THE MOST SUITABLE PLACE TO OPEN A CAFE IN ISTANBUL

INTRODUCTION

- Opening a cafe in Istanbul might be difficult but deciding to do so is even more difficult. You must have great strategies and come up with great ideas to successfully open your cafe.

- While doing so you can count endless challenges, and this project aimes to solve some of these challenges you may encounter.

BUSINESS PROBLEM

- There are so many cafes in Istanbul, therefore if you open your cafe in a place full of cafes, you might not be successful.

- Likewise, if you open your cafe in a place that has no attraction, you still might fail.

- The aim of this project is to show you the best places to open your cafe.

TARGET AUDIANCE

- The project aims to reach people who are;
- - considering about opening a cafe in Istanbul.
- - aiming to find the best place to open a cafe.
- - curious about the most popular venues in Istanbul's boroughs.

DESIRED OUTCOME

The best outcome of this project would be to see;

- The most popular venues in every borough, which are cafes mostly

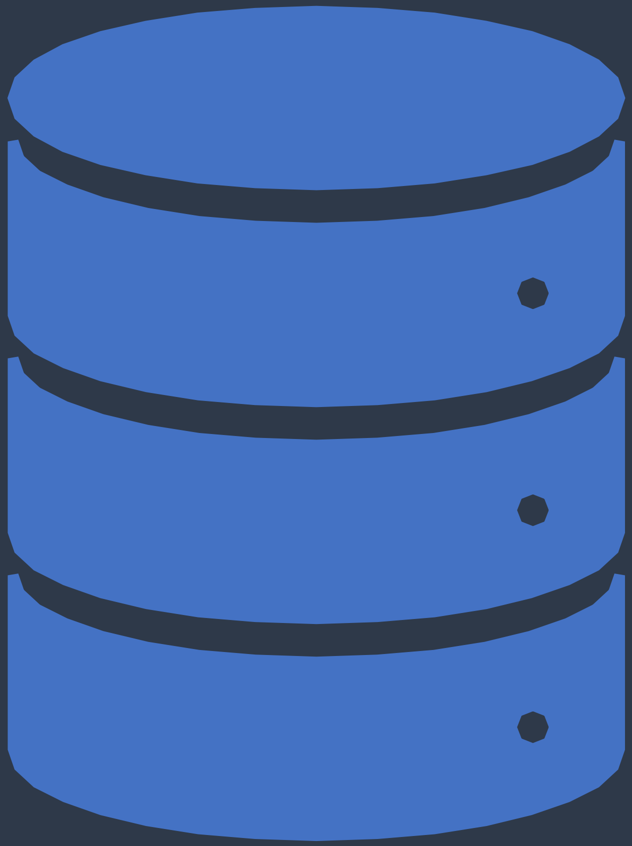
- The places full of cafes, in order to avoid those places

- The places that have enough attraction, not completely empty

- The best place to open your cafe

DATA ACQUISITION

- The sources that are used in this project are:
- - atlasbig.com
- - geopy client
- - Foursquare API



DATA PREPARATION

- Boroughs and their information like are acquired from atlasbig.com
- Geographic coordination information of the boroughs are acquired from geopy client
- Most popular venue categories are acquired from Foursquare API
- Maps are created by using Folium library

METHODOLOGY

- After scaping borough information into a DataFrame from the source web site, each one's latitude and longitude information and were added to the DataFrame. This far, all processes were made by Python libraries.

Getting Istanbul's Boroughs Information

```
# Web scraping atlasbig.com
table = pd.read_html('https://www.atlasbig.com/tr/istanbulun-mahalleleri', encoding = 'utf-8')
ist = table[0]
ist.rename(columns={'Mahalle': 'Neighborhood', 'İlçe': 'Borough', 'Nüfus': 'Population', 'Yüzölçümü (km2)': 'Area (km2)'}, inpla

ist.head()
```

	Neighborhood	Borough	Population	Area (km2)
0	Atakent	Küçükçekmece	93.229	852
1	Adnan Kahveci	Beylikdüzü	86.584	459
2	Zafer	Bahçelievler	85.464	1085
3	Zümrütevler	Maltepe	82.651	3244
4	Halkalı Merkez	Küçükçekmece	78.180	4409

Adding Geographic Information of Boroughs

```
# Copying and shaping the last created DataFrame
nwdf = ist.copy()
nwdf["Latitude"] = np.nan
nwdf["Longitude"] = np.nan
nwdf.drop(['Neighborhood'], axis=1, inplace = True)
nwdf = nwdf.drop_duplicates(subset='Borough', keep='first')

# Adding latitude and longitude information of each borough
for index, row in nwdf.iterrows():
    address = row['Borough']
    geolocator = Nominatim(user_agent="explorer")
    location = geolocator.geocode(address)
    if location != None:
        latitude = location.latitude
        longitude = location.longitude
        nwdf.loc[nwdf.Borough == row['Borough'], ['Latitude', 'Longitude']] = latitude, longitude

# Dropping NaN values and resetting index
nwdf.dropna(subset=['Latitude', 'Longitude'], inplace = True)
nwdf.reset_index(drop = True, inplace = True)

nwdf.head()
```

	Borough	Population	Area (km2)	Latitude	Longitude
0	Küçükçekmece	93.229	852	41.000214	28.780889
1	Beylikdüzü	86.584	459	41.001026	28.641984
2	Bahçelievler	85.464	1085	38.881312	35.627761
3	Maltepe	82.651	3244	40.923542	29.132836
4	Başakşehir	74.815	13835	41.097693	28.806163

METHODOLOGY

- By using a Foursquare app with the help of app’s client ID and client secret, the most popular venue type of each borough were added to the DataFrame.

Using Foursquare API to Get the Most Popular Venue Category

```
# Foursquare settings
CLIENT_ID = 'LFEN4Q2NPD1VGHAZC5FUPDROGL3UUKWYAWOG5OAFKLM1CVJ0'
CLIENT_SECRET = 'ZYRSZE05NUDXFCHNQTV12MYCNYJBXQ2QFOCLHMNQG4QRTXW1'
VERSION = '20180605'
LIMIT = 100

# Copying and shaping the last created DataFrame
final_table = nwdf.copy()
final_table["Most Popular"] = np.nan

# Extracting the category of venues
def get_category_type(row):
    try:
        categories_list = row['categories']
    except:
        categories_list = row['venue.categories']

    if len(categories_list) == 0:
        return None
    else:
        return categories_list[0]['name']


# Creating a Foursquare URL and getting the most popular venue from it
for i in range(34):
    borough_latitude = final_table.loc[i, 'Latitude']
    borough_longitude = final_table.loc[i, 'Longitude']
    borough_name = final_table.loc[i, 'Borough']

    # Creating the URL with our variables
    LIMIT = 100
    radius = 500
    url = 'https://api.foursquare.com/v2/venues/explore?&client_id={}&cli
        CLIENT_ID,
        CLIENT_SECRET,
        VERSION,
        borough_latitude,
        borough_longitude,
        radius,
        LIMIT)

    # Getting the JSON that the URL returned, and getting venues from it
    results = requests.get(url).json()
    venues = results['response']['groups'][0]['items']
    nearby_venues = json_normalize(venues)

    # Getting the most popular venue from the results above, if it exist
    try:
        desired_column = ['venue.categories']
        nearby_venues = nearby_venues.loc[:, desired_column]

        # Filtering the category for each row
        nearby_venues['venue.categories'] = nearby_venues.apply(get_cate

        # Cleaning columns
        nearby_venues.columns = [col.split(".")[-1] for col in nearby_ve

        # Getting the mode value, and adding it to the DataFrame
        modd = nearby_venues.mode()
        popi = modd.at[0,'categories']
        final_table['Most Popular'][i] = popi

    # In case of the mode does not exist
    except:
        pass

# Shaping the DataFrame
final_table.dropna(subset=['Most Popular'], inplace = True)
final_table.reset_index(drop = True, inplace = True)

final_table.head()
```

	Borough	Population	Area (km2)	Latitude	Longitude	Most Popular
0	Küçükçekmece	93.229	852	41.000214	28.780889	Turkish Restaurant
1	Beylikdüzü	86.584	459	41.001026	28.641984	Café
2	Maltepe	82.651	3244	40.923542	29.132836	Café
3	Başakşehir	74.815	13835	41.097693	28.806163	Café
4	Gaziosmanpaşa	73.225	138	41.057526	28.915650	Café

METHODOLOGY

- Since the standard k-means algorithm cannot be directly applicable to categorical data, an encoding process was done to the DataFrame and all values were turned into numeric values. Then the DataFrame was fit into a k-means clustering model and results of each borough were added to the DataFrame.

Encoding DataFrame for Clustering

```
# One hot encoding
ist_onehot = pd.get_dummies(final_table[['Most Popular']], prefix="", prefix_sep="")

# Adding Borough column back to DataFrame
ist_onehot['Borough'] = final_table['Borough']

# Setting Borough as the first column
fixed = [ist_onehot.columns[-1]] + list(ist_onehot.columns[:-1])
ist_onehot = ist_onehot[fixed]

# Grouping the DataFrame by Borough values
ist_grouped = ist_onehot.groupby('Borough').mean().reset_index()

ist_grouped.head()
```

	Borough	Bakery	Bar	Bus Stop	Café	Cocktail Bar	Convenience Store	Gym	Hotel	Turkish Restaurant
0	Arnavutköy	0	0	0	1	0	0	0	0	0
1	Avcılar	0	0	0	1	0	0	0	0	0
2	Bakırköy	0	0	0	0	0	0	1	0	0
3	Bayrampaşa	0	0	0	1	0	0	0	0	0
4	Bağcılar	0	0	0	1	0	0	0	0	0

Clustering

```
# Dropping Borough value since it is not numeric
ist_clustering = ist_grouped.drop('Borough', 1)

# Running K-Means clustering and fitting the new DataFrame
k = 5
kmeans = KMeans(n_clusters = k, random_state = 0).fit(ist_clustering)

#Adding the clustering labels to the DataFrame
final_table.insert(0, 'Cluster Labels', kmeans.labels_)
final_table.head()
```

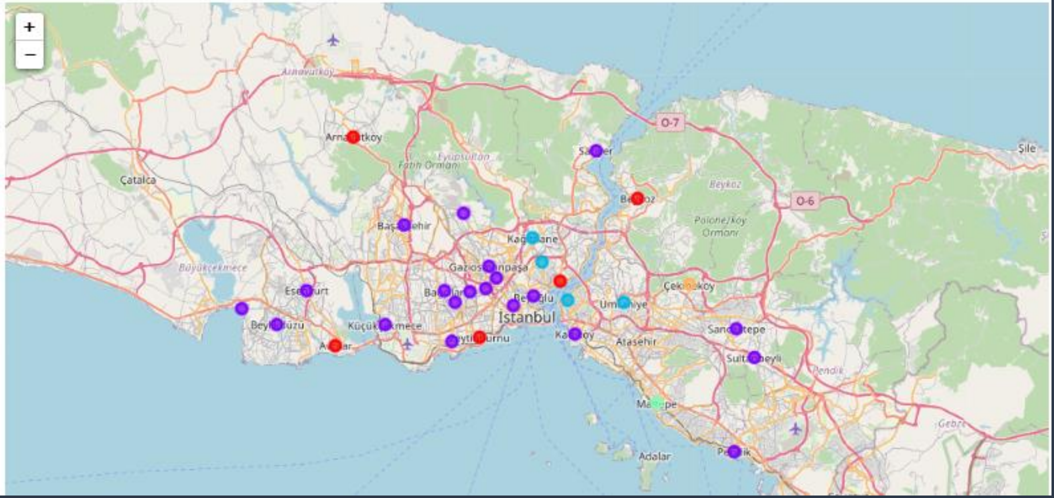
	Cluster Labels	Borough	Population	Area (km2)	Latitude	Longitude	Most Popular
0	1	Küçükçekmece	93.229	852	41.000214	28.780889	Turkish Restaurant
1	1	Beylikdüzü	86.584	459	41.001026	28.641984	Café
2	3	Maltepe	82.651	3244	40.923542	29.132836	Café
3	1	Başakşehir	74.815	13835	41.097693	28.806163	Café
4	1	Gaziosmanpaşa	73.225	138	41.057526	28.915650	Café

METHODOLOGY

- Finally, a map that is based on the cluster and location information was created and shown.

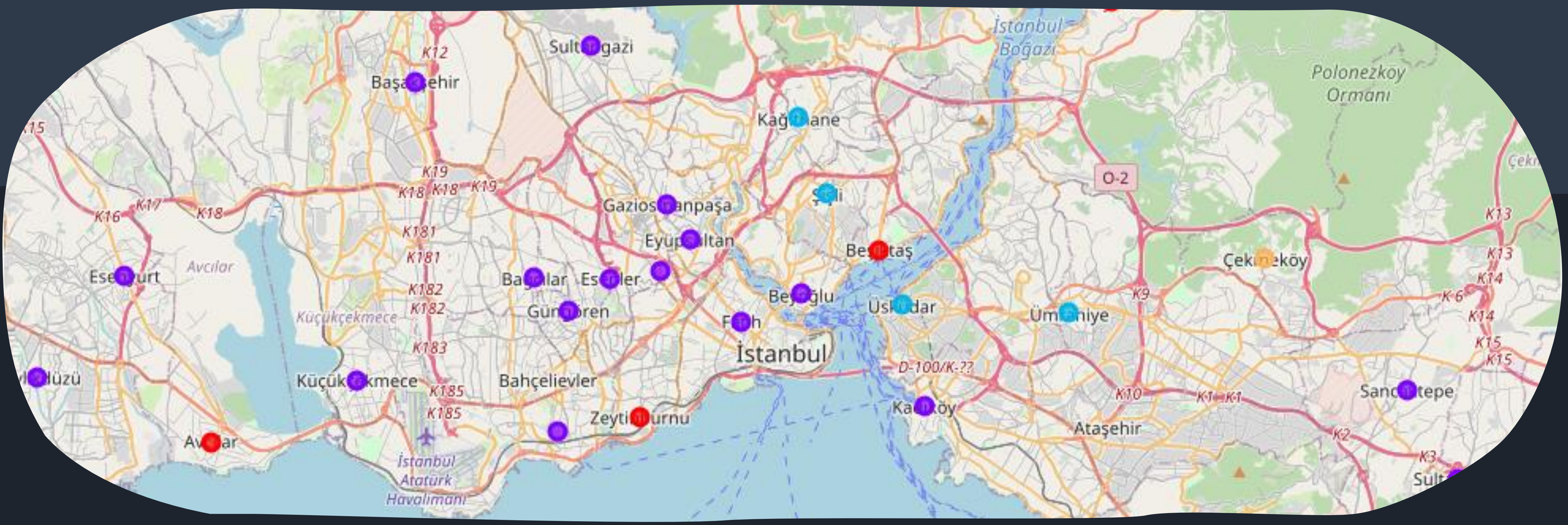
Mapping

```
# Getting Istanbul's coordinates
address = 'Istanbul'
geolocator = Nominatim(user_agent="explorer")
location = geolocator.geocode(address)
latitude = location.latitude
longitude = location.longitude

# Mapping with the newly acquired coordinates
map_clusters = folium.Map(location=[latitude, longitude], zoom_start=10)

# Copying the main DataFrame in order to keep it safe
clustered_table = final_table.copy()

# Setting color scheme for the clusters
x = np.arange(kclusters)
ys = [i + x + (i * x) ** 2 for i in range(k)]
colors_array = cm.rainbow(np.linspace(0, 1, len(ys)))
rainbow = [colors.rgb2hex(i) for i in colors_array]

# Adding markers to the map
for lat, lon, poi, cluster in zip(clustered_table['Latitude'], clustered_table['Longitude'], clustered_table['Borough'], cluster
    label = folium.Popup(str(poi) + ' Cluster ' + str(cluster), parse_html=True, encoding = 'utf-8')
    folium.CircleMarker(
        [lat, lon],
        radius=5,
        popup=label,
        color=rainbow[cluster-1],
        fill=True,
        fill_color=rainbow[cluster-1],
        fill_opacity=0.7).add_to(map_clusters)

map_clusters
```



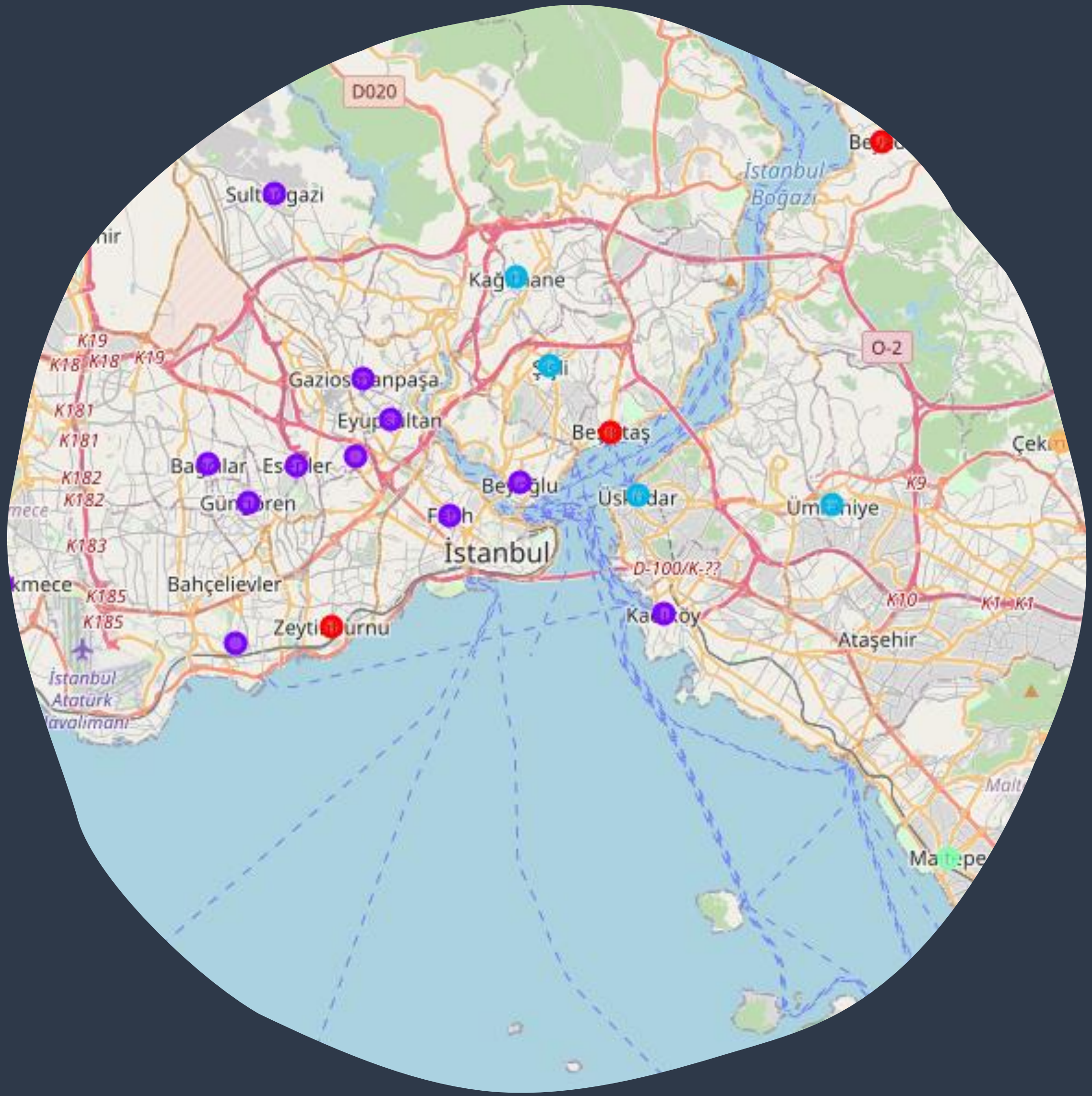
RESULT

- Venues of boroughs can be seen in the final map. All of them were located and clustered previously. Venues and boroughs close to each other can be observed easily.



DISCUSSION

- We can conclude that opening a cafe in a borough which is not too close or too away from others could be the best option.
- According to the map and the algorithm, new cafes in the light blue cluster could be successful.



CONCLUSION

- In this project; the venues in Istanbul, their geographic information and venues in them was analyzed by using several tools. K-Means clustering model was used to cluster venues in boroughs and these clusters was used to build a map. As a future directions, information like population and areas of the boroughs, number of venues and neighborhoods could be used to get a better result.