Project 2023

Sissejuhatus

Teame ütlust, et ühes õiges poisslapse nimes peab olema "R"-täht. Sellest kujunes välja mõte, uurida, kas tuleb esile ka mõni täht või tähed, mis esineb enamikus tänava nimedes. Ilmselt siin võib olla ka mingi seos, et eesti keeles, esineb teatud tähti sagedamini kui teisi, aga samas võib olla tulemus ka erinev. Näiteks tänava nimedeks valitakse sõnu, kus need eesti keelele üldised reeglid tähtede esinemissagedusest ei kehti.

Antud projektis kasutame

https://avaandmed.eesti.ee/ (https://avaandmed.eesti.ee/)

andmebaasi. Täpsemalt oleme valinud välja:

https://avaandmed.eesti.ee/datasets/tallinna-aadresside-tabelid-asumite-kaupa (https://avaandmed.eesti.ee/datasets/tallinna-aadresside-tabelid-asumite-kaupa)

Ja veelgi täpsemalt valime sealt "Tallinna aadresside tabelid asumite kaupa 29.11.2015 10 MB (XLSX)" ehk "aadressid20151129.xlsx":

https://www.tallinn.ee/est/g21786s99123 (https://www.tallinn.ee/est/g21786s99123)

See ei ole küll kõige uuem, kuid on kõige suurema kirjete arvuga andmebaas mis leitav antud teemal.

Antmete puhastamine ja sorteerimine

Esialgselt on meil 272'075 kirjet. Aga meil on ühe tänava kohta mitu kirjet, näiteks iga maja number. Samas me ei soovi anda tänava nimedele erinevaid kaale meie valimis, lähtuvalt näiteks sellest, kui pikk üks või teine tänav on või kui mitu maja sellele tänavale tänase seisuga ehitatud on. Meie soov on kohelda igat tänava nime võrdselt meie valimis.

Seega viskame välja korduvad tänava nimed ja ka majanumbrid.

In [1]:
```
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd

import os
os.chdir(r"C:\Users\Dell\Desktop\Programeerimine\TU - Praktiline andmeteadu
```

In [2]:
```
# Replace 'path_to_excel_file.xlsx' with the path to your Excel file
excel_file_path = 'aadressid20151129.xlsx'

# Read the Excel file into a DataFrame
dataframe = pd.read_excel(excel_file_path)

# Display the first few rows of the DataFrame
print(dataframe.head())
```

```
                                          Täisaadress        Lähiaadress  \
0  Harju maakond, Tallinn, Haabersti linnaosa, As...         Astangu tn
1  Harju maakond, Tallinn, Haabersti linnaosa, As...       Astangu tn 1
2  Harju maakond, Tallinn, Haabersti linnaosa, As...      Astangu tn 19
3  Harju maakond, Tallinn, Haabersti linnaosa, As...    Astangu tn 19/1
4  Harju maakond, Tallinn, Haabersti linnaosa, As...  Astangu tn 19/1-1

      Asum            Linnaosa           Linn
0  Astangu  Haabersti linnaosa  Tallinna linn
1  Astangu  Haabersti linnaosa  Tallinna linn
2  Astangu  Haabersti linnaosa  Tallinna linn
3  Astangu  Haabersti linnaosa  Tallinna linn
4  Astangu  Haabersti linnaosa  Tallinna linn
```

Salvestame selle arvuti jõudluse parandamiseks CSV formaati.

In [3]:
```
# Assuming 'dataframe' is the DataFrame you want to save
# Replace 'path_to_csv_file.csv' with the path where you want to save the C

csv_file_path = '001.csv'

# Save the DataFrame to a CSV file with "|" as the delimiter
dataframe.to_csv(csv_file_path, sep='|', index=False, encoding='utf-8')


# The 'index=False' argument is used to prevent pandas from writing row ind
```

Teeme mälu tühjaks:

In [4]:
```
del dataframe
```

Loeme oma CSV faili sisse ja töötleme seda natukene:

In [17]:
```
# Replace 'path_to_csv_file.csv' with the path to your CSV file
csv_file_path = '001.csv'

# Read the CSV file into a DataFrame with "|" as the delimiter
dataframe = pd.read_csv(csv_file_path, sep='|')

# Display the first few rows of the DataFrame
print(dataframe.head())
```

```
                                          Täisaadress        Lähiaadress  \
0  Harju maakond, Tallinn, Haabersti linnaosa, As...         Astangu tn
1  Harju maakond, Tallinn, Haabersti linnaosa, As...       Astangu tn 1
2  Harju maakond, Tallinn, Haabersti linnaosa, As...      Astangu tn 19
3  Harju maakond, Tallinn, Haabersti linnaosa, As...    Astangu tn 19/1
4  Harju maakond, Tallinn, Haabersti linnaosa, As...  Astangu tn 19/1-1

      Asum            Linnaosa           Linn
0  Astangu  Haabersti linnaosa  Tallinna linn
1  Astangu  Haabersti linnaosa  Tallinna linn
2  Astangu  Haabersti linnaosa  Tallinna linn
3  Astangu  Haabersti linnaosa  Tallinna linn
4  Astangu  Haabersti linnaosa  Tallinna linn
```

```
sssss
```

In [18]:
```
# List of columns to be removed
columns_to_drop = ["Täisaadress", "Asum", "Linnaosa", "Linn"]

# Drop the specified columns from the DataFrame
df_dropped = dataframe.drop(columns=columns_to_drop)

# Display the DataFrame after dropping the columns
print(df_dropped.head())
```

```
         Lähiaadress
0         Astangu tn
1       Astangu tn 1
2      Astangu tn 19
3    Astangu tn 19/1
4  Astangu tn 19/1-1
```

```
Tere
```

In [ ]:
```

```

In [49]:
```
df = df_dropped

# Regular expression to match any digits
pattern = r'\d'

# Keep rows where 'Lähiaadress' does NOT contain any digit
df_filtered = df[~df['Lähiaadress'].str.contains(pattern, na=False)]

# Display the filtered DataFrame
print(df_filtered.head())
```

```
       Lähiaadress
0       Astangu tn
2162     Jalami tn
2246   Kotermaa tn
2271   Moonalao tn
2285  Paldiski mnt
```

Puhastame "tn" ja "mnt" nime lõpust:

In [50]:
```
# Assuming df_filtered is a DataFrame created by filtering df

# Use regex to replace ' tn' or ' mnt' at the end of the string with an emp
df_filtered.loc[:, 'Lähiaadress'] = df_filtered.loc[:, 'Lähiaadress'].str.r

# Display the modified DataFrame
print(df_filtered['Lähiaadress'])
```

```
0              Astangu
2162            Jalami
2246          Kotermaa
2271          Moonalao
2285          Paldiski
              ...
270330    Manufaktuuri
270342           Niidi
270347           Niidi
270498        Puuvilla
270698           Sitsi
Name: Lähiaadress, Length: 2152, dtype: object
```

In [ ]:

In [56]:
```
# Use regex to replace the pattern 'single letter followed by a period and
df_filtered.loc[:, 'Lähiaadress'] = df_filtered.loc[:, 'Lähiaadress'].str.r

# Display the modified DataFrame
print(df_filtered['Lähiaadress'])
```

```
0              Astangu
2162            Jalami
2246          Kotermaa
2271          Moonalao
2285          Paldiski
              ...
270330    Manufaktuuri
270342           Niidi
270347           Niidi
270498        Puuvilla
270698           Sitsi
Name: Lähiaadress, Length: 2152, dtype: object
```

In [ ]:

In [57]:
```
# Use regex to replace the pattern 'single letter followed by a period' wit
df_filtered.loc[:, 'Lähiaadress'] = df_filtered.loc[:, 'Lähiaadress'].str.r

# Display the modified DataFrame
print(df_filtered['Lähiaadress'])
```

```
0              Astangu
2162            Jalami
2246          Kotermaa
2271          Moonalao
2285          Paldiski
              ...
270330    Manufaktuuri
270342           Niidi
270347           Niidi
270498        Puuvilla
270698           Sitsi
Name: Lähiaadress, Length: 2152, dtype: object
```

In [ ]:

In [ ]:

In [58]:
```
# Assuming 'df_filtered' is your pandas DataFrame

# Create a boolean mask where 'Lähiaadress' contains a space, treating NaN
mask = df_filtered['Lähiaadress'].str.contains(' ', na=False)

# Use the mask to filter the DataFrame
rows_with_space = df_filtered[mask]

# Display the rows where 'Lähiaadress' contains a space
print(rows_with_space)
```

```
                                          Lähiaadress
6666    Harku järve kaldaala (end.) Järve-Männiku kinn...
6760                Keskküla tänava ja Reha tänava maa-ala
7299                                        Oja tn lõik A
8673                                        Rocca al Mare
9442                                        Roostiku tn C
10400             Jõeküla tee lõik Lemle ja Abara tn vahel
10471          Jõeoti tn lõik Tiskre tee ja Abara tn vahel
12190                               Haabersti liiklussõlm
16202                               Väike-Õismäe puhkeala
25944                                           Umboja A
27723                                        August Alle
28057                           Ferdinand Johann Wiedemanni
29516                                        Johann Köleri
41397                                   Järvevana ülesõit
42332                                Pärnu maantee viadukt
44196                                      Eduard Viiralti
46981                           Tornimäe tänava pikendus
50376                                       Rudolf Tobiase
52093                                  Kaljase tänav/Koge
55422                                        Ants Lauteri
56880                                          Georg Otsa
58293                                        Peeter Süda
61293                                    Johannes Kappeli
61567                                 Karl August Hermanni
67048                                        Uue Maailma
67330                                        Uue Maailma
71368                                    Suur Rannavärav
71369                                    Suur Rannavärav
71652                                      Taani Kuninga
72966                                   Väike Rannavärav
72967                                   Väike Rannavärav
80504                                 Hipodroomi liiklussõlm
140631  Laagna tee ja Peterburi tee vahel asuv Rahu te...
193920  Sõpruse pst lõik Tammsaare tee ja Vilde tee vahel
195708  Sõpruse pst lõik Tildri tn ja Tammsaare tee vahel
205372  Vanemuise tänava lõik Ugala tn ja Olevi tn vah...
210316  Valdeku tn lõik Männiku tee ja raudtee ülesõid...
211770                                          von Glehni
211774                                          von Glehni
216782  Pärnu mnt lõik Pääsküla raudteeületuskoha ja l...
216783                               Pääsküla raudteejaama
220693                                    Väikese Illimari
222102                                       Ernst Särgava
222228                  Hunditubaka tee/Pune tee/Karukella
228790                   Haljasala maa-ala Mähe tee kõrval
232895                                    Pirita rannaala
247379                                 Kopli kalmistupark
250550                                   Stroomi rannaala
```

```
251087                                  Pikakari supelrand
263872    Lõime tänava lõik Sõle ja Randla tänavate vahel
```

In [ ]:

In [59]:
```
# Assuming 'df_filtered' is your pandas DataFrame

# Create a boolean mask where 'Lähiaadress' contains a space
mask = df_filtered['Lähiaadress'].str.contains(' ', na=False)

# Drop rows where 'Lähiaadress' contains a space by inverting the mask
df_filtered = df_filtered[~mask]

# Display the DataFrame after dropping the rows
print(df_filtered)
```

```
         Lähiaadress
0             Astangu
2162           Jalami
2246         Kotermaa
2271         Moonalao
2285         Paldiski
...               ...
270330   Manufaktuuri
270342          Niidi
270347          Niidi
270498       Puuvilla
270698          Sitsi

[2102 rows x 1 columns]
```

In [ ]:

In [61]:
```
# Sort 'df_filtered' by 'Lähiaadress' in ascending order
df_filtered = df_filtered.sort_values(by='Lähiaadress', ascending=True)

# Display the sorted DataFrame
print(df_filtered)
```

```
       Lähiaadress
251088       Aarde
63489         Aasa
210495        Aate
222079       Abaja
10297        Abara
...            ...
160917         NaN
193939         NaN
214120         NaN
233061         NaN
242016         NaN

[2102 rows x 1 columns]
```

In [ ]:

In [63]:
```
# Drop duplicates in 'Lähiaadress' column, keeping the first occurrence and
df_filtered = df_filtered.drop_duplicates(subset='Lähiaadress', keep='first

# Display the DataFrame after dropping the duplicates
print(df_filtered)
```

```
        Lähiaadress
251088        Aarde
63489          Aasa
210495         Aate
222079        Abaja
10297         Abara
...             ...
96066         Ülase
76670      Ülemiste
193936  Üliõpilaste
149526        Ümera
25193           NaN

[1470 rows x 1 columns]
```

In [ ]:

In [70]:
```
# Assuming 'df_filtered' is your pandas DataFrame

# Create a boolean mask where 'Lähiaadress' is not empty and not equal to '
mask = (df_filtered['Lähiaadress'].astype(str) != '') & (df_filtered['Lähia

# Use the mask to filter the DataFrame
df_filtered = df_filtered[mask]

# Display the DataFrame after removing the specified rows
print(df_filtered)
```

```
        Lähiaadress
251088        Aarde
63489          Aasa
210495         Aate
222079        Abaja
10297         Abara
...             ...
96066         Ülase
76670      Ülemiste
193936  Üliõpilaste
149526        Ümera
25193           NaN

[1470 rows x 1 columns]
```

In [ ]:

In [71]:
```
# Assuming 'dataframe' is the DataFrame you want to save
# Replace 'path_to_csv_file.csv' with the path where you want to save the C

csv_file_path = '002.csv'

# Save the DataFrame to a CSV file with "|" as the delimiter
df_filtered.to_csv(csv_file_path, sep='|', index=False, encoding='utf-8')

# The 'index=False' argument is used to prevent pandas from writing row ind
```

Näeme, et meil jäi alles 1470 kirjet. Seda pole just palju. Võiks ju terve eesti tänavanimee peale seda uuringut laiendada, aga ei tea sellise andmebaasi olemasolu hetkel. Samas tulevikus võib alati asja edasi arendada.

In [ ]:

Loeme tähed üle

Selleks kõigepealt lööme sõnad tähtedeks ja loeme ka sõna pikkused üle.

In [ ]:

In [80]:
```
# Replace 'path_to_csv_file.csv' with the path to your CSV file
csv_file_path = '002.csv'

# Read the CSV file into a DataFrame with "|" as the delimiter
df3 = pd.read_csv(csv_file_path, sep='|')

# Display the first few rows of the DataFrame
print(df3.head())
```

```
  Lähiaadress
0       Aarde
1        Aasa
2        Aate
3       Abaja
4       Abara
```

In [ ]:

In [83]:
```
# Assuming 'df3' is your pandas DataFrame with only one column 'Lähiaadress

# List of new columns to be added, including 'Pikkus' for length
new_columns = ['Pikkus', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J',
               'Z', 'Ž', 'T', 'U', 'V', 'W', 'Õ', 'Ä', 'Ö', 'Ü', 'X', 'Y']

# Initialize the new columns with default value 0
for column in new_columns:
    df3[column] = 0

# Define a function to count the occurrences of each character
def count_chars(word):
    if pd.isnull(word):  # Check if the entry is NaN
        return {}
    char_count = {}
    word = str(word).upper()  # Convert to string and upper case to make th
    for char in word:
        if char in new_columns:
            if char in char_count:
                char_count[char] += 1
            else:
                char_count[char] = 1
    return char_count

# Iterate over each row and count the occurrences of each letter
for index, row in df3.iterrows():
    address = row['Lähiaadress']
    # Count the characters in 'Lähiaadress'
    counts = count_chars(address)
    # Assign the length of 'Lähiaadress' to 'Pikkus' handling NaN separatel
    df3.at[index, 'Pikkus'] = len(address) if pd.notnull(address) else 0
    # Assign the counts to the respective columns
    for char, count in counts.items():
        df3.at[index, char] = count

# Display the DataFrame with the new columns
print(df3)
```

```
      Lähiaadress  Pikkus  A  B  C  D  E  F  G  H  ...  T  U  V  W  Õ  Ä
Ö  \
0           Aarde       5  2  0  0  1  1  0  0  0  ...  0  0  0  0  0  0
0
1            Aasa       4  3  0  0  0  0  0  0  0  ...  0  0  0  0  0  0
0
2            Aate       4  2  0  0  0  1  0  0  0  ...  1  0  0  0  0  0
0
3           Abaja       5  3  1  0  0  0  0  0  0  ...  0  0  0  0  0  0
0
4           Abara       5  3  1  0  0  0  0  0  0  ...  0  0  0  0  0  0
0
...           ...     ... .. .. .. .. .. .. .. ..  ... .. .. .. .. .. ..
..
1465        Ülase       5  1  0  0  0  1  0  0  0  ...  0  0  0  0  0  0
0
1466     Ülemiste       8  0  0  0  0  2  0  0  0  ...  1  0  0  0  0  0
0
1467  Üliõpilaste      11  1  0  0  0  1  0  0  0  ...  1  0  0  0  1  0
```

In [ ]:

In [82]:
```
# Assuming 'dataframe' is the DataFrame you want to save
# Replace 'path_to_csv_file.csv' with the path where you want to save the C

csv_file_path = '003.csv'

# Save the DataFrame to a CSV file with "|" as the delimiter
df3.to_csv(csv_file_path, sep='|', index=False, encoding='utf-8')

# The 'index=False' argument is used to prevent pandas from writing row ind
```

In [ ]:

Loeme kokku tähed

In [87]:
```
# Assuming 'df3' is your DataFrame and it has been properly loaded with the

# Define the columns that we want to sum up, excluding 'Lähiaadress' and 'P
columns_to_sum = [
    'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', '
    'P', 'Q', 'R', 'S', 'Š', 'Z', 'Ž', 'T', 'U', 'V', 'W', 'Õ', 'Ä', 'Ö', '
    'X', 'Y'
]

# Sum up the specified columns and create a new dataframe df11 with the res
df11 = pd.DataFrame(df3[columns_to_sum].sum()).transpose()
df11.columns = columns_to_sum

# Display the dataframe
df11
```

Out[87]:

	A	B	C	D	E	F	G	H	I	J	...	T	U	V	W	Õ	Ä	Ö	Ü
0	1396	96	0	183	914	8	127	190	1087	146	...	384	688	341	4	130	197	23	58

1 rows × 32 columns

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: