

Using Foursquare and eBay to analyze computer & hardware sellers in my area

Applied Data Science Capstone Project

Lukas Weidich

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Who will profit from the insights	1
2	Data	2
2.1	Foursquare	2
2.2	eBay	2
2.3	Additional information	2
3	Methodology	3
3.1	Retrieving the data	3
3.2	Handling missing information and combining the two datasets	3
3.3	Exploratory data analysis	7
3.4	Machine learning	9
4	Results, Discussion and Conclusion	14

1 Introduction

1.1 Motivation

As a customer, I am always overwhelmed by all the buying options I am confronted with on a daily basis. Especially when looking for computers and hardware, there are seemingly endless stores and possibilities just around the corner. While these stores are professional businesses, I wonder whether private offerings are also that strongly represented within my local area. To find out, I will make use of business data provided by Foursquare and compare it to currently active eBay listings near me. By also visualizing the data, I hope to underline my findings even more.

1.2 Who will profit from the insights

Both, buyers and sellers, can profit from the results portrayed in this short report. Buyers can use the data to further understand the available options to choose from when buying computers or hardware. Sellers can increase their competitive advantages by getting to know their surroundings, such as other stores or active listings nearby. Also, anyone interested in the vast field of data science is able to get an impression of what to expect from the IBM Data Science Professional Certificate.

2 Data

I will use two different datasets to gain more insights into the questions I posed above. One will be Foursquare, which was introduced during the course. The other data will be provided by eBay. I will download the eBay data and load the file from my local directory, so I do not need to expose my credentials within the coding. The Foursquare data however will be fetched within the code cells.

2.1 Foursquare

Foursquare will provide general information about professional sellers and businesses offering computers in my area. I will use the location (longitude, latitude, city name) to cluster the data. Additionally, the entries are visualized by being shown on a map. Besides the given columns, I will also one hot encode the type of business (professional or private), depending on the entry itself.

2.2 eBay

By making use of the eBay API, I can access and filter all eBay listings near me that are currently active. Similar to the Foursquare data, I am mainly interested in the location of the seller, the sellers name and whether the seller is a private person or a professional business.

2.3 Additional information

Of course, I will use the same search radius for both API calls, so I have a completely comparable dataset to work with. However, the Foursquare call is limited to 200 results.

3 Methodology

3.1 Retrieving the data

I can access the active eBay listings near me by fetching data recursively from the eBay API. I wrote a small seeder script that stores the data in csv format and saves it locally. From there, I can easily read the csv in my notebook. The Foursquare data will be loaded within the code cells.

3.2 Handling missing information and combining the two datasets

After fetching the data, it is time to combine and unify the datasets, so I can work with coherent data in the next steps. For my purposes, it will be sufficient to have the following columns for both datasets:

- Name of business/seller/store/item
- City
- State
- Latitude
- Longitude
- Source (eBay/Foursquare)

This is what the two datasets look like after fetching:

```
df_foursquare.head()
```

		Name	Latitude	Longitude	Source
0		Computer Service Dröge	52.279037	8.902915	Foursquare
1		TM Computer e.K.	52.116941302856596	8.67265034442815	Foursquare
2		TM Computer e.K. - Thomas Müller (Büro)	52.15775752422116	8.63429950693619	Foursquare
3		Schormann-Computer	52.18319156583845	8.874798774686496	Foursquare
4		4you-computer.de	52.24314880371094	8.840249061584473	Foursquare

Figure 3.1: Foursquare dataset right after fetching with some columns already removed

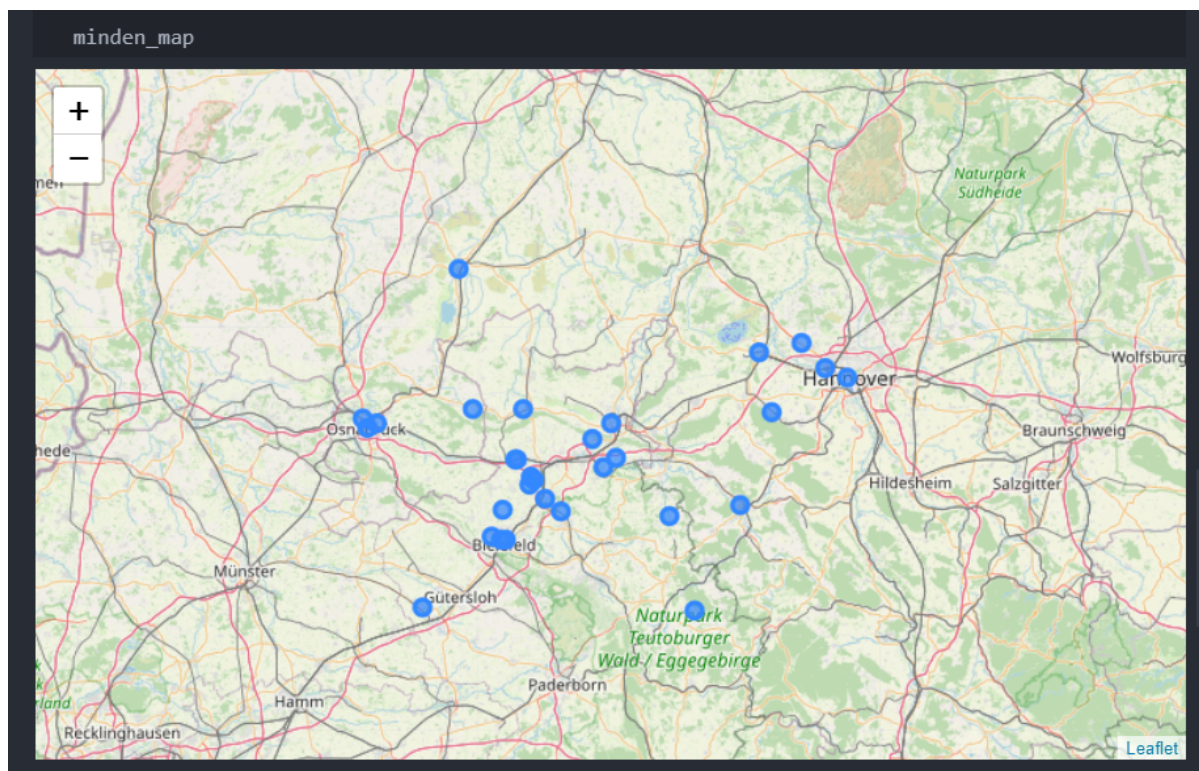


Figure 3.2: Foursquare dataset visualized in map

```
df_ebay.head()
```

	itemId	title	globalId	primaryCategory/categoryId	primaryCategory/categoryName
0	233800267763	ACER Nitro 5 (AN515-52-76YJ) schwarz Gaming- No...	EBAY-DE	177	PC Notebooks & Netbooks
1	184560017711	MacBook Pro (Retina 15 Zoll, Anfang 2013), 8 G...	EBAY-DE	111422	Apple Notebooks
2	154228405289	MacBook Pro 13" 2016 Touch Bar, generalüberholt	EBAY-DE	111422	Apple Notebooks
3	133590834366	HP Pavillion Notebook, 17", 16GB, AMD A8- 6410 ...	EBAY-DE	177	PC Notebooks & Netbooks
4	383843713150	Laptop Acer Extensa 5635*15,6 Zoll*Intel Core ...	EBAY-DE	177	PC Notebooks & Netbooks

Figure 3.3: Excerpt of eBay dataset right after fetching

The Foursquare dataset is missing city and state, while the eBay dataset is missing coordinates unfortunately, as well as state. In the next step, I will preprocess the data and add the missing columns.

To preprocess the data, I will use OpenStreetMap¹. It provides a reverse geocoding function, allowing to search for locations for example by name or coordinates and receiving vast information in return. See below for the code and result.

¹<https://geocoder.readthedocs.io/providers/OpenStreetMap.html>

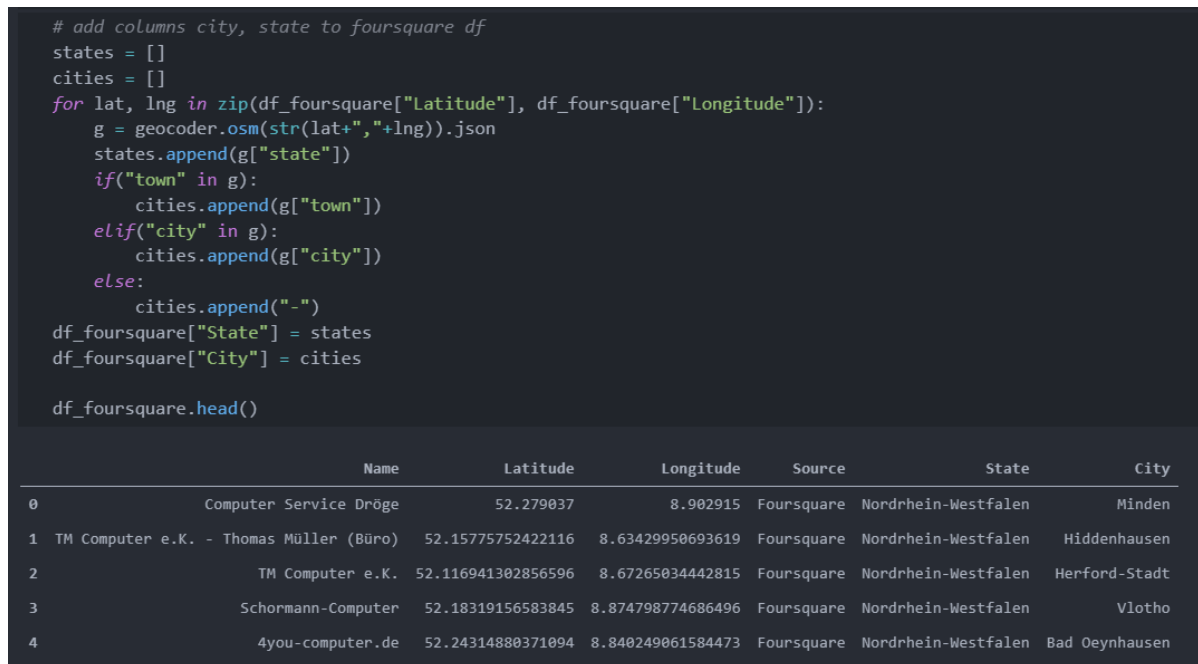


Figure 3.4: Foursquare data with added state and city

The eBay data is missing state, latitude and longitude. These columns will also be added by using OpenStreetMap. See the following results.

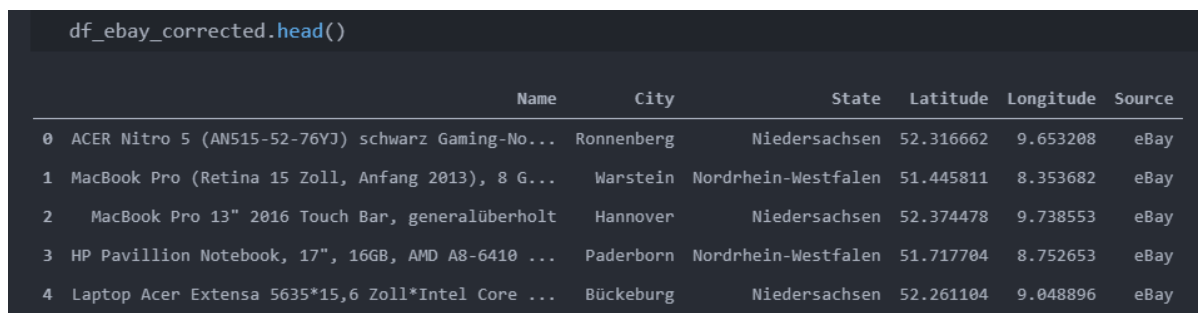


Figure 3.5: eBay data with added state and coordinates

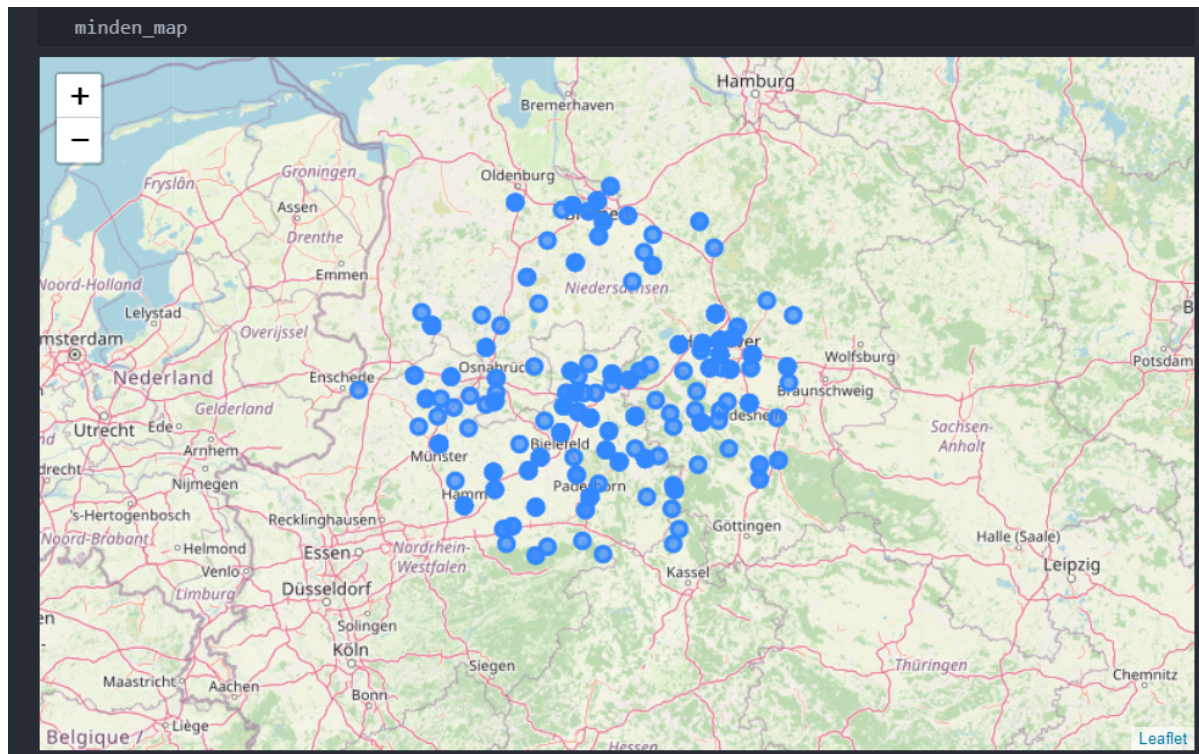


Figure 3.6: eBay dataset visualized in map

The two datasets are now combined and stored as a csv locally, to make sharing the data easier. This coherent structure now allows for some exploratory data analysis in the following subsections.

3.3 Exploratory data analysis

In this section, I will demonstrate how I managed to get a general overview over the data. This chapter only covers methodology, the results are discussed in 4.

3.3.1 Statistical key figures and indicators

To begin with some basic operations after creating a combined dataset, I described the given dataframe.

	Name	City	State	Latitude	Longitude	Source
count	630	630	630	630.000000	630.000000	630
unique	598	139	10	NaN	NaN	2
top	15.6 Zoll Laptop 4GB RAM 64GB SSD Intel Celero...	Wedemark	Niedersachsen	NaN	NaN	eBay
freq	4	106	345	NaN	NaN	600
mean	NaN	NaN	NaN	52.343838	8.919098	NaN
std	NaN	NaN	NaN	0.703020	1.630962	NaN
min	NaN	NaN	NaN	45.947330	-3.272988	NaN
25%	NaN	NaN	NaN	52.021274	8.531007	NaN
50%	NaN	NaN	NaN	52.374478	9.120040	NaN
75%	NaN	NaN	NaN	52.562287	9.703703	NaN
max	NaN	NaN	NaN	54.740692	11.954355	NaN

Figure 3.7: Description of dataset

3.3.2 Visualizing entries using a map

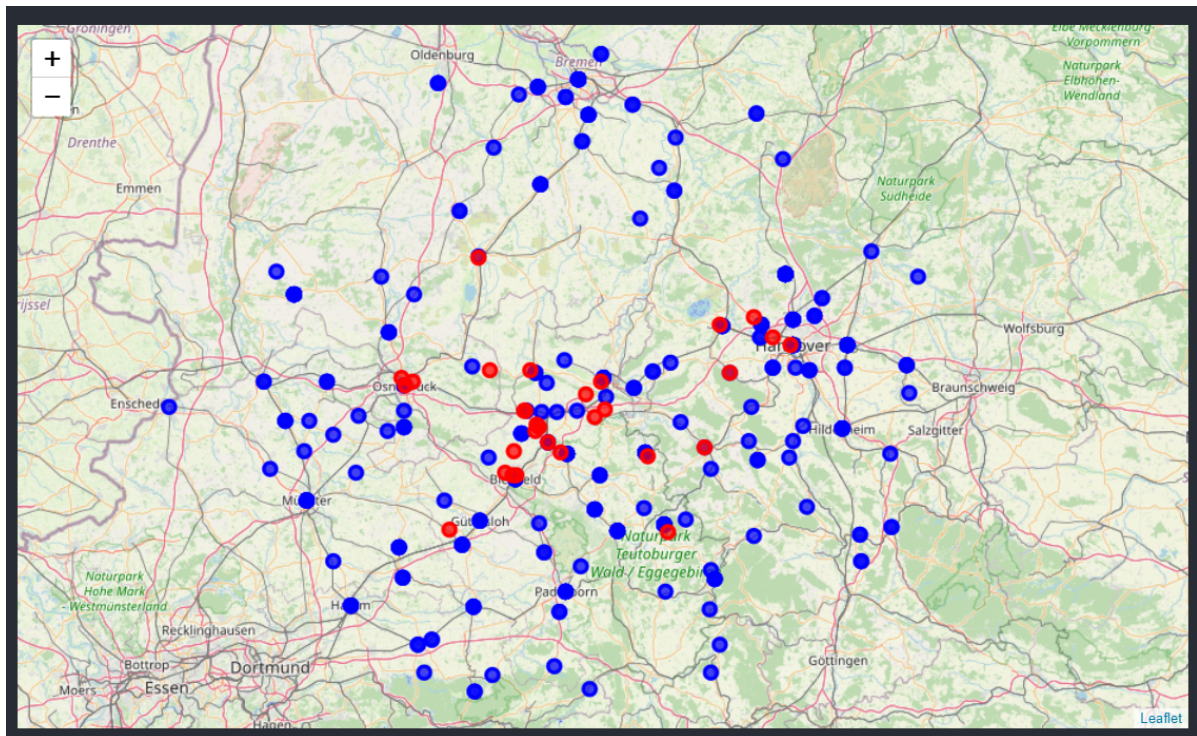


Figure 3.8: Visualization of dataset

This is what the map looks like when both sources are considered. Foursquare entries are colored red, eBay ones are blue.

3.4 Machine learning

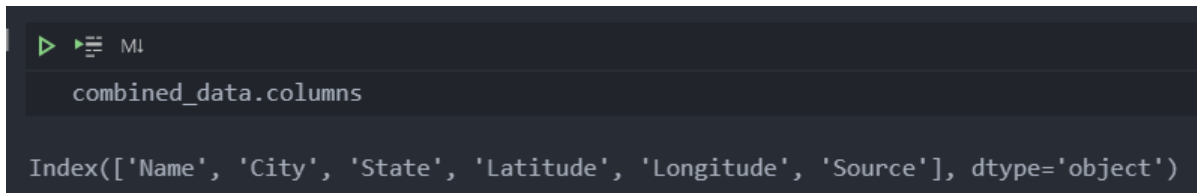
Next, I will use the dataset to demonstrate the KNN algorithm. I want to cluster my entries by the state they are based in. KNN will be trained with the coordinates of the entries as independent variable. The target value therefore is the state of the entries.

3.4.1 Clustering entries using KNN

To set up KNN, the data first needs to be split by separating independent and dependent variables.

Display Columns

To begin with and to get a feeling for the data I am working with, I chose to display the available columns within my dataset.



```
combined_data.columns

Index(['Name', 'City', 'State', 'Latitude', 'Longitude', 'Source'], dtype='object')
```

Figure 3.9: Columns of dataset

Because in my case, the state is dependent of the coordinates, the only three coordinates will be Latitude, Longitude and State. The first two being the independent variable, the latter being the target value that depends on the first two. So for the next step, choose X and y accordingly.

Select columns for X and y

As explained, X will contain Latitude and Longitude, whereas y will consist of the state.



```

X = combined_data[['Latitude', 'Longitude']].values.astype(float)
X[0:5]

array([[52.3166623,  9.6532075],
       [51.4458105,  8.3536824],
       [52.3744779,  9.7385532],
       [51.7177044,  8.752653 ],
       [52.2611037,  9.0488959]])

y = combined_data['State'].values
y[0:5]

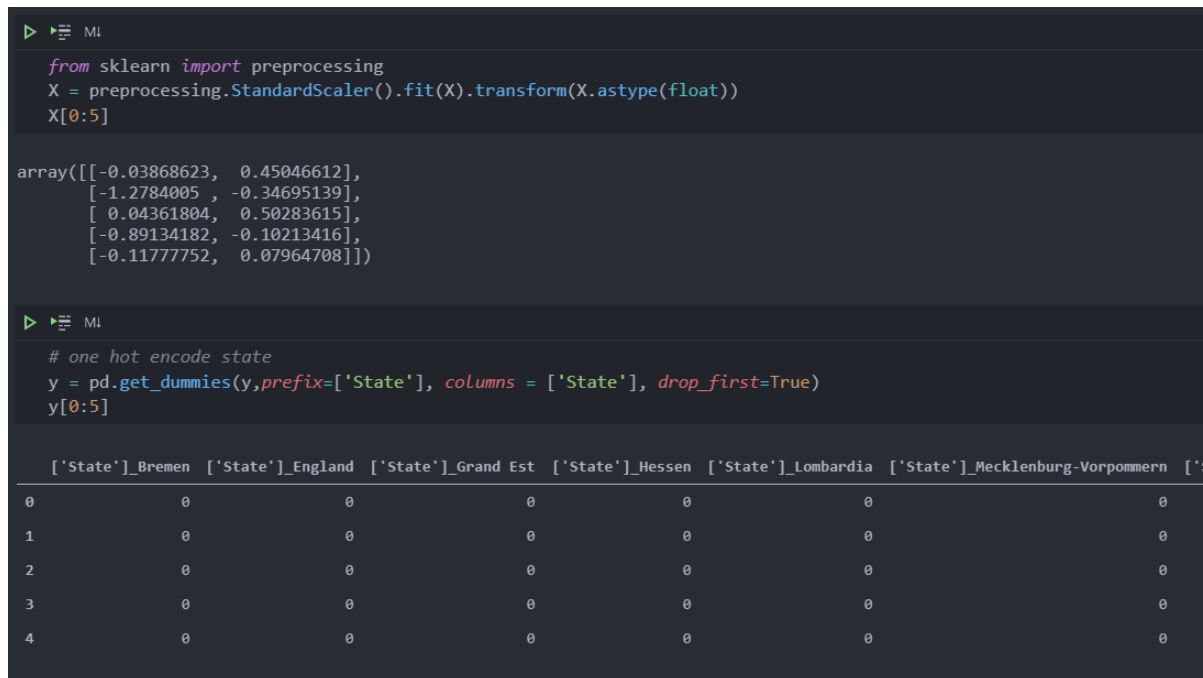
array(['Niedersachsen', 'Nordrhein-Westfalen', 'Niedersachsen',
       'Nordrhein-Westfalen', 'Niedersachsen'], dtype=object)
```

Figure 3.10: Selecting columns for KNN

Note that the coordinates are not standardized yet, and therefore need to be preprocessed. As of now, y also contains categorical data. The data will be hot encoded in the next step.

Preprocess X and y

Data Standardization give data zero mean and unit variance, it is good practice, especially for algorithms such as KNN which is based on distance of cases. Therefore, I begin with scaling X as in the course example.



```

from sklearn import preprocessing
X = preprocessing.StandardScaler().fit(X).transform(X.astype(float))
X[0:5]

array([[ -0.03868623,  0.45046612],
       [-1.2784005 , -0.34695139],
       [ 0.04361804,  0.50283615],
       [-0.89134182, -0.10213416],
       [-0.1177752 ,  0.07964708]])

```

```

# one hot encode state
y = pd.get_dummies(y,prefix=['State'], columns = ['State'], drop_first=True)
y[0:5]

```

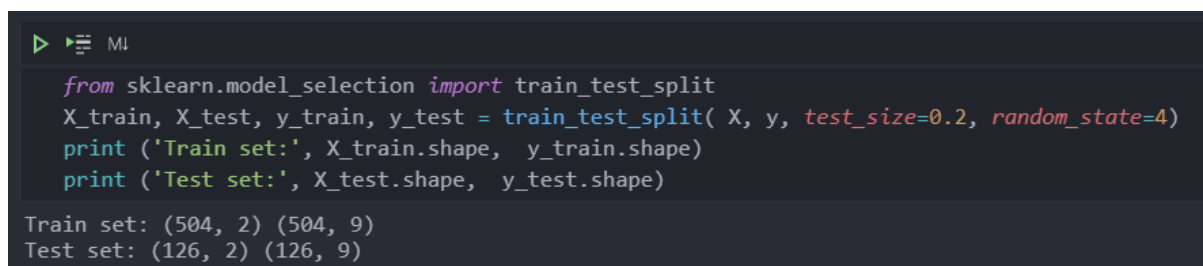
	['State']_Bremen	['State']_England	['State']_Grand Est	['State']_Hessen	['State']_Lombardia	['State']_Mecklenburg-Vorpommern	['State']_North Rhine-Westphalia
0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0

Figure 3.11: Preprocessing X and y

Additionally, I also one hot encoded the categorical state value. Note that there are states which are further away than 100km. This is because of the geocoder api and will be explained in chapter 4.

Define test and train set

Next, the data will be split into train and test set.



```

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.2, random_state=4)
print ('Train set:', X_train.shape, y_train.shape)
print ('Test set:', X_test.shape, y_test.shape)

```

```

Train set: (504, 2) (504, 9)
Test set: (126, 2) (126, 9)

```

Figure 3.12: Test Train Split

Begin classification

With the test and train set in hand, I can now find most accurate k for KNN. In the plot below, ks from 0 to 100 were tested.

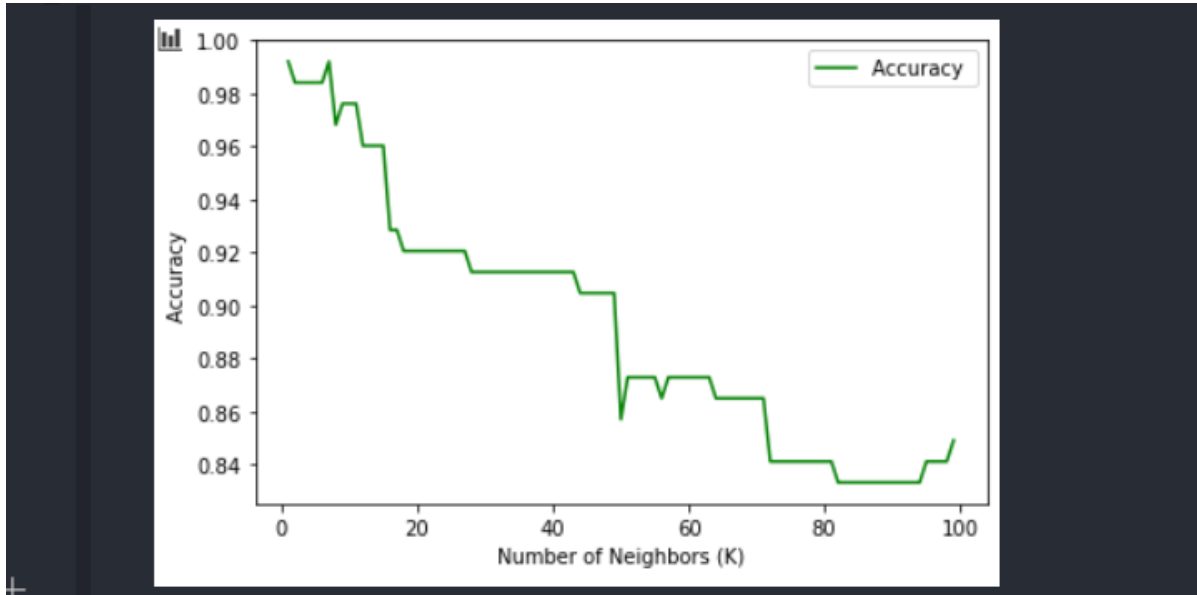


Figure 3.13: Finding best k for KNN

0 and 6 were the most accurate. I will continue using $k=6$ in the following steps when making the prediction and once again comparing the accuracy.


```
▶ ML
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics

k = 6
neigh = KNeighborsClassifier(n_neighbors = k).fit(X_train,y_train)
neigh

KNeighborsClassifier(n_neighbors=6)

▶ ML
yhat = neigh.predict(X_test)
yhat[0:5]

array([[0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 1, 0, 0, 0, 0, 0, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0],
       [0, 0, 0, 0, 0, 0, 1, 0, 0]], dtype=uint8)

▶ ML
from sklearn import metrics
print("Train set Accuracy: ", metrics.accuracy_score(y_train, neigh.predict(X_train)))
print("Test set Accuracy: ", metrics.accuracy_score(y_test, yhat))

Train set Accuracy:  0.9682539682539683
Test set Accuracy:  0.9841269841269841
```

Figure 3.14: Predicting state based on coordinates and accuracy

4 Results, Discussion and Conclusion

In the beginning, i posed the question whether the information I gained during this short project might be insightful for customers and sellers. While it is certainly useful to have the data of two completely different sources combined and unified in one document, I was not able to do as much with the data as my initally intended to. It was possible to handle visualizing the data and applying KNN, but for more results (sales forecasts, most attractive location based on sales, etc.) the data was missing more columns such as actual sales data and reports. Nonetheless, I was able to apply many concepts I learned during the course and worked with my own goal in mind, always double checking with the labs from different weeks.

I was able to collect data from computer stores in my area using eBay and Foursquare. The results are somewhat insightful, as I could unify the two datasets and visualize the different locations and offerings within my area. One thing I noticed quite early when preprocessing the data is that some entries were mistakenly matched to a wrong state or even country, when geocoding the city names. While the coordinates could be easily identified and mapped to a town or city, the city name alone that was provided by eBay is not enough as there are multiple cities with the same name. However, these small outliers did not affect the result very much.

As for the conclusion of this project, I would like to say that I really enjoyed applying the concepts and methods that were presented in the course. The capstone project is more or less constructed and not really tied to a real world business case, but I still think that it acts great as a foundation for future work. The contents of the 9 courses do not cover everything I will ever come across in the field of data science, but I was able to gain a solid understanding of basic concepts and can make use of that in the future.