```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly
remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
from zipfile import ZipFile
import numpy as np
import matplotlib.pyplot as plt
```

```
df0 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk0.jso
n')
df1 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk1.jso
n')
df2 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk2.jso
n')
df3 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk3.jso
n')
df4 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk4.jso
n')
df5 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk5.jso
n')
df6 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk6.jso
n')
df7 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk7.jso
n')
df8 =
pd.read_json('/content/drive/MyDrive/interview/dutch_tweets_chunk8.jso
n')
```

```
full_data = df0.append(df1)
full_data.append(df2)
full_data.append(df3)
full_data.append(df4)
full_data.append(df5)
full_data.append(df6)
full_data.append(df7)
```

```
full_data.append(df8)
print('total df shape is: ', np.shape(full_data))

total df shape is:  (54161, 23)

full_data.head(5)

                                           full_text  \
0  @pflegearzt @Friedelkorn @LAguja44 Pardon, wol...
1  RT @grantshapps: Aviation demand is reduced du...
2  RT @DDStandaard: De droom van D66 wordt werkel...
3  RT @DDStandaard: De droom van D66 wordt werkel...
4  De droom van D66 wordt werkelijkheid: COVID-19...

                                    text_translation
created_at  \
0  @pflegearzt @Friedelkorn @ LAguja44 Pardon wol... 2020-03-09
12:26:29
1  RT @grantshapps: Aviation demand is reduced du... 2020-03-09
12:26:34
2  RT @DDStandaard: The D66 dream come true: COVI... 2020-03-09
12:26:37
3  RT @DDStandaard: The D66 dream come true: COVI... 2020-03-09
12:26:37
4  The D66 dream becomes reality: COVID-19 super ... 2020-03-09
12:26:47

   screen_name                                        description  \
0  TheoRettich  I ❤science, therefore a Commie.   ☭ FALGSC: P...
1  davidiwanow  I tweet a lot but love to engage & converse. P...
2     EricL65                                                None
3     EricL65                                                None
4    EhrErwin  Budget-Life Coach. Time management Coaching. b...

                                    desc_translation  weekofyear
weekday  \
0  I ❤science, Therefore a Commie. ☭ FALGSC: Par...          11
0
1  I tweet a lot but love to engage and converse....          11
0
2                                               None          11
0
3                                               None          11
0
4  Budget-Life Coach. Time management coaching. h...          11
0

   day  month  ...                         point  latitude longitude
altitude  \
0    9      3  ...  (52.5001698, 5.7480821, 0.0)  52.50017  5.748082
```

```
0.0
1    9      3  ...  (52.3727598, 4.8936041, 0.0)  52.37276  4.893604
0.0
2    9      3  ...                          None       NaN       NaN
0.0
3    9      3  ...                          None       NaN       NaN
0.0
4    9      3  ...  (52.3727598, 4.8936041, 0.0)  52.37276  4.893604
0.0

        province  hisco_standard  hisco_code industry
sentiment_pattern  \
0      Flevoland            None        None    False
0.0
1  Noord-Holland            None        None    False
0.0
2          False            None        None    False
0.0
3          False            None        None    False
0.0
4  Noord-Holland            None        None    False
0.0

  subjective_pattern
0                0.0
1                0.0
2                0.0
3                0.0
4                0.0

[5 rows x 23 columns]
```

```python
for col in full_data.columns:
    print(col)
```

```
full_text
text_translation
created_at
screen_name
description
desc_translation
weekofyear
weekday
day
month
year
location
point_info
point
latitude
```

```
longitude
altitude
province
hisco_standard
hisco_code
industry
sentiment_pattern
subjective_pattern
```

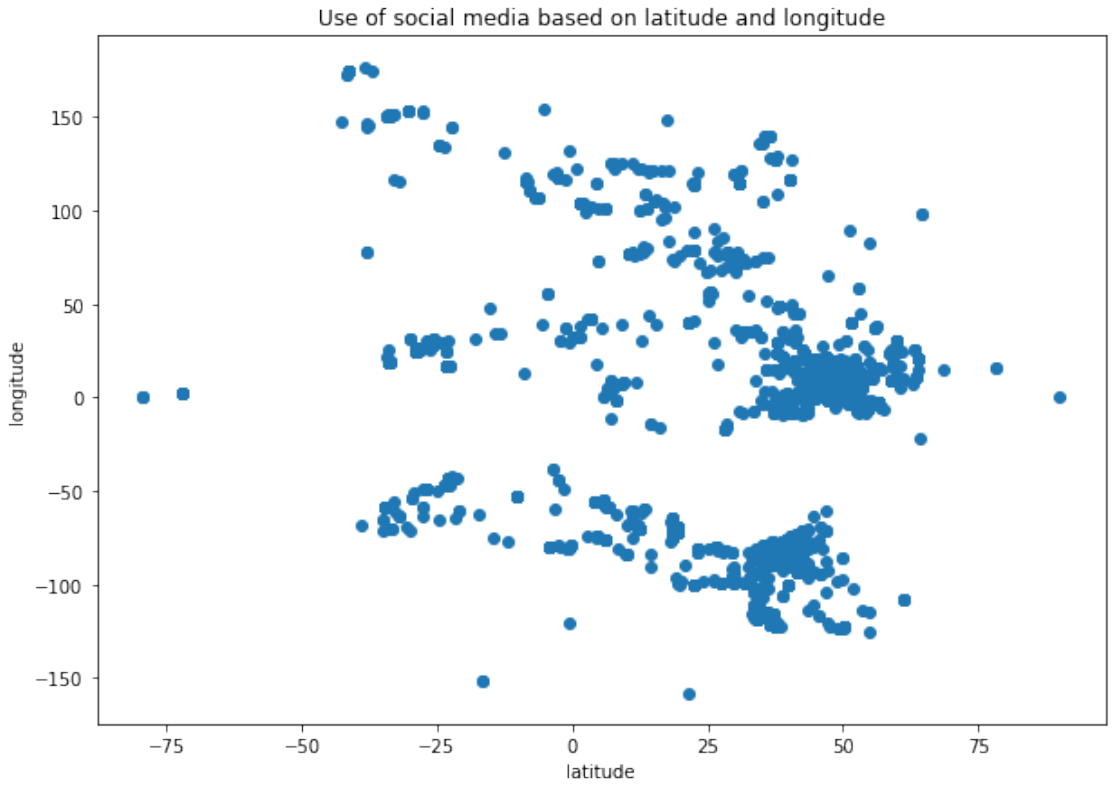
```
long_array = full_data['longitude'].array
lat_array = full_data['latitude'].array
is_long_nan = [long_array == 'NaN']
is_lat_nan = [lat_array == 'NaN']
full_data['is long nan'] = [full_data['longitude'].isnull()][0]
full_data['is lat nan'] = [full_data['latitude'].isnull()][0]
data_with_no_missing_loc = full_data.dropna(subset = ['latitude',
'longitude'])
print('shape of data with missing longitude and latitude: ',
np.shape(data_with_no_missing_loc))
print('total df shape is: ', np.shape(full_data))
print('we can notice that there are a lot of missing values for
longitude and latidude. We will still plot these points to see where
the hotspots of recorded social media use are ')
```

```
shape of data with missing longitude and latitude:  (26822, 25)
total df shape is:  (54161, 25)
we can notice that there are a lot of missing values for longitude and
latidude. We will still plot these points to see where the hotspots of
recorded social media use are
```

```
#now we can plot points of data based on latitude and longitude

plt.figure(figsize = (10,7))
latitudes = data_with_no_missing_loc['latitude'].array
longitudes = data_with_no_missing_loc['longitude'].array
plt.scatter(latitudes, longitudes)
plt.title('Use of social media based on latitude and longitude')
plt.xlabel('latitude')
plt.ylabel('longitude')
```

```
Text(0, 0.5, 'longitude')
```



What do we notice above? we notice some clustering of data. I will hopefully be able to use K-means clustering algorithsm to find wehre the majority of the hotspots come from. Upon first look of the data, I see that there are clusters around (lat, long) = (40, -90), (40, 10), and maybe something somewhere else however I do not know, so we will see what happens when we apply 3, 6, 12, and 27 nearest neighbors. Without knowing how many provinces there are, we will just make some guesses which incude these 3, 6, 12, 27. To do this I will be using K nearest neighbors from sklearn to try and identify the provice that each data point should belong to. WE will see how accurate this is!

```
data_with_no_missing_loc[['latitude', 'longitude']]
data_with_no_missing_loc['latitude']
no_missing_provinces_df =
data_with_no_missing_loc[data_with_no_missing_loc['province'] !=
False]
print('size of data frame containing false for some provinces: ',
np.shape(data_with_no_missing_loc))
print('size of data frame not containing false for some provinces: ',
np.shape(no_missing_provinces_df))
```

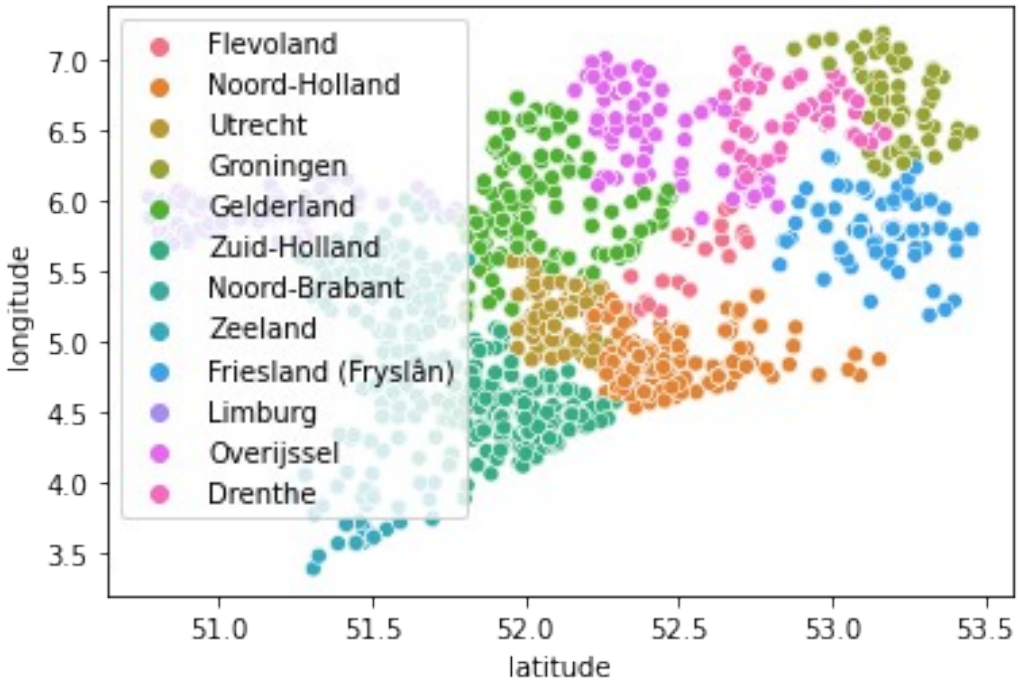
```
size of data frame containing false for some provinces:  (26822, 25)
size of data frame not containing false for some provinces:  (21443,
25)
```

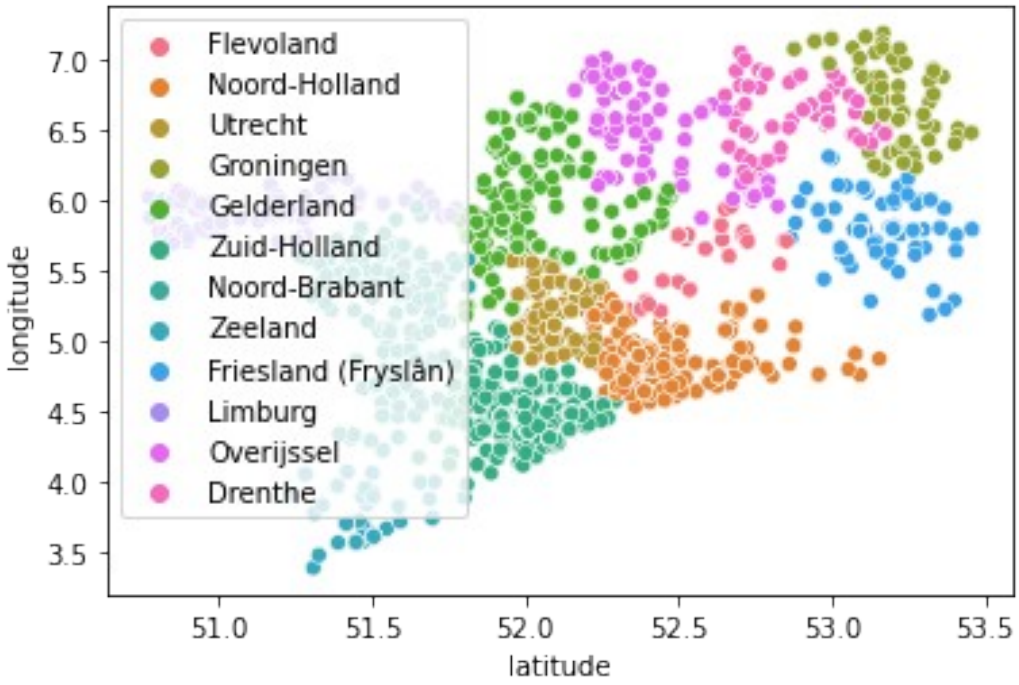
So it turns out that some of the provinces are also missing. With this being said, when we compare the size of the df without those missing provinces vs the size with the missing

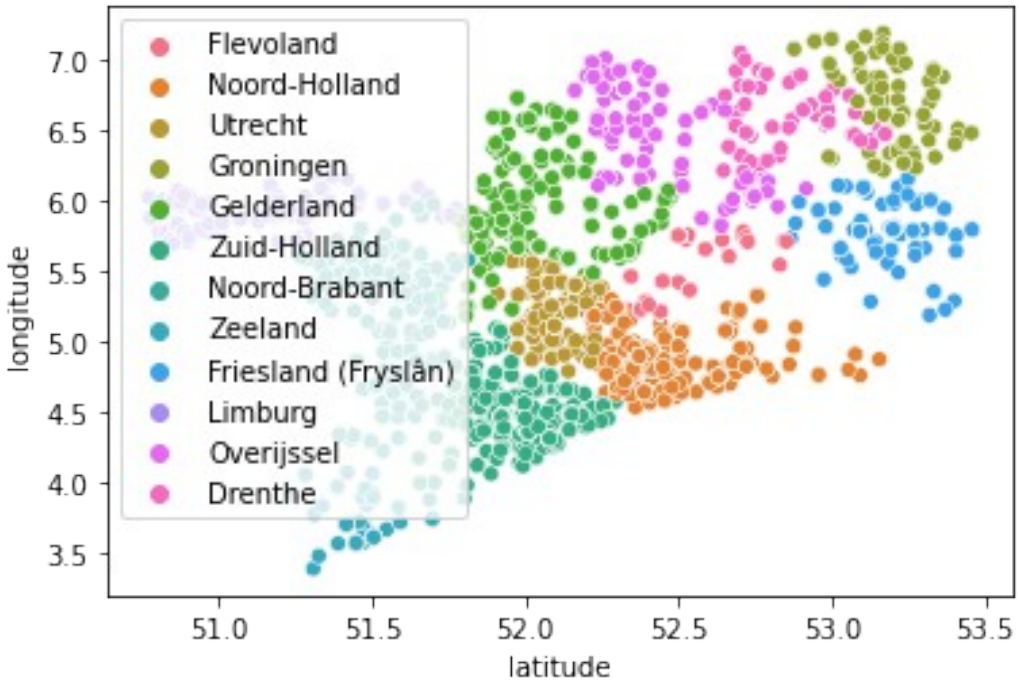
provinces, we are losing about 5000 points of data, but we will proceed without the missing data so the classification is able to run

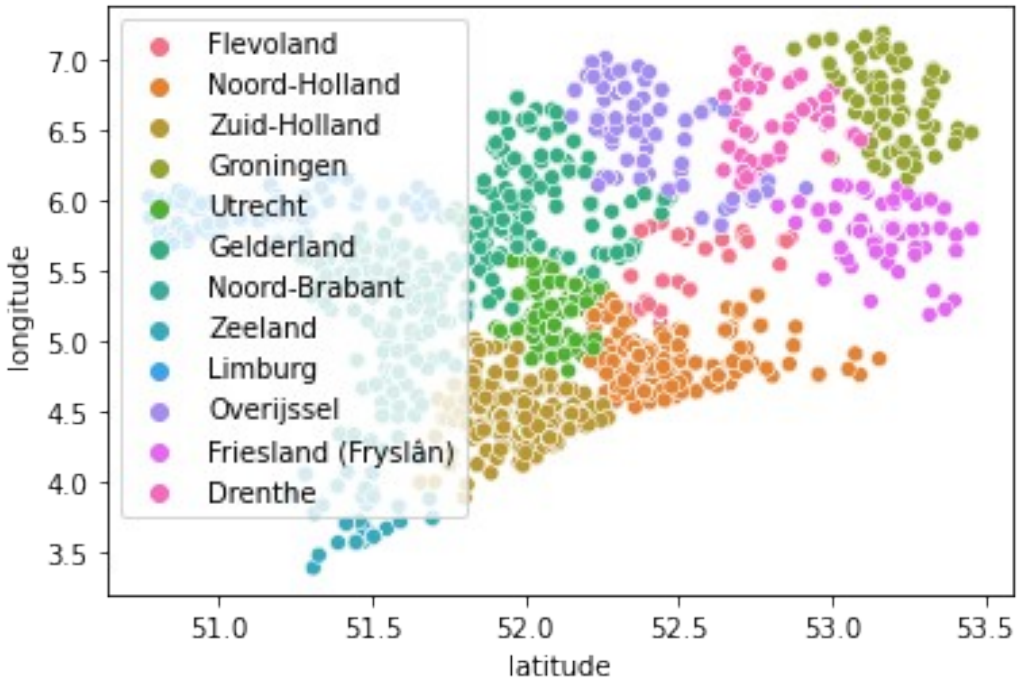
```
#we will run the sklearn nearest neighbor algorithm to categorize the
data from our training set using 4 different nearest neighbor sizes
import seaborn as sns
from sklearn import neighbors, datasets
n_neighbors = [3, 6, 12, 27]
array_of_fits = []
accuracy_score_on_train = []
final_latitudes = no_missing_provinces_df['latitude']
final_longitudes = no_missing_provinces_df['longitude']
for num_neighbors in n_neighbors:
  X_train = no_missing_provinces_df[['latitude', 'longitude']]
  y_train = no_missing_provinces_df['province']
  neigh = neighbors.KNeighborsClassifier(n_neighbors=num_neighbors)
  array_of_fits.append(neigh.fit(X_train, y_train))
  accuracy_score_on_train.append(neigh.score(X_train, y_train))
  ax = plt.subplots()
  sns.scatterplot(x = final_latitudes, y = final_longitudes, hue =
neigh.predict(X_train))
  plt.show
for i in range(4):
  print('the accuracy of our model for ', n_neighbors[i], ' is
supposed to be: ', accuracy_score_on_train[i])
```

```
the accuracy of our model for  3  is supposed to be:
0.999440376812946
the accuracy of our model for  6  is supposed to be:
0.9986009420323648
the accuracy of our model for  12  is supposed to be:
0.9974816956582567
the accuracy of our model for  27  is supposed to be:
0.9935176980832906
```


Flevoland
Noord-Holland
Utrecht
Groningen
Gelderland
Zuid-Holland
Noord-Brabant
Zeeland
Friesland (Fryslân)
Limburg
Overijssel
Drenthe
longitude
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
51.0
51.5
52.0
52.5
53.0
53.5
latitude


Flevoland
Noord-Holland
Utrecht
Groningen
Gelderland
Zuid-Holland
Noord-Brabant
Zeeland
Friesland (Fryslân)
Limburg
Overijssel
Drenthe
longitude
7.0
6.5
6.0
5.5
5.0
4.5
4.0
3.5
51.0
51.5
52.0
52.5
53.0
53.5
latitude





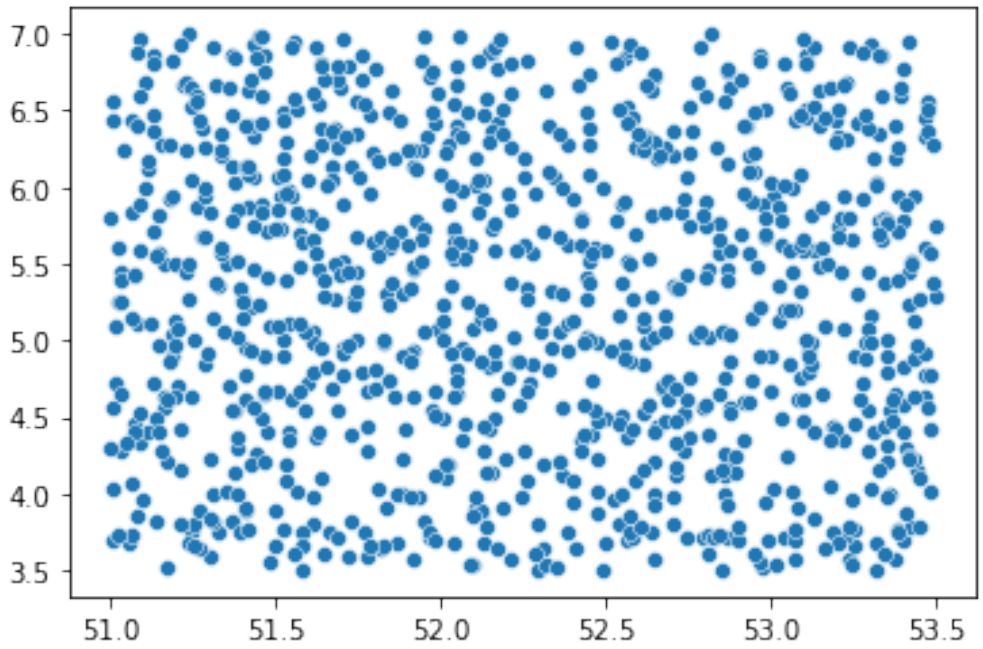
Ok so what do our plots show. They show that I can plot the data and categorize the data into their provinces when provinces are given. Not very informative about the model I have created. I need to find a way to create a testing data set, by creating tuples of points representing latitudes and longitudes for random uses of social media, and then categorize them. I did not need to duplicate the graphs, but I will leave it for now.

My next goal will be to test these 4 models with different neighbors on a test set.

```
#generating testing data using random uniform distributed between 3.5
and 7 for longitude and 51 and 53.5 for latitude
sample_lats = []
sample_longs = []
for i in range(1000):
  sample_lats.append(np.random.uniform(51, 53.5))
  sample_longs.append(np.random.uniform(3.5, 7))
sns.scatterplot(sample_lats, sample_longs)
print('this is our randomly generated data set for 1000 points')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7ff025b88290>
```



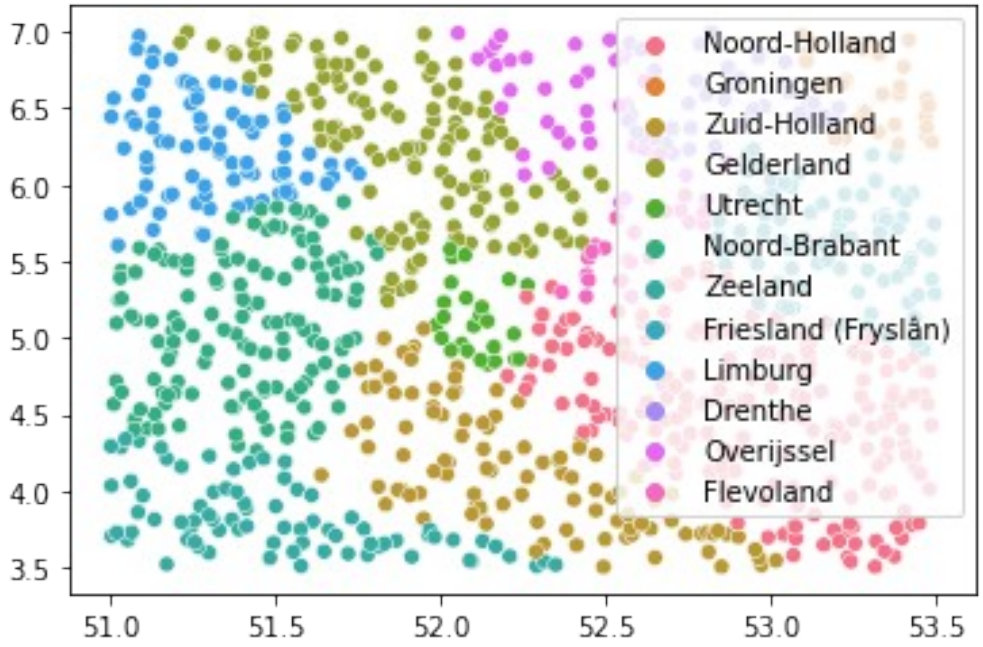

```
X_test = [sample_lats, sample_longs]
print(np.shape(X_test))
y_test = array_of_fits[1].predict(np.transpose(X_test))
sns.scatterplot(X_test[0], X_test[1], hue = y_test)
plt.show
```

```
(2, 1000)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/base.py:451:
UserWarning: X does not have valid feature names, but
KNeighborsClassifier was fitted with feature names
  "X does not have valid feature names, but"
```

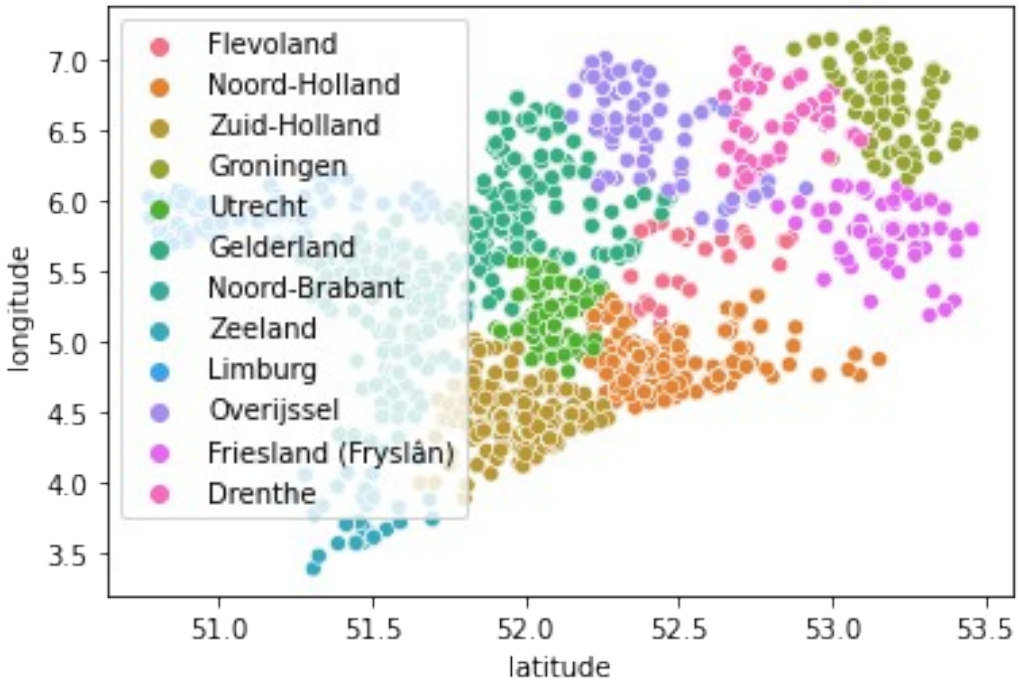
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
FutureWarning: Pass the following variables as keyword args: x, y.
From version 0.12, the only valid positional argument will be `data`,
and passing other arguments without an explicit keyword will result in
an error or misinterpretation.
  FutureWarning
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```




```
sns.scatterplot(x = final_latitudes, y = final_longitudes, hue =
neigh.predict(X_train))
plt.show
```

```
<function matplotlib.pyplot.show(*args, **kw)>
```



So when we compare graphs above, at least for our second model(because I ran out of time I could not try others) we see that it appears to have similar plt colors for each province. It is a little hard to see because I did not match the color and the legend is in the way. Unfortunatelly I could not get more plots to try the different models

What did I just find out, that all the false values were all the places that weren't in that original cluster that we saw around (50, 10). With this being the case we will make a new approch, maybe not having a training y. Maybe unsupervised learning will allow us to categorize these clusters better. We will not have information such as the name of the province, but we will gather a larger range of locational hotspots, which maybe with more time and resources(such as looking up the geographic lats and longs), we could categorize each province from the larger data set.