op verschillende manieren. Deze worden door de cleaning functie omgezet in nan. Deze rows moeten ook gedropt worden.

```
df = df[~(df['review_clean'] == "nan")]
```

De index word gereset.

```
df.reset_index(inplace=True, drop=True)
```

De scrape dataset moet nu nog bij de kaggle dataset gezet worden.

We voeren eerste de cleaning uit

```
dfTripadvisor["review_clean"] = dfTripadvisor["review"].progress_apply(
lambda x: clean_text(x))
```

```
82%|████████  | 264/320 [00:15<00:02, 19.54it/s]
```

Als dat klaar is maken we een nieuwe dataframe waarin alles staat dat we nodig hebben.

```
df = dfTripadvisor[['review', 'is bad review', 'review_clean']]
```

Alle reviews die in de scrape dataset gelabeld werden als neutraal (2) zijn er ook uitgehaald.

De dataframes moeten nu worden samengevoegd

```
df = dfTripadvisor[['review', 'is bad review', 'review_clean']]
frames = [dfKaggle, dfTripadvisor]
df = pd.concat(frames)
```

Nu moet de dataframe gezet worden in een sql database.

```
from sqlalchemy import create_engine

db_connection_str = 'mysql+pymysql://root:pw@host/DbName'
db_connection = create_engine(db_connection_str)
```

Hiervoor word de df to sql functie gebruikt.

```
df.to_sql('hotelReviews', con=db_connection, if_exists='replace',index_
label='id')
```

Nu kan de database worden opgehaald met een SQL statement

```
df = pd.read_sql("SELECT * from users", db_connection)
```

4. MODEL BOUWEN

Nu de dataset voorbereid is kunnen we beginnen aan het model.

Ik heb gebruikt gemaakt van TfidVectorizer

```
vectorizer = TfidfVectorizer(use_idf=True,lowercase=True,strip_accents=
'ascii')
```

De label word opgeslagen in de y, x krijgt de reviews die in de vectorizer.fit_transform zitten.

```
y = df['is bad review']
x = vectorizer.fit_transform(df['review_clean'])
```

De data word gesplit.

```
x_train, x_test, y_train, y_test = train_test_split(x,y,random_state =
604)
```

De classifier, naive bayes in dit geval word ingesteld en begint met trainen.

```
clf = naive_bayes.MultinomialNB()
clf.fit(x_train,y_train)
```

De accuracy word gemeten.

```
predicted = clf.predict(x_test)
accuracy_score = metrics.accuracy_score(predicted,y_test)
score  0.91751520973423
```

Als we de code opnieuw runnen maar dit keer met

```
clf = LogisticRegression()
```

krijgen we de volgende score.

```
score  0.9425453828912043
```

Met

```
clf = LinearSVC()
```

krijgen we een score van

```
score  0.9420232025419345
```

De 3 classifiers die zijn gebruikt:

4.1 MULTINOMIAL NAIVE BAYES

Vergeleken met Naive Bayes is Multinomial naive Bayes meer geschikt voor tekst documenten. Naive Bayes is meer afhankelijk aan of bepaalde woorden er zijn of niet. Multinomial Naive Bayes kijkt meer naar de word count en de aantallen of woorden er zijn of niet zijn. Zo word er meer context gegeven.

4.2 LOGISTIC REGRESSION

Logistische regressie is een classificatie-algoritme dat wordt gebruikt om de kans op succes en mislukking van een gebeurtenis te vinden. Het wordt gebruikt als de variabele binair is, dus een 0 of een 1. Het ondersteunt het categoriseren van gegevens in aparte groepen door de relatie van een bepaalde set gelabelde gegevens te bestuderen.

4.3 LINEAR SVM

SVM of Support Vector Machine is een lineair model voor classificatie- en regressieproblemen. Het algoritme creëert een lijn dat de gegevens in aparte groepen verdeelt. Het kan lineaire en niet-lineaire problemen oplossen en werkt goed voor veel praktische problemen.

5. RESULTS

Als we de scores van alle classifiers onder elkaar zetten. Dan scoort Logistic Regression het beste. Daarna Linear SVM en als laatste multinomial Naive Bayes. Het verschil tussen de eerste 2 is niet groot.

Logistic Regression:

```
score  0.9425453828912043
```

Linear SVM:

```
score  0.9420232025419345
```

Naive Bayes:

```
score  0.91751520973423
```

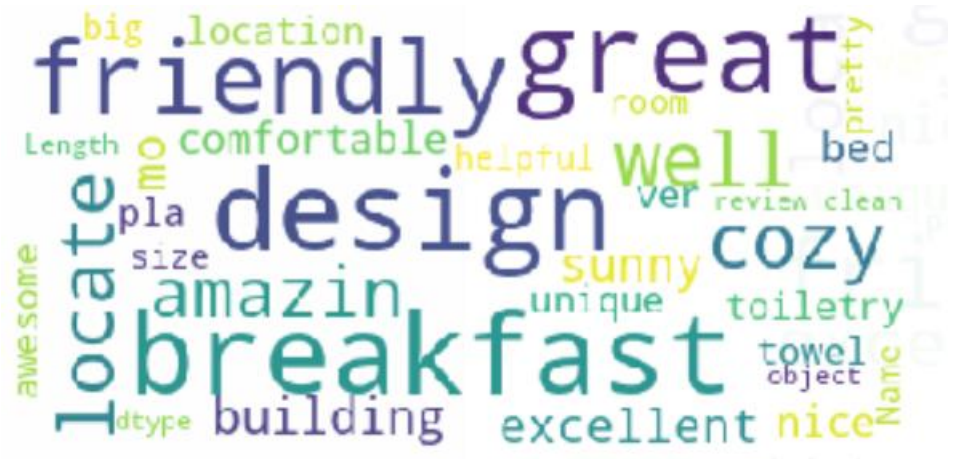
In de dataset heeft er wat cleaning plaats gevonden. Het is dus tijd om een wordcloud hiervan te maken. Laten we beginnen met de positieve reviews.

```
text = df.loc[df['is bad review'] == 0]
text = text['review_clean']

wordcloud = WordCloud(collocations=False, max_font_size = 50,background
_color='white').generate(str(text))
```

Ik heb Hierna nog een lijst met stopwoorden toegevoegd en nog een paar zelf toegevoegd.

```
from wordcloud import STOPWORDS
wordjes = ['staff','thick','hotel', 'first'] + list(STOPWORDS)
```



Nu de negatieve reviews.

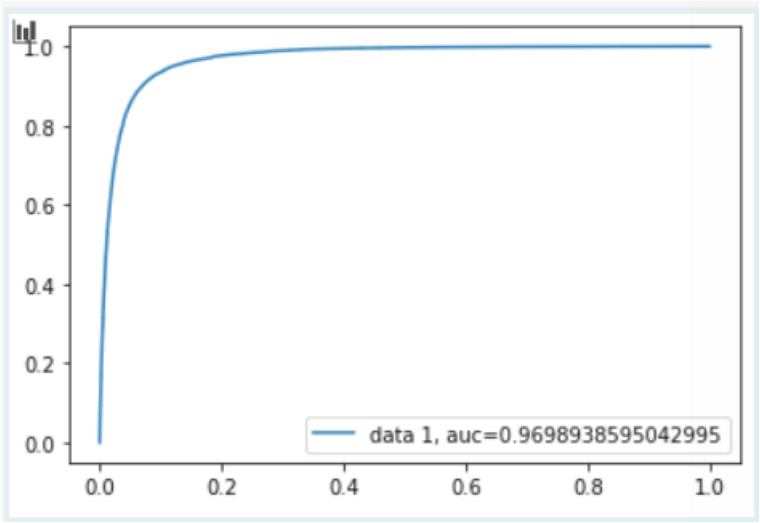
```
text = df.loc[df['is bad review'] == 1.0]
```

Area under the curve:

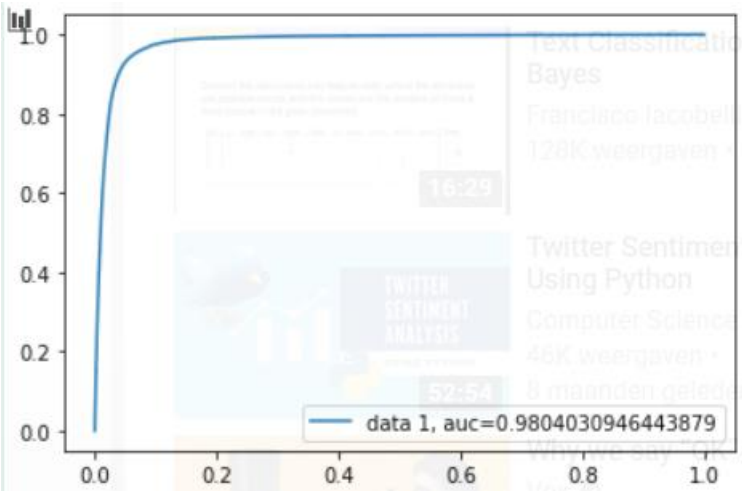
```
y_pred_proba = clf.predict_proba(x_test)[::, 1]
fpr, tpr, _ = metrics.roc_curve(y_test,  y_pred_proba)
auc = metrics.roc_auc_score(y_test, y_pred_proba)

plt.plot(fpr, tpr, label="data 1, auc="+str(auc))
plt.legend(loc=4)
plt.show()
```

Naïve Bayes:



Logistic Regression:



5.1 TESTEN MET EIGEN REVIEWS

Deze test heb ik hier gedaan met de naive_bayes.MultinomialNB() classifier. Ik heb de volgende code gebruikt:

```
testReviewArray = ["Loved this hotel it was great"]
testReviewinput = [clean_text(testReviewArray[0])]
testReview = vectorizer.transform(testReviewinput)
```

```
print(testReviewArray[0],"\n\n",testReviewinput,"\n")

if clf.predict(testReview) == 0:
    print("Good review")
elif clf.predict(testReview) == 1:
    print("Bad review")
else:
    print(clf.predict(testReview))
```

```
Loved this hotel it was great

 ['love this hotel it be great']
```

Deze review word gezien als goed!

```
if clf.predict(testReview) == 0:...

Good review
```

Nu testen met een negatieve review.

```
testReviewArray = ["Do not go to this hotel, everything was dirty!
the toilets, the rooms. The staff was also very rude."]...

Do not go to this hotel, everything was dirty! the toilets, the rooms. The
staff was also very rude.

 ['do not go to this hotel everything be dirty the toilet the room the sta
ff be also very rude']

Bad review
```

Met de logistic regression classifier heb ik geprobeerd om een negatieve review van het internet af te testen.

```
Such a gorgeous property but they they are rude every time and this time n
eglected to tell me construction would start at 6:30am - VERY loud constru
ction. Who does that without informing a guest? Never again

 ['such gorgeous property but they they be rude every time and this time n
eglect to tell me construction would start at be very loud construction wh
o do that without inform guest never again']

Bad review
```

Ik heb iemand anders gevraagd om een positieve review voor me te schrijven zodat ik nog een test uit kan voeren met LinearSVC.

```
I recommend this hotel, nice spacious rooms for a decent price. It really
felt like coming home after a long trip. I was kind of jet-lagged so I was
happy that the room was clean and ready to use. There was a wonderful smel
l in the room, that must be their signature sent! So homey, I loved it.

 ['recommend this hotel nice spacious room for decent price it really felt
like come home after long trip be kind of jet lag so be happy that the roo
m be clean and ready to use there be wonderful smell in the room that must
be their signature send so homey love it']

Good review
```

Het programma werkt dus goed met alle drie classifiers zover ik heb getest.

6. CONCLUSIE

Van alle classifiers blijkt Logistic Regression het beste te werken. Het verschil tussen de accuracy scores van Linear SVM zijn niet groot. Alle scores waren boven de 90% dus het model werkt goed. Dit heb ik getest door alle modellen op 3 verschillende manieren te testen. Daarnaast kloppen de wordclouds van de dataset. Dat betekend dat alle data correct gelabeld is. Ik heb enorm veel van deze opdracht geleerd. Nu ik weet hoe ik pandas en machine learning kan gebruiken zal ik dit nu ook vaker gebruiken voor persoonlijke projecten. Ik had niet verwacht dat ik in zo een korte tijd zoveel nieuwe skills op kon doen dus daar ben ik blij me.