esade | Business School
RAMON LLULL UNIVERSITY

MASTER's Final Project

Track: *In-Company Project*

Track coordinator: *Prof. Ignasi Algueró*

Title: Deep Air: A Machine Learning Model to Predict Air Pollution in Spain

Organization: *300000 km/s*

Programme: *Master of Science in Business Analytics*

Course 2019 - 2020

Student: *Davide Callegaro, Peter Hendrik Bruins*

Tutor: *Prof. Esteve Almirall*

NOT CONFIDENTIAL

Note: Due to the scope and focus of the project, this document contains a lot of code and plots. We know this and we would not have included them if we were not believing this was necessary to understand what we did. However, we do have this opinion and we hope you will agree with us after reading the document.

Table of Contents

Total number of words: 7810

1. Executive Summary

1.1 Introduction

Air pollution is a structural problem that sometimes can be episodic. Today there are several ways to measure air pollution. It can be described through air stations - points with high accuracy - or through satellite sense - e.g., S5p/tropomi with 5 km resolution. It can be described through simulations or even Land Use Regressions models (LURs) - e.g., Lobelia/isglobal.
However, all these methods have drawbacks:

- Air stations are zonal indicators - i.e., not very detailed.
- Satellite sensing has low resolutions.
- Simulations are too complex and require much computational power.
- LURs do not describe urban complexity, which is a fundamental feature.

Together with 300,000 km/s, a tech-urban start-up located in Barcelona, Davide, and Peter - from now on, we - considered a new scenario to test. As air pollution is a problem of mobility generated by land-use composition, we have tried to predict it using measurements from existing air stations and several features, including but not limited to: urban density, income data, land use composition, building density, dwelling information, and population density and composition.

This project aims to predict the NO2 levels of Spain - the harmful human particles - corner by corner, using correlation matrices, random forest regression trees, graph representations, and spatial lagged features used applied to Moran's I. The spatial lag was introduced to use the strength of data that we possess, namely the geographical information, in the best possible way. By doing so, we were able to introduce 'spatiality' into our machine learning vocabulary. We could extract robust information that would usually be lost in traditional machine learning techniques such as random forests or XGBoost. To ensure we would only use spatial lagged features that possessed powerful information, we looked at the so-called Moran's I coefficient. This coefficient is a measure for spatial autocorrelation, which in simple terms, is a representation of how good it is possible to predict a feature with the knowledge of the value of the same feature in geographically neighboring areas.

In the future, we will create a Graph Neural Network that will be used to evaluate the impact of potential transportation policies on the NO2 levels. The graph nodes will consist of our dataset's different locations, each represented by a single row of the final dataset. The edges will represent the number of people traveling by a vehicle from one location to another in November 2019, a representative month. For instance, from 'Rojales' to 'Montesinos, Los y Algorfa' (two different nodes-locations in our dataset), 241 people traveled in November 2019. The goal here is to see what happens to the network if specific nodes are taken out. In other words, since most of NO2 is emitted by vehicles, what happens if we limit movements from specific locations?
As an ultimate goal, together with 300000 km/s, we plan to re-calibrate the model using daily data from the measurement stations or little tubes from 4sfera. This experiment can be even scaled using data from OSM (open street maps) to replicate the predictions in other cities outside Spain.

1.2 Data and methodologies

For this project, we have extracted data from multiple sources. Some of the data sets were publicly available, and others were given upon request. All the final datasets can be found in the Datasets folder in our Sync account.

- INE (Instituto Nacional de Estadística) data is all publicly available and found on their website. (INE, 2011)
- EEA (European Environmental Agency) data is all publicly available and found on their website. (EEA, 2019)
- WAQI (World Quality Air Index) data is all publicly available and found on their website. (WAQI, 2020)
- The datasets we had access from 300000 km/s are private but not confidential.

Our methodology was quite clear from the offset because we knew where 300.000km/s wanted us to go from the beginning. This matter meant that we primarily knew what kind of data we should gather and where we should get it from day one. Since we had four primary data sources, we split up the work per data source to ensure they were all adjusted, such they could be used in our final prediction model. Though some data proved challenging to get, we managed to ensure all the necessary data was in our possession. From this moment on, we needed to get very organized to make sure that the different codes and data would end up in places we did not want it to go.

By making sure all the workflows concerning the different data sources did not collide with one another, we could have a good structure in review sessions with 3000.000km/s. This organization was extra important when we started to work with geodata, as this was not Esteve's nor our area of expertise, but it is where mistakes are easy to make and be quite impactful. After all data was gathered, building the model and assessing lagged variables had to be done in a structured manner. This evaluation is the case because the choices to be made are sometimes subjective if a strict methodology is not followed.

1.3 Delivery

To ease the workload, we created five different notebooks, four to clean each data source (Areas_movements.ipynb, EEA.ipynb, INE.ipynb, WAQI.ipynb), and one to merge all the cleaned datasets (Datasets.ipynb). An extract of our code source from each notebook, but WAQI.ipynb is shown in section 2. In the Annexes, we included the WAQI notebook extract since we subsequently discarded it from our analysis as we had access to other sources more aligned with our objective. All the libraries used in this section are shown in the Annexes part under the WAQI section.

Getting all of these data sets was quite a rough process, where lots of problems arose. Especially getting the data from the Spanish national bureau of statistics was not clear cut at all. Luckily all our stakeholders delivered eventually. When our biggest challenge began, learning to work with data types unknown to us, such as geodata, by having just the internet and biweekly zoom call with 300.000km/s. We were thrilled that with the help of Pablo Martinez, the firm's CEO, we were able to finalize our datasets mid-august. Our learnings from A.I. 1 turned out to be true in our project. As a data scientist, 80% of the time will be cleaning and fixing the data, and the remaining 20% will do data science. Furthermore, A.I. 1 and 2 gave us the essential skills and knowledge base to build the models, which we did in the next part.

In Correlations_and_models.ipynb we built our ML model. We developed a Correlation matrix to see how features interact, and we removed those highly correlated. We then built a Random Forest Regressor to see the feature importance on the Y label (the NO2). We then iterated the process, binning features such as houses size per area - up to 60 m2 small, up to 105 m2 medium, rest large - or people by age per area - young, medium, old. We made sure that we closely followed the methodology, as learned at ESADE, a process where we kept doing sanity checks as often as possible to minimize the margin of error. Unfortunately, even though we tried our best, errors did creep into the dataset. At these points, we were lucky to have the support of 300.000km/s to point us at the made errors. This support taught us that sanity checks are of utmost importance in a project as big as this, but never to be trusted.

Before finalizing our model, we made sure we used all the power our dataset gave us. Accordingly, we built our spatial lagged features using Geoda, a geospatial analysis tool, and pysal, a library in python specifically aimed at geospatial analysis. As this was an area unknown to us, we were happy to get the knowledge of 300.000km/s on the subject. Geospatial analysis is a world on its own, and it is quite a niche. It requires a concrete knowledge base since geographical relation is, by definition, non-mathematical relations. This fact is actual since geography is a human construct by nature. However, the foundation is similar to any data science project, so the general rational applied in thinking with data was still very relevant.

In Stellar_Graph.ipynb, we have been building our network and trained with TensorFlow. Luckily A.I. 2 gave us a brief introduction into deep learning; however, GNN is still a complicated and new data science area. Due to the work's complexity, we did not fully finish the GNN before the 2nd of October. We plan to complete this last precious step in the next few months together with the data scientists of 300.000km/s.

1.4 Final results and considerations

Our final deliverable was a model which used the best combination of ‘normal’ and spatial-lagged feature to predict NO2 level in Spain. We started our initial search to the best possible model the 30+ features and ended up with a model that uses eight features to predict NO2 throughout Spain. Based on the Moran’s I score and multiple tryouts between different combinations of features being either spatially lagged or not. We arrive at a model that has an 88.8% accuracy of predicting NO2 levels throughout Spain. We found that the percentage of space used for residential buildings and the number of homes with a surface between 61 and 90 m2 were the most potent predictors of NO2 levels. Other notable predictors were houses with a surface between 45 and 60 m2 and the number of people between 0 and 25 living per square kilometer. We could predict N02 level with the precision we were able to, mostly based on residential information, which shows how much good city planning can matter for cities’ livability.

In our opinion, multiple sectors can leverage the results of our model. The first is the public sector, as urban planning matters concerning the question of pollution. By taking smart strategies to reduce traffic between places, cities might have more impact at a lower cost compared to routes taken right now. In the future, these conclusions might even get more support when we finish our GNN model. This model will give us information on what happens if we adjust particular traffic flows within the whole structure. An example can be building offices in Sant Cugat to reduce traffic inflow into Barcelona and improve Barcelona's air quality. This action contrasts with those taken nowadays, where politicians try to solve issues by taking measures where the pollution is too high.

The second is the business sector, given the granularity of the predictions. The real estate market might use our models to predict how air pollution will behave in the future. This is valuable information when making investment strategies or evaluating the current portfolio. More generally, businesses can use this information to plan their offices in the right locations to give them the possibility to market themselves as engaged businesses. This is not only what customers are demanding these days, but also of increasing interest to employees.

Finally, countries can use these models to check whether their more high-level pollution planning works according to their plan. In such a scenario, our predictions can be used to benchmark the areas in which particular pollution minimalizing measures have been taken to review their success. This will increase the time to market of successful ideas as it will take less time to confirm the results. Furthermore, it will enable a more rapid rollout of new ideas as a bad one will be killed sooner. All of this will save not only costs but with a bit of luck, and it might give an edge in saving the environment.

In conclusion, we want to thank Prof. Esteve Almirall for his great support and mentorship, Pablo Martinez and André Resende for their availability and collaboration, and all the professors in ESADE that taught us the best tools to do real Data Science.

Ps. All our notebooks and data can be accessed via this Link. The password is ESADE, the link will expire 12-11-2020. We hope you will enjoy our exciting project.

2 Data

2.3 INE - Instituto Nacional de Estadística

We used the data from the 2011 census by the INE - Instituto Nacional de Estadística. The information we gathered was granular, ranging from income divided by area, population by area and age, dwelling composition (building binned by square meters), and surface division (division of Spain by polygons representing areas and surface of each polygon). In our notebook we cleaned and explored the data, merging into one single dataset that we later used as part of our predictive features. Below the process is shown.

2.3.1 Data cleaning

2.3.1.1 Census

```
INE_01 = pd.read_csv('../INE/indicadores_seccion_censal_csv/C2011_ccaa01_Indicadores.csv', usecols = [1,2,3,4,5,6,7,8,9,10,114,115,116,117,124,125,126,127,128,129,130,131,132,133])
INE_02 = pd.read_csv('../INE/indicadores_seccion_censal_csv/C2011_ccaa02_Indicadores.csv', usecols = [1,2,3,4,5,6,7,8,9,10,114,115,116,117,124,125,126,127,128,129,130,131,132,133])
INE_03 = …….
```

```
INE_All = pd.concat([INE_01,INE_02,INE_03,INE_04,INE_05,INE_06,INE_07,INE_08,INE_09,INE_10,INE_11,INE_12,INE_13,INE_14,INE_15,INE_16,INE_17,INE_18,INE_19])
```

2.3.1.2 Locations

We combined all the different file locations into one large dataset. The code can be seen in the notebook, we only report the output below.

	CSEC	CDIS	CMUN	CPRO	Shape_Leng	Shape_area	Shape_len	geometry
0	1	1	1	1	34474.734278	1.551393e+07	34474.734278	MULTIPOLYGON (((-2.49116 42.85691, -2.49103 42...
1	2	1	1	1	8620.042319	4.410972e+06	8620.042319	POLYGON ((-2.49361 42.85699, -2.49361 42.85699...
2	1	1	2	1	47379.027701	3.535737e+07	47379.027700	MULTIPOLYGON (((-2.95561 42.98869, -2.95559 42...
3	2	1	2	1	31169.713203	3.873652e+07	31169.713203	POLYGON ((-2.89021 43.04566, -2.89014 43.04547...
4	3	1	2	1	4244.249418	8.494741e+05	4244.249418	POLYGON ((-3.00099 43.05210, -3.00185 43.05156...

2.3.1.3 Income

We combined all the files in the Income_data folder with a for loop to build the DataFrame Income which we then cleaned and explored. Part of the code we used is reported below, together with the output of the section.

```
import os
## CPRO 1
exc = pd.read_csv('../INE/Income_data/30851.csv', sep=';')
exc['Unidades territoriales'] = exc['Unidades territoriales'].astyp
e(str)
exc = exc[exc['Periodo']==2017]
exc = exc[exc['Indicadores de renta media'] == 'Renta media por per
sona']
exc['Total'] = exc['Total'].fillna(method='ffill')

## All the other provinces
Path = "../INE/Income_data/"
filelist = os.listdir(Path)
location = []
Income = pd.DataFrame()
for i in filelist:
    if i.endswith(".csv") and i != '30851.csv':
        with open(Path + i, 'r') as f:
            df = pd.read_csv(Path + i, sep=';')
            Income = pd.concat([Income, df], ignore_index=True)
## Final Income dataset
Income = pd.concat([Income, exc], ignore_index=True)
Income['Unidades territoriales'] = Income['Unidades territoriales']
.astype(str)
Income = Income[Income['Unidades territoriales'].str.contains('|'.j
oin(['sección']))]
Income= Income[Income['Periodo']==2017]...
```

	Total	CPRO	CMUN	CDIS	CSEC
0	10448.0	25	1	1	1
1	11666.0	25	2	1	1
2	11646.0	25	3	1	1
3	11140.0	25	3	1	2
4	11185.0	25	3	1	3

2.3.2 Data Processing

First, we changed the CRS to Lon Lat format from UTM to create centroids – centroids are better calculated with plain rather than spheric coordinates. We then used GeoSeries.centroid to calculate centroid of INE data, and matched those points with shapes.geojson data by summing up all data of the centroids within the multi-polygons of shapes['geometry']. The left part of the output is shown at the end of the code boxes.

```
INE_final_gpd = gpd.GeoDataFrame(INE_final, crs="EPSG:4326", geomet
ry= 'geometry')
```

```
INE_final_gpd['centroid'] =  INE_final_gpd['geometry'].centroid
INE_final_gpd = INE_final_gpd.drop(['geometry','Shape_area'],axis=1
)

INE_final_gpd = gpd.GeoDataFrame(INE_final_gpd, crs="EPSG:4326", ge
ometry= 'centroid')
INE_final_gpd.head()
```

	tot_pop	Male	Female	sub_16_age	16_to_64_age	64_more_age	tot_house	first_home	second_home	vacation_home	...	46_60_m2	61_75_m2	76_90_m2	91_105_m2	106_120_m2	121_150_m2
0	1460	760.0	695.0	170.0	930.0	360.0	1145.0	595.0	460.0	85.0	...	20.0	55.0	170.0	125.0	115.0	35.0
1	1350	700.0	655.0	150.0	870.0	335.0	710.0	550.0	145.0	15.0	...	90.0	25.0	60.0	150.0	70.0	105.0
2	765	325.0	445.0	70.0	590.0	105.0	405.0	295.0	20.0	95.0	...	0.0	0.0	140.0	100.0	0.0	0.0
3	1750	800.0	950.0	335.0	1140.0	275.0	780.0	650.0	15.0	115.0	...	0.0	0.0	215.0	100.0	110.0	0.0
4	1815	780.0	1035.0	470.0	1120.0	225.0	1100.0	610.0	40.0	450.0	...	0.0	60.0	250.0	160.0	65.0	0.0

2.4 EEA - European Environment Agency

These datasets consist of a massive pool of air stations data collected in 2019 in Spain. Some interesting features we noticed are the altitude at which the station is located, the measurements of O3 and NO2 (the first not used in the project), the time of measurement (a column showing the initial time and one showing the end time), and the location of each station (expressed in two columns, one for latitude and one for longitude). We had to combine different datasets in the notebook – both the metadata and the measurements. The process is shown below. Additionally, later on we had access to another dataset, a grid of Spain (1km x 1km) that we used to interpolate data.

2.4.1 Data preparation

2.4.1.1 EEA Data

```
'''
import os
import glob2
os.chdir("../EEA/data")
extension = 'csv'
all_filenames = [i for i in glob2.glob('*.{}'.format(extension))]
#combine all files in the list
combined_csv = pd.concat([pd.read_csv(f) for f in all_filenames])
combined_csv.to_csv( "EEA_total.csv", index=False, encoding='utf-8-
sig')
os.chdir('/Users/davidecallegaro/Sync/300.000kms/Notebooks')
os.getcwd()
'''
```

2.4.1.2 Meta Data

```
EEA_meta = pd.read_csv('../EEA/PanEuropean_metadata.csv',sep='\t')
EEA_meta = EEA_meta[(EEA_meta['Countrycode'] == 'ES')]
EEA_meta = EEA_meta.drop(['Countrycode','Namespace','Timezone','Sam
plingPoint','SamplingProces','Sample','MeasurementType','AirQuality
StationType','AirQualityStationNatCode','AirQualityStationEoICode',
'EquivalenceDemonstrated', 'BuildingDistance','KerbDistance','Measu
rementEquipment','AirQualityStationArea','InletHeight', 'Projection
','ObservationDateEnd','AirPollutantCode','ObservationDateBegin','A
irQualityNetwork'],axis=1)
EEA_meta = EEA_meta.drop_duplicates()
EEA_meta.sort_values(by='Altitude', ascending = False).head()
```

	AirQualityStation	Longitude	Latitude	Altitude
24017	STA_ES1982A	0.729556	42.051341	1570.0
18353	STA_ES0009R	-3.142500	41.274170	1360.0
24626	STA_ES2083A	-6.192600	43.069600	1253.0
18228	STA_ES0007R	-3.534170	37.237220	1230.0
19995	STA_ES1310A	2.214273	42.312078	1226.0

2.4.1.3 NO2 Data

```
EEA_final = EEA_meta.merge(EEA_tryout, on=['AirQualityStation'])
EEA_final.sort_values(by='DatetimeBegin').head()
```

	AirQualityStation	Concentration	DatetimeBegin	DatetimeEnd
1181345	STA_ES1672A	8.0	2017-01-01 00:00:00 +01:00	2017-01-01 01:00:00 +01:00
1181346	STA_ES1672A	9.0	2017-01-01 01:00:00 +01:00	2017-01-01 02:00:00 +01:00
1181347	STA_ES1672A	11.0	2017-01-01 02:00:00 +01:00	2017-01-01 03:00:00 +01:00
1181348	STA_ES1672A	8.0	2017-01-01 03:00:00 +01:00	2017-01-01 04:00:00 +01:00
1181349	STA_ES1672A	9.0	2017-01-01 04:00:00 +01:00	2017-01-01 05:00:00 +01:00

2.4.1.4 Merging all Datasets

```
EEA_tryout = pd.read_csv('../EEA/data/EEA_total.csv', sep=',')
EEA_tryout = EEA_tryout.drop(['Countrycode','Namespace','SamplingPo
int','Sample','SamplingProcess', 'AirQualityStationEoICode','AirPol
lutant','AirPollutantCode', 'AveragingTime','UnitOfMeasurement','Va
lidity','Verification','AirQualityNetwork'], axis=1)
EEA_tryout.sort_values(by='DatetimeBegin').head()
```

	AirQualityStation	Longitude	Latitude	Altitude	Concentration	DatetimeBegin	DatetimeEnd
2949307	STA_ES1672A	-2.619444	42.518333	480.0	8.0	2017-01-01 00:00:00 +01:00	2017-01-01 01:00:00 +01:00
2949308	STA_ES1672A	-2.619444	42.518333	480.0	9.0	2017-01-01 01:00:00 +01:00	2017-01-01 02:00:00 +01:00
2949309	STA_ES1672A	-2.619444	42.518333	480.0	11.0	2017-01-01 02:00:00 +01:00	2017-01-01 03:00:00 +01:00
2949310	STA_ES1672A	-2.619444	42.518333	480.0	8.0	2017-01-01 03:00:00 +01:00	2017-01-01 04:00:00 +01:00
2949311	STA_ES1672A	-2.619444	42.518333	480.0	9.0	2017-01-01 04:00:00 +01:00	2017-01-01 05:00:00 +01:00

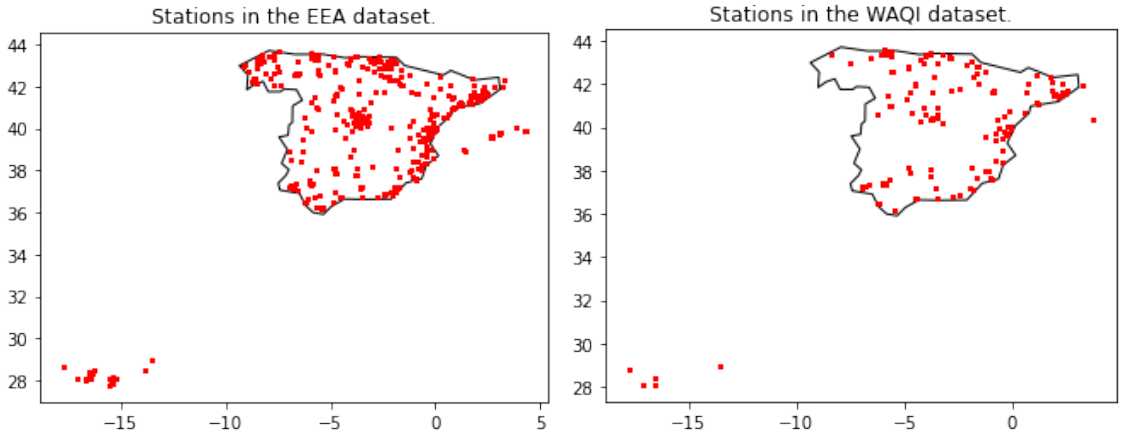
```
gdf_EEA = gpd.GeoDataFrame(EEA_final, geometry=gpd.points_from_xy(E
EA_final.Longitude, EEA_final.Latitude))
gdf_EEA = gdf_EEA.drop(['Longitude','Latitude'],axis=1)
gdf_EEA.head()
```

	AirQualityStation	Altitude	Concentration	DatetimeBegin	DatetimeEnd	geometry
0	STA_ES0001R	917.0	2.66	2019-01-25 01:00:00 +01:00	2019-01-25 02:00:00 +01:00	POINT (-4.35056 39.54694)
1	STA_ES0001R	917.0	1.91	2019-01-25 02:00:00 +01:00	2019-01-25 03:00:00 +01:00	POINT (-4.35056 39.54694)
2	STA_ES0001R	917.0	1.43	2019-01-25 03:00:00 +01:00	2019-01-25 04:00:00 +01:00	POINT (-4.35056 39.54694)
3	STA_ES0001R	917.0	1.12	2019-01-25 04:00:00 +01:00	2019-01-25 05:00:00 +01:00	POINT (-4.35056 39.54694)
4	STA_ES0001R	917.0	1.18	2019-01-25 05:00:00 +01:00	2019-01-25 06:00:00 +01:00	POINT (-4.35056 39.54694)

2.4.1.5 Plot

We imported the map of Spain from GeoPandas and plotted the station points on the map we downloaded from GeoPandas datasets to better visualize our data.

```
import geoplot
import mapclassify
world = gpd.read_file(gpd.datasets.get_path('naturalearth_lowres'))
spain = world[world['name'] == ('Spain')]
fig, ax = plt.subplots()
ax.set_aspect('equal')
spain.plot(ax=ax, color='white', edgecolor='black')
gdf_EEA.plot(ax=ax, marker='o', color='red', markersize=1)
plt.title("Stations in the EEA dataset.")

plt.show();
```



Compared to the previous graph there are many more stations in the EEA dataset.

2.4.2 Extra Data preparation

We had access to the grid dataset only subsequently, therefore we processed it in a second moment.

2.4.2.1 Grid Spain

```
grid = gpd.read_file("../EEA/2017Shapefile/Interpolation data 1 km grid/EEA_1kmgrid_2017.shp")
pd.set_option('display.max_columns', None)
grid["COUNTRY"].unique()
spain_grid = grid[grid['COUNTRY'] == 'ES']
spain_grid = spain_grid.drop(['COUNTRY','OBJECTID','CellCode', 'PM10_avg', 'PM10_p904', 'PM25_avg', 'dif_PM_avg', 'dif_PMp904', _PM25', 'PM10_avg_b', 'PM10avg_ut', 'PM25_avg_b', 'PM25avg_ut', 'POINT_X', 'POINT_Y','NO2_avg_b', 'dif_O3p932', 'dif_NO2avg','dif_O3_S35', 'dif_O3_S10', 'POPULATION','EofOrigin', 'NofOrigin', 'CODE'], axis=1)
```

From this data, we dropped all but the columns shown below. For each station, we had to drop Latitude and Longitude columns and create one column that combined both features (the point coordinate). The process is shown below.

	LAT	LON	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10	geometry
33280	-9.28496	42.8856	111.845	5387.27	2.744	0.0	0.0	11.975	22174.500000	POLYGON ((2760000.000 2399000.000, 2760000.000...
33281	-9.29662	42.9204	105.190	4304.51	2.655	0.0	0.0	12.170	21000.599609	POLYGON ((2760000.000 2403000.000, 2760000.000...
33718	-9.27014	42.8792	111.845	5387.27	2.775	0.0	0.0	11.954	22174.500000	POLYGON ((2761000.000 2398000.000, 2761000.000...
33719	-9.27305	42.8879	111.845	5387.27	0.942	0.0	0.0	11.968	22174.500000	POLYGON ((2761000.000 2399000.000, 2761000.000...
33720	-9.27596	42.8966	105.190	4304.51	0.541	0.0	0.0	11.999	21000.599609	POLYGON ((2761000.000 2400000.000, 2761000.000...

Creating the coordinate (LAT, LON)

```
spain_grid_up = gpd.GeoDataFrame(spain_grid, geometry=gpd.points_from_xy(spain_grid.LON, spain_grid.LAT))
spain_grid_up = spain_grid_up.drop(['LAT','LON'],axis=1)
spain_grid_up.head()
```

	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10	geometry
33280	111.845	5387.27	2.744	0.0	0.0	11.975	22174.500000	POINT (42.88560 -9.28496)
33281	105.190	4304.51	2.655	0.0	0.0	12.170	21000.599609	POINT (42.92040 -9.29662)
33718	111.845	5387.27	2.775	0.0	0.0	11.954	22174.500000	POINT (42.87920 -9.27014)
33719	111.845	5387.27	0.942	0.0	0.0	11.968	22174.500000	POINT (42.88790 -9.27305)
33720	105.190	4304.51	0.541	0.0	0.0	11.999	21000.599609	POINT (42.89660 -9.27596)

2.5 Areas and movements

We had access to a few datasets from 300000km/s, one with all the geometric information of Spain (areas and perimeter of polygons in which Spain is divided), another of the movements of people from one location of the map to another during November 2019, and another with the building density information of each area. The entire content of the datasets

is property of 300000km/s. The datasets were cleaned, so we only explored them. Below the content of the notebook in which we explore the first two datasets is shown.

2.5.1 Data exploration

2.5.1.1 Areas

This dataset contains Spain divided in geospatial areas, including population, perimeters of the areas, surface of each district's areas, and name.

```
noviembre_pols = gpd.read_file('../Pablo/noviembre_pols.geojson')
noviembre_pols = noviembre_pols.drop(['rowid','objectid_1','id_grupo'],axis=1)
noviembre_pols.head()
```

	objectid	sum_pob_as	shape_leng	name_celda	shape__area	shape__length	geometry
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...
1	1589	6609	176455.378779	Belalcázar y otros municipios	7.979909e+08	176455.378779	MULTIPOLYGON (((-4.86709 38.65973, -4.86693 38...
2	261	5508	93715.435696	Navarrés y otros municipios	2.971162e+08	93715.435696	MULTIPOLYGON (((-0.66458 39.10707, -0.66041 39...
3	1257	5161	98840.385938	Casavieja y otros municipios	3.121015e+08	98840.385938	MULTIPOLYGON (((-4.89526 40.41188, -4.89504 40...
4	3178	16920	21437.000417	Palma de Mallorca (SCD Number 5-B)	6.731973e+06	21437.000417	MULTIPOLYGON (((2.70288 39.54769, 2.70390 39.5...

2.5.1.2 Movements

This dataset shows the streams of people from one site to another in the November 2019.

```
noviembre = gpd.read_file('../Pablo/noviembre.geojson')
noviembre = noviembre.drop(['rowid','objectid'],axis=1)
noviembre.head()
```

	celda_destino	celda_origen	flujo	n_destino	nombre_celda_destino	nombre_celda_origen	p_pob_casa	p_pob_sale	pob_casa	pob_resid	pob_sale	geometry
0	002A	001A	107	101	Formentera del Segura y otros municipios	Montesinos, Los y Algorfa	61,77	38,23	4882	7903	3021	LINESTRING (-0.77723 38.04169, -0.74723 38.09734)
1	023A	001A	463	101	Almoradí	Montesinos, Los y Algorfa	61,77	38,23	4882	7903	3021	LINESTRING (-0.77723 38.04169, -0.79547 38.09948)
2	058A	001A	983	101	Rojales	Montesinos, Los y Algorfa	61,77	38,23	4882	7903	3021	LINESTRING (-0.77723 38.04169, -0.71885 38.07398)
3	061A	001A	213	101	San Miguel de Salinas	Montesinos, Los y Algorfa	61,77	38,23	4882	7903	3021	LINESTRING (-0.77723 38.04169, -0.80311 37.97484)
4	091A	001A	167	101	Orihuela (distrito 05)	Montesinos, Los y Algorfa	61,77	38,23	4882	7903	3021	LINESTRING (-0.77723 38.04169, -0.85410 37.97972)

2.6 Datasets

2.6.1 EEA Data

```
spain_grid = gpd.read_file('../Datasets/spain_grid.geojson')
spain_grid.head()
```

	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10	geometry
0	111.845	5387.27	2.744	0.0	0.0	11.975	22174.500000	POINT (42.88560 -9.28496)
1	105.190	4304.51	2.655	0.0	0.0	12.170	21000.599609	POINT (42.92040 -9.29662)
2	111.845	5387.27	2.775	0.0	0.0	11.954	22174.500000	POINT (42.87920 -9.27014)
3	111.845	5387.27	0.942	0.0	0.0	11.968	22174.500000	POINT (42.88790 -9.27305)
4	105.190	4304.51	0.541	0.0	0.0	11.999	21000.599609	POINT (42.89660 -9.27596)

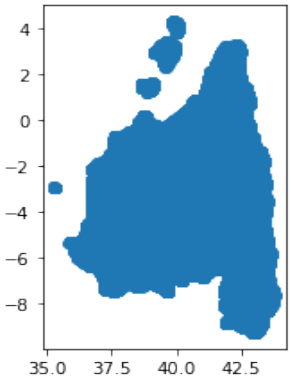
```
spain_grid.crs
```

```
<Geographic 2D CRS: EPSG:4326>
Name: WGS 84
Axis Info [ellipsoidal]:
Lat[north]: Geodetic latitude (degree)
Lon[east]: Geodetic longitude (degree)
Area of Use:
name: World
bounds: (-180.0, -90.0, 180.0, 90.0)
Datum: World Geodetic System 1984
Ellipsoid: WGS 84 - Prime Meridian: Greenwich
```

We realized that the coordinates of Spain map were upside down. Therefore, we built a function that swapped the coordinates.

```
spain_grid.plot()
```

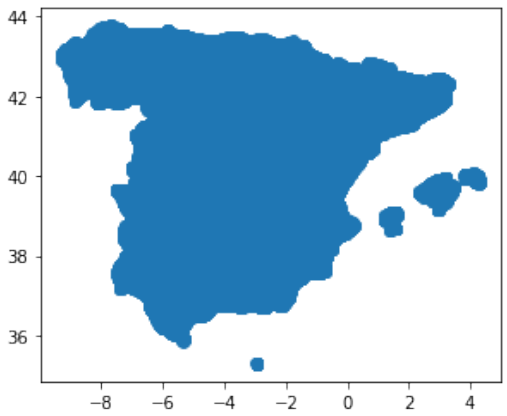
Spain grid coordinates



```
def swap_xy(geom):
    if geom.is_empty:
        return geom

    if geom.has_z:
        def swap_xy_coords(coords):
            for x, y, z in coords:
                yield (y, x, z)

    else:
        def swap_xy_coords(coords):
            for x, y in coords:
                yield (y, x)

    # Process coordinates from each supported geometry type
    if geom.type in ('Point', 'LineString', 'LinearRing'):
        return type(geom)(list(swap_xy_coords(geom.coords)))
    elif geom.type == 'Polygon':
        ring = geom.exterior
        shell = type(ring)(list(swap_xy_coords(ring.coords)))
        holes = list(geom.interiors)
        for pos, ring in enumerate(holes):
            holes[pos] = type(ring)(list(swap_xy_coords(ring.coords
)))
        return type(geom)(shell, holes)
    elif geom.type.startswith('Multi') or geom.type == 'GeometryCol
lection':
        # Recursive call
        return type(geom)([swap_xy(part) for part in geom.geoms])
    else:
        raise ValueError('Type %r not recognized' % geom.type)
```

```
spain_grid.geometry = spain_grid.geometry.map(swap_xy)
spain_grid.plot()
```

Spain grid coordinates



2.6.2 INE Data

```
INE = gpd.read_file('../Datasets/INE_final_gpd.geojson')
INE.head()
```

	tot_pop	Male	Female	sub_16_age	16_to_64_age	64_more_age	tot_house	first_home	second_home	vacation_home	less_30_m2
0	1460	760.0	695.0	170.0	930.0	360.0	1145.0	595.0	460.0	85.0	5.0
1	1350	700.0	655.0	150.0	870.0	335.0	710.0	550.0	145.0	15.0	0.0
2	765	325.0	445.0	70.0	590.0	105.0	405.0	295.0	20.0	95.0	0.0
3	1750	800.0	950.0	335.0	1140.0	275.0	780.0	650.0	15.0	115.0	0.0
4	1815	780.0	1035.0	470.0	1120.0	225.0	1100.0	610.0	40.0	450.0	0.0

2.6.3 Shapes Spain

```
shapes = gpd.read_file('../Datasets/shapes.geojson')
shapes.head()
```

	objectid	sum_pob_as	shape_leng	name_celda	shape__area	shape__length	geometry
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((37.64752 -0.90576, 37.64680 -0...
1	1589	6609	176455.378779	Belalcázar y otros municipios	7.979909e+08	176455.378779	MULTIPOLYGON (((38.65973 -4.86709, 38.65867 -4...
2	261	5508	93715.435696	Navarrés y otros municipios	2.971162e+08	93715.435696	MULTIPOLYGON (((39.10707 -0.66458, 39.10254 -0...
3	1257	5161	98840.385938	Casavieja y otros municipios	3.121015e+08	98840.385938	MULTIPOLYGON (((40.41188 -4.89526, 40.41148 -4...
4	3178	16920	21437.000417	Palma de Mallorca (SCD Number 5-B)	6.731973e+06	21437.000417	MULTIPOLYGON (((39.54769 2.70288, 39.54661 2.7...

2.6.4 Merges

We merge by intersecting the geolocation coordinates of the points with the polygon areas of the other dataset. If a point in 'spain_grid' is found in ‘shapes’, then the two rows are merged. Then, we grouped the merged dataset (x) by ID, polygon and name to obtain aggregate data.

```
shapesVspain = gpd.sjoin(shapes, spain_grid, how="inner", op='inter
sects')
shapesVspain.head()
```

	objectid	sum_pob_as	shape_leng	name_celda	shape__area	shape__length	geometry	index_right	O3_p932	O3_SOMO35
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...	386920	124.477	8583.16
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...	387567	124.477	8583.16
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...	388215	124.274	8567.95
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...	386274	119.926	7714.50
0	2585	11395	53268.077784	Cartagena (distrito 05)	6.457586e+07	53268.077784	MULTIPOLYGON (((-0.90576 37.64752, -0.90567 37...	386921	124.477	8583.16

```
x = shapesVspain.groupby('objectid').mean()
x = x.drop('index_right', axis = 1)
x.head()
```

objectid	sum_pob_as	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut
1	7682.0	35771.338209	5.101956e+07	35771.338210	120.717098	8101.700588	8.572569	0.094363	0.027648	17.526804
2	14354.0	63295.634626	8.100284e+07	63295.634626	125.605278	7048.907215	9.247557	0.085032	0.010408	25.948532
3	5071.0	120439.774794	2.772914e+08	120439.774794	125.485913	8027.057319	7.279243	0.022464	0.006684	21.436130
4	6756.0	262576.084345	8.967069e+08	262576.084345	118.838681	7949.650277	4.458692	0.009230	0.006449	21.537608
5	6218.0	18204.294363	6.805165e+06	18204.294363	118.133429	7308.860000	13.053857	0.763214	0.035393	17.339571

```
y = x.merge(shapesVspain[['objectid','geometry','name_celda']], lef
t_index = True , right_on='objectid')
```

```
y = y.drop_duplicates()
y.head()
```

	sum_pob_as	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut
1038	7682.0	35771.338209	5.101956e+07	35771.338210	120.717098	8101.700588	8.572569	0.094363	0.027648	17.526804
2052	14354.0	63295.634626	8.100284e+07	63295.634626	125.605278	7048.907215	9.247557	0.085032	0.010408	25.948532
346	5071.0	120439.774794	2.772914e+08	120439.774794	125.485913	8027.057319	7.279243	0.022464	0.006684	21.436130
1483	6756.0	262576.084345	8.967069e+08	262576.084345	118.838681	7949.650277	4.458692	0.009230	0.006449	21.537608
1650	6218.0	18204.294363	6.805165e+06	18204.294363	118.133429	7308.860000	13.053857	0.763214	0.035393	17.339571

```
z = gpd.GeoDataFrame(y, crs="EPSG:4326", geometry='geometry')
pd.set_option('display.max_columns', None)

df = gpd.sjoin(z, INE, how="inner", op='intersects')
df = df.drop(['index_right','objectid'], axis = 1)

df['Total_income_x_tot_pop'] = df.tot_pop * df.Total

df.head()
```

	sum_pob_as	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut
1038	7682.0	35771.338209	5.101956e+07	35771.338210	120.717098	8101.700588	8.572569	0.094363	0.027648	17.526804
1038	7682.0	35771.338209	5.101956e+07	35771.338210	120.717098	8101.700588	8.572569	0.094363	0.027648	17.526804
1038	7682.0	35771.338209	5.101956e+07	35771.338210	120.717098	8101.700588	8.572569	0.094363	0.027648	17.526804
2052	14354.0	63295.634626	8.100284e+07	63295.634626	125.605278	7048.907215	9.247557	0.085032	0.010408	25.948532
2052	14354.0	63295.634626	8.100284e+07	63295.634626	125.605278	7048.907215	9.247557	0.085032	0.010408	25.948532

```
group_df_right = df.iloc[:,12:].groupby(df.index).sum()
group_df_right.head()
```

	tot_pop	Male	Female	sub_16_age	16_to_64_age	64_more_age	tot_house	first_home	second_home	vacation_home	less_30_m2
0	8930	4695.0	4230.0	1520.0	5825.0	1585.0	4000.0	3245.0	170.0	580.0	0.0
1	6285	3090.0	3190.0	810.0	3675.0	1800.0	4370.0	2545.0	1105.0	720.0	0.0
2	5885	3010.0	2880.0	840.0	3770.0	1285.0	4055.0	2375.0	595.0	1085.0	0.0
3	5740	3050.0	2700.0	580.0	3245.0	1925.0	8565.0	2595.0	5210.0	760.0	35.0
4	17260	9020.0	8235.0	2920.0	12130.0	2210.0	9425.0	7335.0	955.0	1120.0	0.0

We created the Income column as total income of an area divided by the population of that area.

```
group_df_left = df.iloc[:,:13].groupby(df.index).mean()
group_df_left.head()
```

	sum_pob_as	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut
0	11395.0	53268.077784	6.457586e+07	53268.077784	123.945625	8542.565937	8.057984	0.115195	0.033375	18.244891
1	6609.0	176455.378779	7.979909e+08	176455.378779	119.989637	7825.892713	6.122181	0.006209	0.004144	24.218680
2	5508.0	93715.435696	2.971162e+08	93715.435696	118.637381	7915.287211	3.331214	0.011616	0.000000	12.670469
3	5161.0	98840.385938	3.121015e+08	98840.385938	117.125316	7001.497636	3.114153	0.025687	0.009524	20.451901
4	16920.0	21437.000417	6.731973e+06	21437.000417	108.150000	6171.693333	13.050167	0.902500	0.056932	22.012667

```
group_df = group_df_left.merge(group_df_right, left_index = True ,
right_index = True)
group_df = group_df.drop(['sum_pob_as', 'Total'], axis = 1)

final = group_df.merge(df[['geometry','name_celda']], left_index =
True , right_index = True)
final = final.drop_duplicates()
final_gdf = gpd.GeoDataFrame(final, crs="EPSG:4326", geometry='geom
etry')
final.head()
```

	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10
0	53268.077784	6.457586e+07	53268.077784	123.945625	8542.565937	8.057984	0.115195	0.033375	18.244891	26102.251343
1	176455.378779	7.979909e+08	176455.378779	119.989637	7825.892713	6.122181	0.006209	0.004144	24.218680	24994.937778
2	93715.435696	2.971162e+08	93715.435696	118.637381	7915.287211	3.331214	0.011616	0.000000	12.670469	25397.019438
3	98840.385938	3.121015e+08	98840.385938	117.125316	7001.497636	3.114153	0.025687	0.009524	20.451901	24468.385733
4	21437.000417	6.731973e+06	21437.000417	108.150000	6171.693333	13.050167	0.902500	0.056932	22.012667	22980.366862

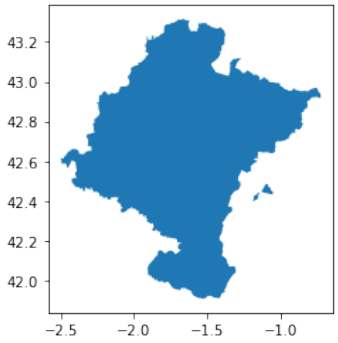
2.6.5 Finalize the Dataset

After some discussions, we realized that Navarra lacked census data, so we decided to drop it from our final dataset.

2.6.5.1 Navarra region

```
Navarra = gpd.read_file("../INE/REFERE_Pol_Navarra/REFERE_Pol_Navar
ra.shp")
Navarra = Navarra.to_crs("EPSG:4326")

Navarra.plot()
```

Navarra coordinates



2.6.5.2 Final data

```
df = gpd.read_file('../Datasets/final_gdf.geojson')
df_without_navarra = gpd.overlay(df, Navarra, how='difference')
build_density = gpd.read_file('../builtdensity/builtdensity.shp')
build_density = build_density.drop(['rowid','id','nombre_cel','nombre_cel','p_pob_casa','p_pob_sale','pob_resid','OGC_FID','m.celda_or','pob_casa','pob_sale','pk'], axis = 1)
build_density = build_density.dropna()
build_density.head()
```

	srf_tot	srf_housin	geometry
1	2808866.0	2067231.0	POLYGON ((2.15903 41.50740, 2.15904 41.50738, ...
2	1710586.0	565717.0	MULTIPOLYGON (((1.16553 41.13831, 1.16277 41.1...
3	2117162.0	1748717.0	POLYGON ((-15.64948 28.16497, -15.64937 28.164...
4	3160944.0	1144317.0	POLYGON ((-15.66313 27.91201, -15.66324 27.911...
5	2198805.0	1574528.0	POLYGON ((-0.38336 39.57589, -0.38336 39.57589...

```
build_density = build_density.to_crs(epsg = 3035)
build_density['centroid'] =  build_density['geometry'].centroid
build_density = build_density.drop(['geometry'],axis=1)
build_density = gpd.GeoDataFrame(build_density, crs="EPSG:4326", geometry= 'centroid')
```

```
build_density = build_density.to_crs(epsg = 4326)
ultimate_move_df = gpd.sjoin(df_without_navarra, build_density, how="inner", op='intersects')
ultimate_move_df = ultimate_move_df.drop(['index_right'], axis = 1)
ultimate_move_df.head()
```

	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10
0	53268.077784	6.457586e+07	53268.077784	123.945625	8542.565937	8.057984	0.115195	0.033375	18.244891	26102.251343
1	176455.378779	7.979909e+08	176455.378779	119.989638	7825.892713	6.122181	0.006209	0.004144	24.218680	24994.937778
2	93715.435696	2.971162e+08	93715.435696	118.637381	7915.287211	3.331214	0.011616	0.000000	12.670469	25397.019438
3	98840.385938	3.121015e+08	98840.385938	117.125316	7001.497636	3.114153	0.025687	0.009524	20.451901	24468.385733
4	21437.000417	6.731973e+06	21437.000417	108.150000	6171.693333	13.050167	0.902500	0.056932	22.012667	22980.366862

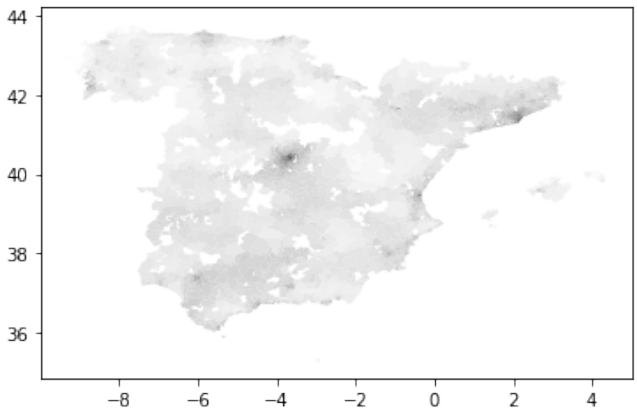
```
ultimate_move_df = gpd.sjoin(ultimate_move_df, nodes_l, how="inner"
, op='intersects')
ultimate_move_df = ultimate_move_df.drop(['index_right'], axis = 1)
ultimate_move_df[ultimate_move_df.name_celda == 'Madrid (SCD El Gol
oso-El Pardo-Fuentelarreina)']
```

	shape_leng	shape__area	shape__length	O3_p932	O3_SOMO35	NO2_avg	weight_urb	weight_tr	NO2_avg_ut	O3_S10	tot_pop	Male	Female	sub_16_age	16_to_64_age
1464	96824.310651	2.156116e+08	96824.310651	132.023402	8364.525561	12.289407	0.048879	0.012599	27.492126	25211.949976	21285	10320.0	10930.0	5075.0	13825.0

```
ultimate_move_df.plot(column='NO2_avg', cmap='gist_yarg', linewidth
=0.8)
```

```
movements = gpd.read_file('../Datasets/movements.geojson')
nodes = movements.copy(deep = True)
nodes_l = nodes.drop([ 'nombre_celda_origen','celda_destino', 'celd
a_origen', 'flujo', 'n_destino', 'nombre_celda_destino', 'p_pob_cas
a','p_pob_sale', 'pob_resid', 'geometry'], axis = 1)
nodes_l['geometry'] = movements.geometry.boundary.explode()[:,0]
nodes_l = nodes_l.drop_duplicates()
```

NO2 quantity by region



Export file

```
ultimate_move_df.to_file("../Datasets/ultimate_move_df.geojson", dr
iver='GeoJSON')
ultimate_move_df.to_file("../Datasets/ultimate_move_df.shp")
```

3 Machine Learning Model

Our model consisted of predicting the NO2 levels (Y) from other features (X), a supervised learning technique. Supervised learning is the machine learning task of inferring a function from labeled training data. The training data consist of a set of training examples. To achieve that, we used a combination of correlation matrices, random forest regressors, and special autocorrelation obtained from lagged features obtained from our initial set of features. We show an extract of our notebook code below.

3.1 Main concepts

3.1.1 Spatial autocorrelation

Spatial autocorrelation measures the relative proximity of objects with other objects. This concept violates a fundamental assumption of general statistical correlation, according to which each variable is independent of any other variable. Instead, objects that are close to each other spatially will tend to have similar characteristics, while further apart objects are more likely to have different characteristics. Spatial autocorrelation is generally classified as positive, negative, or zero. Positive autocorrelation is when objects are grouped, and similar values appear close to each other. Negative spatial autocorrelation results in a scattered model, in which different values are close to each other, and similar values are far apart. Zero spatial autocorrelation is a random model; there is no relationship between space-based variables.

3.1.2 Rook vs. queen contiguity

Two options are available — rook and queen — their names come from the movements of chess pieces. The rook can move only to polygons that share a border of some length with its current polygon. This behavior is to be seen in terms of geographical adjacency, where the areas that only share a single touchpoint are not considered neighbors. In rook contiguity, if areas only share a single touchpoint, they are not considered first order spatially related. Consequently, rook contiguity is a more stringent definition of geographical neighborhood relation. Rook requires a shared border that touches are more than just one point; some length is required. Queen contiguity needs no specific length of the shared border; it can be a single point. This is why we chose Queen contiguity to make sure we did not miss valuable information regarding neighborhood relations.

3.1.3 Regression

Regression is used to explain spatial patterns and investigate their underlying structures and drivers. A regression always involves a target, also known as the dependent variable, and multiple explanatory variables, known as independent variables. The strength of the correlation between the target and the independent variables determines the quality of the regression. The target variable describes the pattern or process which needs to be understood. In our case, this is the NO2 level in different areas. In our case, the independent variables all the features we got from the four datasets are used to describe this pattern. When considering all the individual independent variables together, one can represent pattern previously unknown patterns quite well. This representation is done using the fact that all these variables have a different positive or negative correlation with the target variable. When taking all of the correlation to collaborate, the independent variables' predictive power can become quite substantial. Regression models have an increased complexity compared to standard linear regression as they take into account

p-value, R^2, and the estimation error to determine what model most accurately describes the pattern of the target variable.

3.1.4 Decision trees

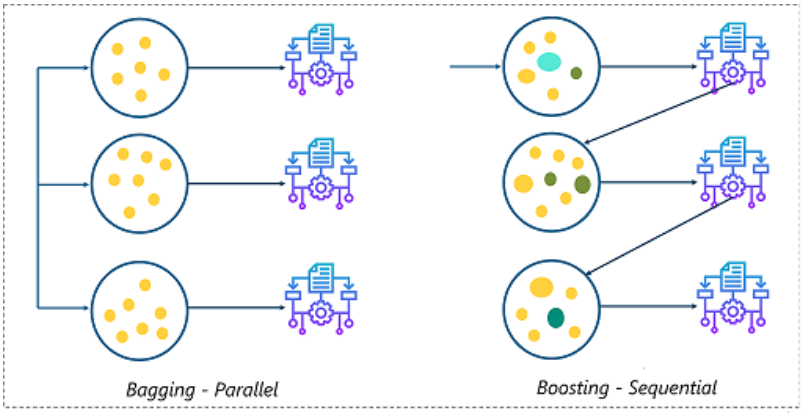
A decision tree is a supervised machine learning algorithm mainly used for Regression and Classification. (Sharma, 2020) It breaks down a data set into smaller and smaller subsets while at the same time an associated decision tree is incrementally developed. The final result is a tree with decision nodes and leaf nodes. (Tyagi, 2018) A decision tree can handle both categorical and numerical data. The term Classification and Regression Tree (CART) analysis is an umbrella term used to refer to both of the above procedures. Ensemble learning has many types, but two more popular ensemble learning techniques are mentioned below.

3.1.4.1 Bagging

The scope of bagging is implementing similar machine learners on small sample populations and then considering each prediction's average. Bagging works effectively in models with high variance (overfitting) or if many observations overlap during the sampling of train data using many single learners, all of the same size, operating on the same algorithm. However, it does not cope well when there is a high bias (underfitting). Indeed, given its structure, it ignores the models with the highest and the lowest result.

3.1.4.2 Boosting

Boosting is different from bagging in that it operates sequentially rather than in parallel: in each iteration of the learner, the model's weights are adjusted based on the last classification. In general, this practice decreases the bias error and copes with underfitting, building robust predictive models. The net error is revaluated in each learning step. Nevertheless, boosting performs poorly in overfitted or high variance models. Besides, time and computation can be much larger than bagging.



3.1.4.3 What is a Random Forest?

Random forest is a versatile machine learning method capable of performing regression, classification, dimensionality reduction, missing values, and outlier values. It is a particular type of ensemble learning method, where a group of weak models is combined to form a robust model. The random forest starts with a standard machine learning technique called a "decision tree," which, in ensemble terms, corresponds to a weak learner. In a decision tree, the input

data enters at the top, and, as it traverses down the tree, it gets bucketed into smaller and smaller sets. In Random Forest, multiple trees grow instead of a single tree, and each tree gives a classification based on attributes. In the classification, the result chosen has the highest number of votes, whereas, in a regression, the result is equal to the average of all the outputs by different trees.

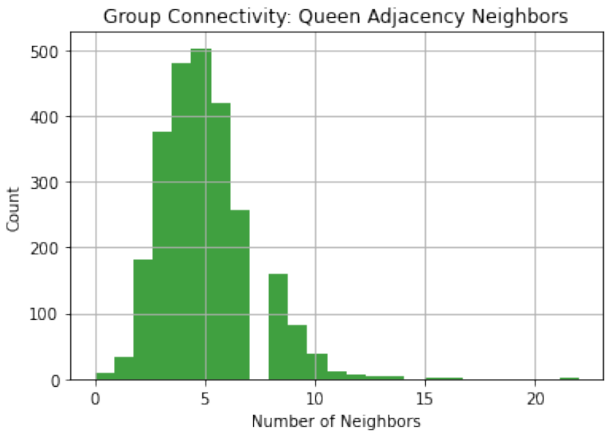
3.2 Model

3.2.1 Queen matrix and Spatial weights

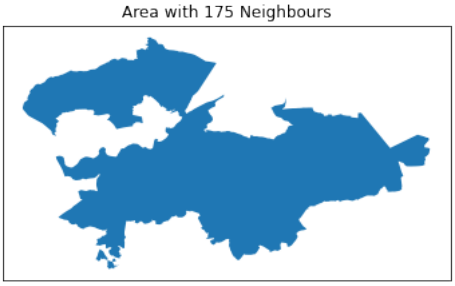
```
geo = gpd.read_file('../Datasets/ultimate_move_df.shp')
queen_w = Queen.from_shapefile("../Datasets/ultimate_move_df.shp")
n, bins, patches = plt.hist(queen_w.cardinalities.values(), 25, fac
ecolor='green', alpha=0.75)
```

Identifying Unusual Block Groups Based on Neighbor Counts

Now that the adjacency matrix is built, we can examine some characteristics of the links formed. Below we plot a histogram of the number of neighbors. As you can see, the median number of neighbors for these units is somewhere around 50. Two oddities that we can see here are that there is one block group with 0 neighbors and one block group with 175 neighbors. We examine these further below.



We can easily identify the “island” block group by selecting for the block group with 0 neighbors. Given that this is not a typical area within Spain, we will drop it from the analysis. However, in general, areas with no neighbors could still be included in a spatial regression model, although they would receive no spatial weight and therefore would be defined as having no relationship to any other geographic units. From the histogram you can also see that there is a huge outlier - a block group that has 175 neighbors. We plot it below.

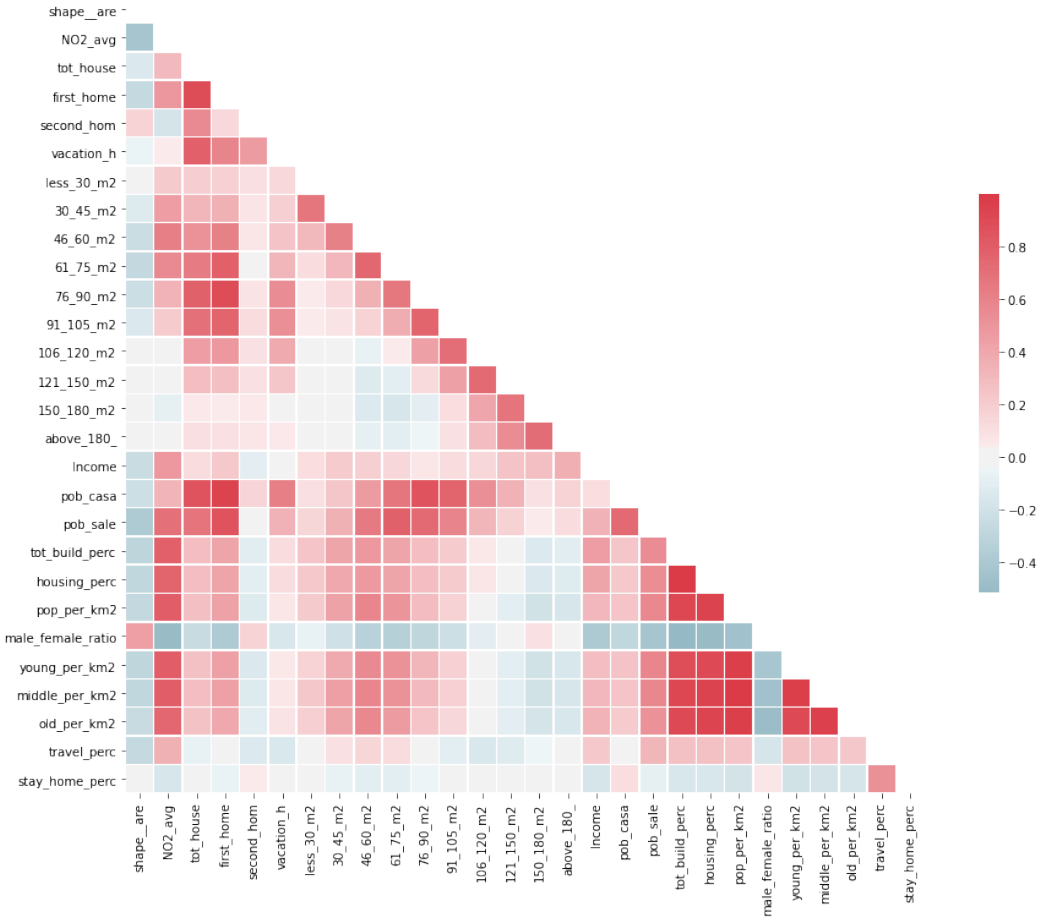


From now on we work only with data without islands.

3.2.2 New Dataset

We do some feature engineering to reduce the noise of features: male/female in a certain area; residential building density per area; young, adult, and old per area; percentage of people that leave and arrive. First we plotted a correlation matrix to view which features to discard based on high correlation, then we ran the random forest regressor to see the accuracy and select the best explanatory features.

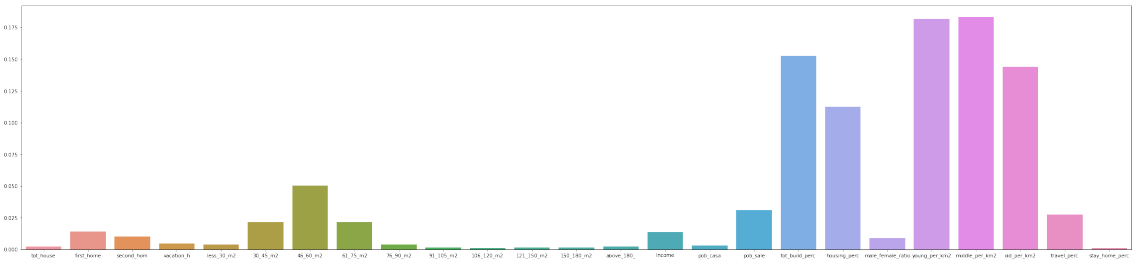
Correlation matrix



```
seed=7
kfold=KFold(n_splits=10, random_state=seed, shuffle = True)
num_trees=100
num_features=5
max_depth = 5
model=RandomForestRegressor(n_estimators=num_trees, max_features=nu
m_features,max_depth=max_depth, random_state=seed)
model.fit(X,NO2_avg)
results_NO2_avg=cross_val_score(model, X, NO2_avg, cv=kfold)

print(f'Random Forest - Accuracy {results_NO2_avg.mean()*100:.3f}%
std {results_NO2_avg.std()*100:3f}%')
res_w1["Res"]=results_NO2_avg
res_w1["Type"]="Random Forest"
resall=pd.concat([resall,res_w1], ignore_index=True)
```

```
Random Forest - Accuracy 85.746% std 0.444071%

Average NO2 level: 12.847696532150087

Expected error range: [11.856271169278983, 13.83912189502119]
```

Feature importance – Higher is better

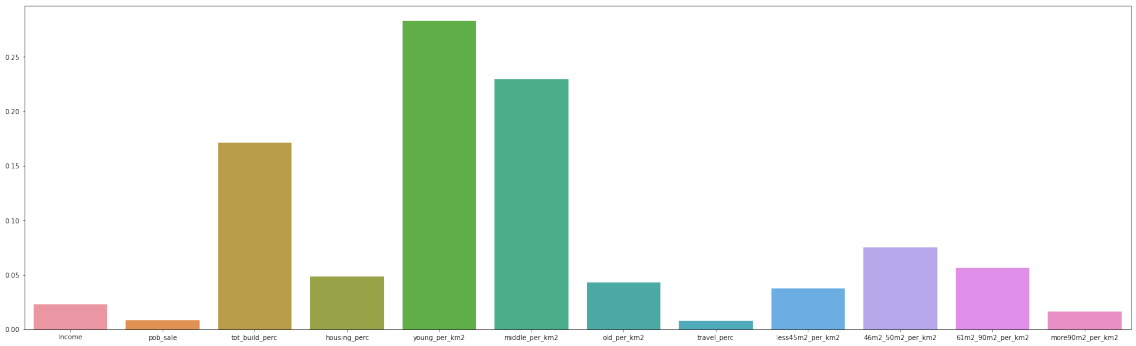


We iterate the model with new features, which are the result of binning features that seems irrelevant when taken into account separately. This mainly meant we had to bin some of the data that displays the surface of residential houses.

3.2.3 New dataset (2)

We performed another iteration, this time binning house size in 4 buckets too.

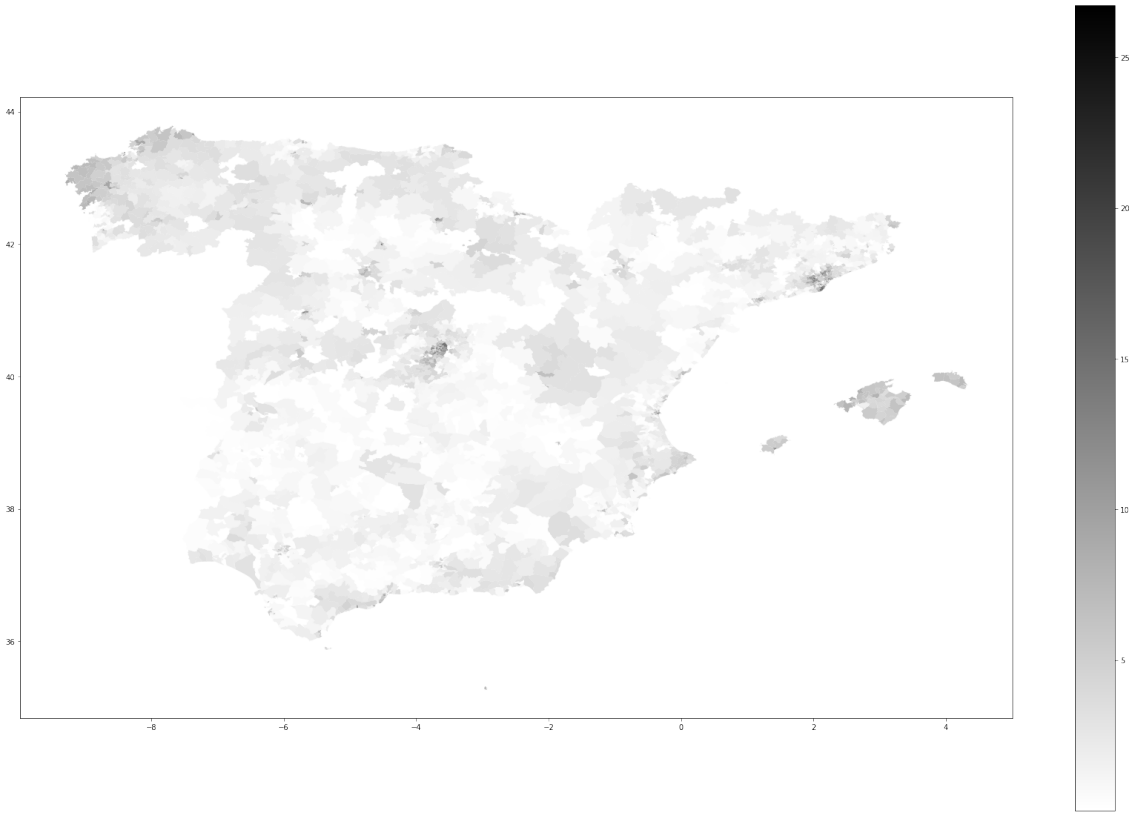
Feature importance – Higher is better



In the previous model, total population per km2 accounted most of the weight, followed by the youngest segment of population. The density of building per km2 did not play a relevant role. In the adjusted model, the results are somehow consistent, but now building density plays a major role in the prediction.
To give a better visual representation of the results from the model above, we measured the absolute prediction difference and plotted this. We saw that especially the densely populated areas were reason of concern. To counter this we introduced spatial lag, that is thought to be strong in regions where spatial autocorrelation is high like cities.

Map accuracy

Map accuracy – darker is less accurate

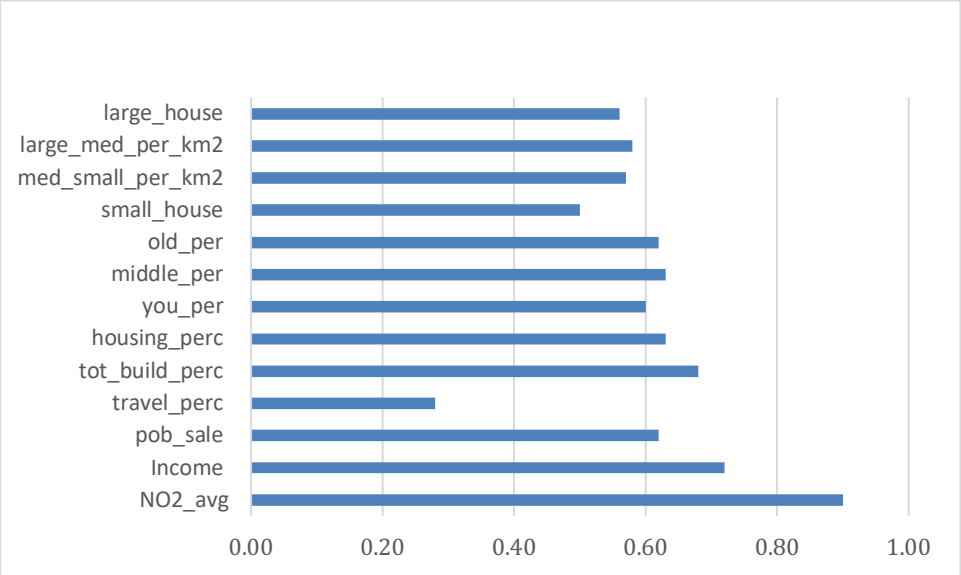


3.2.4 Lagged spatial regression

Here we want to add very useful but yet unused data, spatial information. As it is unknown which features would increase in explanatory value when accounting for spatial lag. This was the first thing that needed to be sorted out. The degree to which a feature has a spatial relation is explained by the Moran's I statistic. A zero implies there is no extra information to be added by using spatial lag. A one on the other hand implies that knowing the neighboring values of a feature gives an almost 100% certainty of the value of the feature at hand. Later on, we would decide that the cutoff point to include a feature as spatial lagged was a Moran I of around 0.6. Values close to 0.6 were all individually tested to check for the best performance.

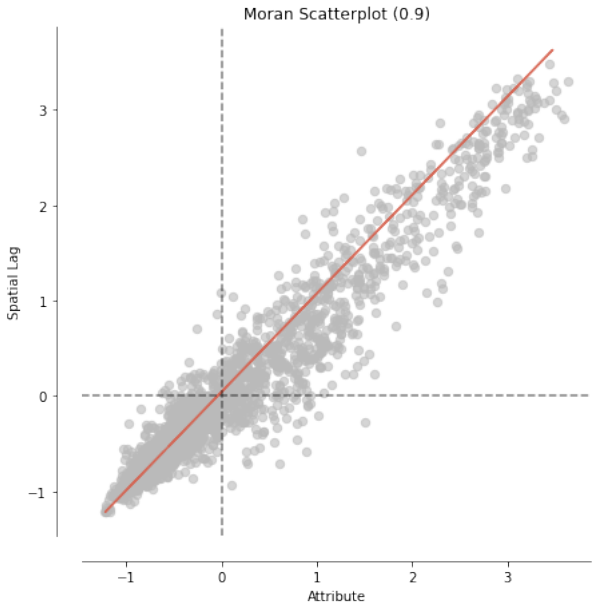
	income_lag	pob_sale_lag	travel_perc_lag	tot_build_perc_lag	housing_perc_lag	young_per_km2_lag	middle_per_km2_lag	old_per_km2_lag
0	8049.013547	2152.833333	0.199463	0.019723	0.010799	18.371270	68.863829	18.055824
1	9151.277970	3057.000000	0.221474	0.016318	0.007816	15.028485	50.776721	9.051079
2	11020.721722	3237.666667	0.284120	0.043781	0.026579	36.338145	170.389794	48.901862
3	9129.130428	1617.416667	0.227910	0.004075	0.002256	2.235243	9.863729	3.518146
4	13079.713108	4625.000000	0.323486	0.100756	0.051589	153.415612	502.060415	99.827321

Spatial auto correlations (Moran I) of features.



Moran autocorrelation scatterplot of NO2 (Moran I of 0.9)

```
from esda.moran import Moran
moran_Y  = Moran(Y , queen_w)
from splot.esda import moran_scatterplot
fig, ax = moran_scatterplot(moran_Y , aspect_equal=True)
plt.show()
```



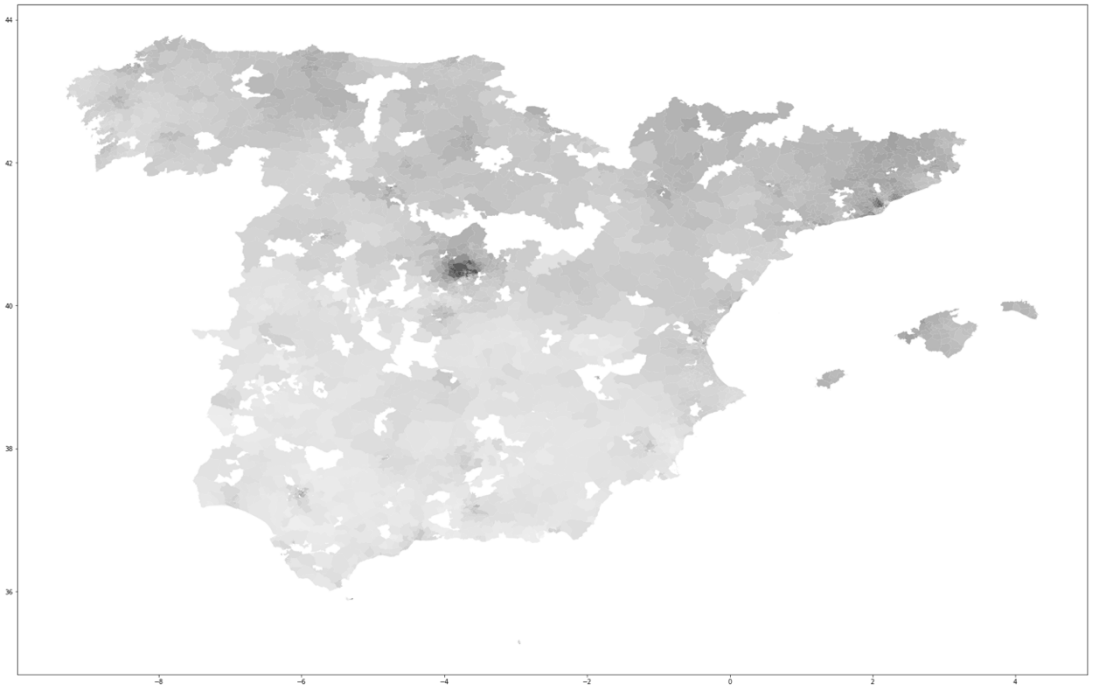
3.2.4.1 Map of features

The next step was to plot all the individual features -lagged and non-lagged- to see what actually was happening "under the hood" of these features. An excerpt of this is the plot of lagged income below. We can clearly see income in Spain is highest in Madrid and Barcelona. Furthermore, we see that in general the north is richer than the south.

```
income_lag1 = pd.DataFrame(income_lag)
income_lag1 = income_lag1.merge(df_tryout.geometry, how = 'left', l
eft_index = True, right_index = True)
income_lag1 = gpd.GeoDataFrame(income_lag1, crs="EPSG:4326", geomet
ry='geometry')
income_lag1.head()
```

	0	geometry
0	8049.013547	POLYGON ((-1.05858 38.34556, -1.05858 38.34556...
1	9151.277970	POLYGON ((-1.19809 38.28787, -1.20045 38.28562...
2	11020.721722	POLYGON ((-8.27051 43.28381, -8.27050 43.27841...
3	9129.130428	POLYGON ((-4.49641 38.99369, -4.48571 38.98113...
4	13079.713108	POLYGON ((1.93249 41.53670, 1.93238 41.53643, ...

Map of income – darker is higher income

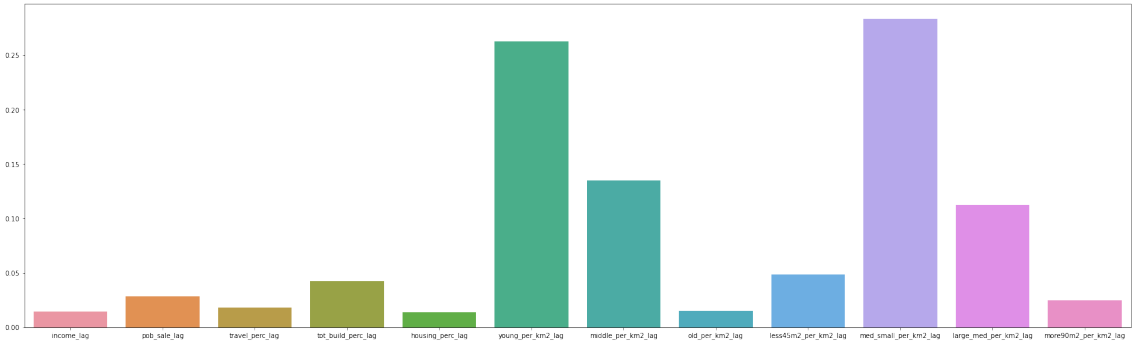


3.2.5 Random forest on lagged features

```
resall =pd.DataFrame()
res_w1 =pd.DataFrame()
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score

seed=7

kfold=KFold(n_splits=10, random_state=seed, shuffle = True)

num_trees=100
num_features=5
max_depth = 5
model=RandomForestRegressor(n_estimators=num_trees, max_features=nu
m_features,max_depth=max_depth, random_state=seed)
model.fit(lag_matrix,Y)

results_NO2_avg=cross_val_score(model, lag_matrix, Y, cv=kfold)

print(f'Random Forest - Accuracy {results_NO2_avg.mean()*100:.3f}%
std {results_NO2_avg.std()*100:3f}%')

res_w1["Res"]=results_NO2_avg
res_w1["Type"]="Random Forest"

resall=pd.concat([resall,res_w1], ignore_index=True)
```

```
Random Forest - Accuracy 84.102% std 2.821056%
```

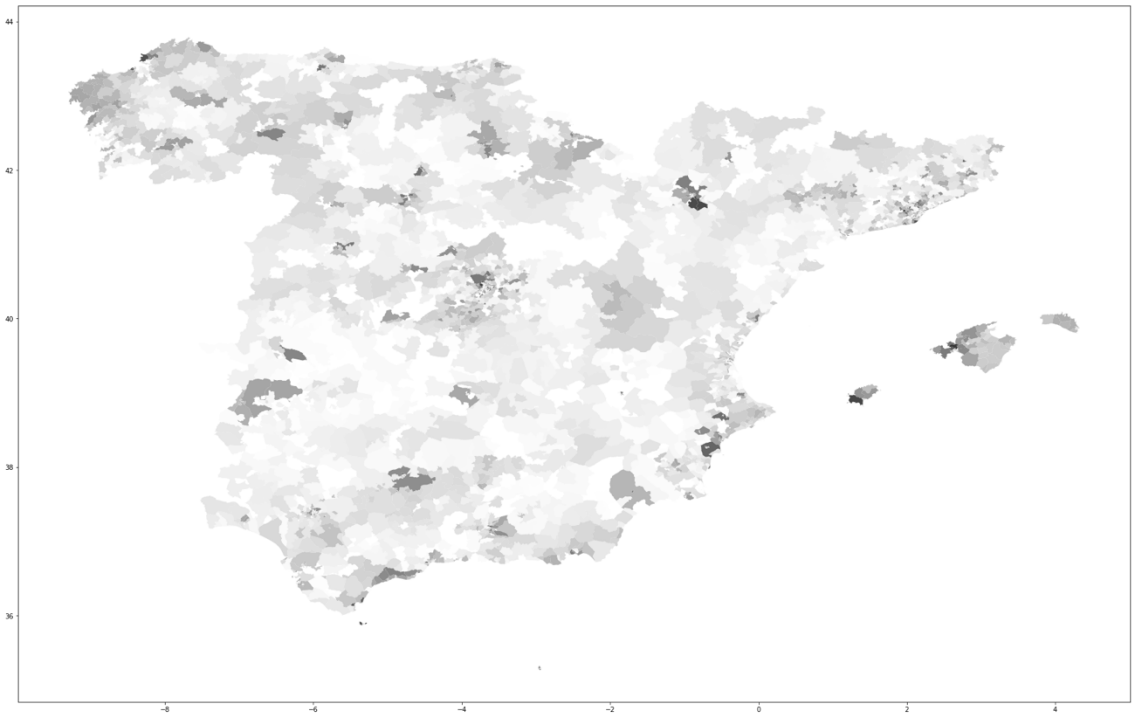
Feature importance – Higher is better



Our first try with lagged features actually performs worse than our non-lagged model. At this point it is hard to pinpoint the exact reason of this issue. However, it is likely that we have to set a bar in the Moran I upon which we select when to use the non-lagged and when to use the lagged feature. After different rounds of testing models, we found the cut off to be around 0.6. Additionally, we found that we use some features which are too highly correlated with each other. This implies we have to investigate the features individually and only select those most relevant. The result is the selection in the next model.

3.2.5.1 Map accuracy

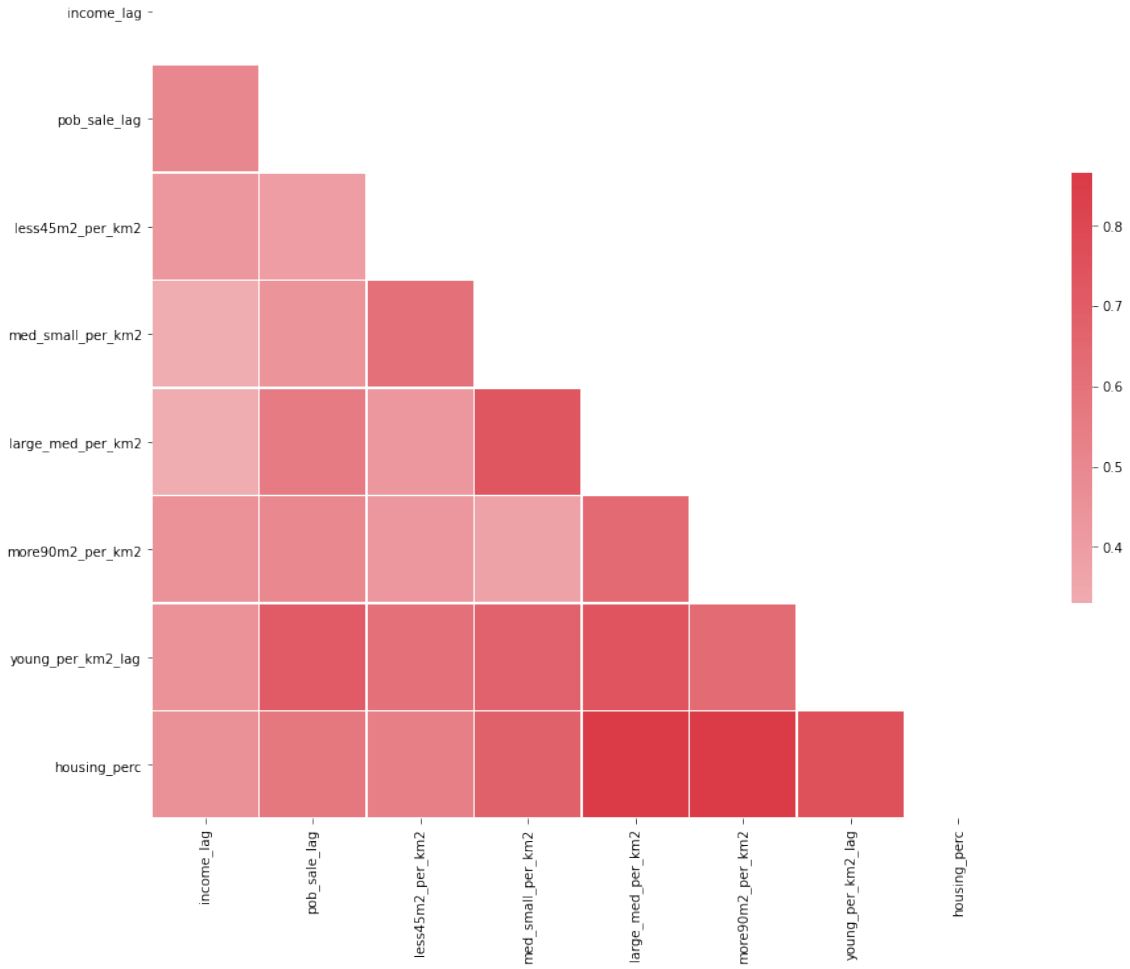
Map accuracy – darker is less accurate



3.2.6 Random forest on mixed features

```
mix_matrix = np.concatenate(( income_lag,  pob_sale_lag, less45m2_p
er_km2_lag, med_small_per_km2, large_med_per_km2,more90m2_per_km2,y
oung_per_km2_lag,housing_perc),axis=1)
names= ['income_lag','pob_sale_lag','less45m2_per_km2','med_small_p
er_km2','large_med_per_km2','more90m2_per_km2','young_per_km2_lag',
'housing_perc']

mix_matrix1= pd.DataFrame(mix_matrix)
mix_matrix1.columns = names
f, ax = plt.subplots(figsize=(16, 12))

mask = np.zeros_like(mix_matrix1.corr(), dtype=np.bool)
mask[np.triu_indices_from(mask)] = True
cmap = sns.diverging_palette(220, 10, as_cmap=True)
sns.heatmap(mix_matrix1.corr(), mask=mask, cmap=cmap, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
```

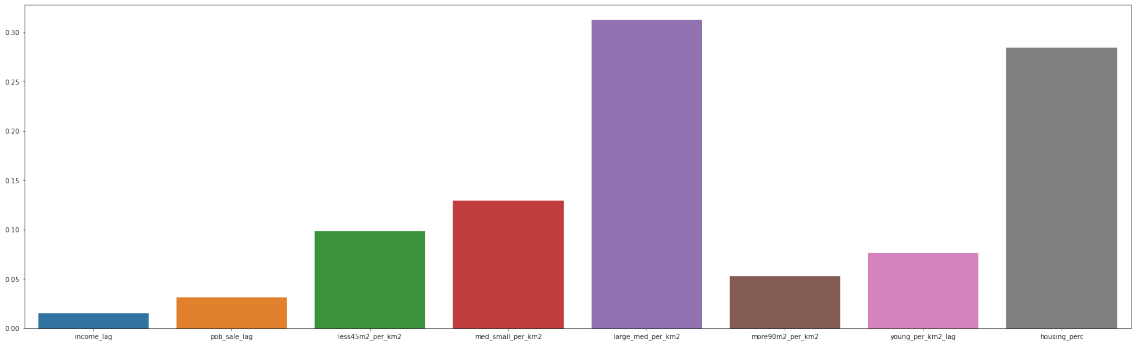
Correlation matrix – higher is more correlate



3.2.7 Random forest on mixed features performance

```
Random Forest - Accuracy 88.651% std 0.979779

Average NO2 level: 12.847696532150087

Expected error range: [11.856561640693192, 13.838831423606981]
```

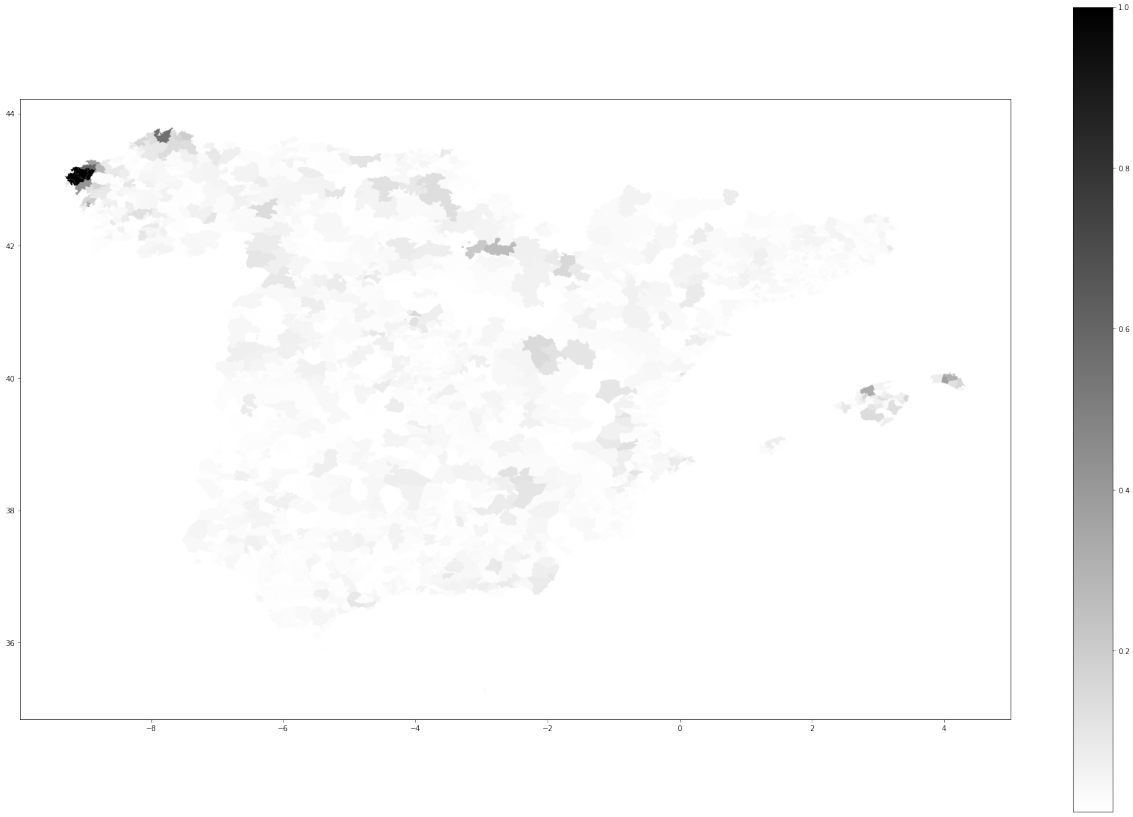
Feature importance – Higher is better



We get a great performance boost by selecting the most relevant features whilst also taking into account which features perform best in lagged and non-lagged form. Since we were not able to improve the random forest anymore after this point, we took this as the end point for our random forest models. To check if this really is the best we could do, the last step we took was comparing the performance of this model with that of XGBoost on the same set of features.

3.2.7.1 Map accuracy

Map accuracy – darker is less accurate

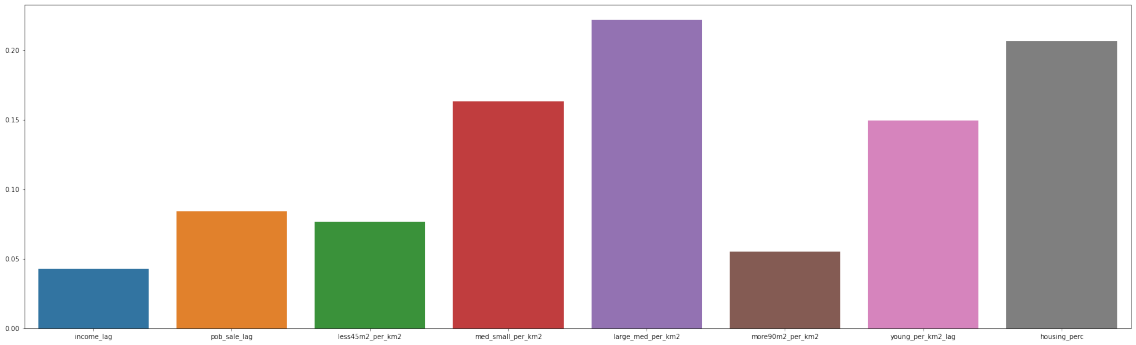


3.2.8 XGBoost

```
model12=xgb.XGBRegressor(colsample_bytree=0.4,
                 gamma=0,
                 learning_rate=0.07,
                 max_depth=3,
                 min_child_weight=1.5,
                 n_estimators=10000,
                 reg_alpha=0.75,
                 reg_lambda=0.45,
                 subsample=0.6,
                 seed=42)
model12.fit(mix_matrix,Y)
results_NO2_avg=cross_val_score(model12, mix_matrix, Y, cv=kfold)

print(f'Random Forest - Accuracy {results_NO2_avg.mean()*100:.3f}%
std {results_NO2_avg.std()*100:3f}%')
```

```
Random Forest - Accuracy 88.876% std 1.376834%
```

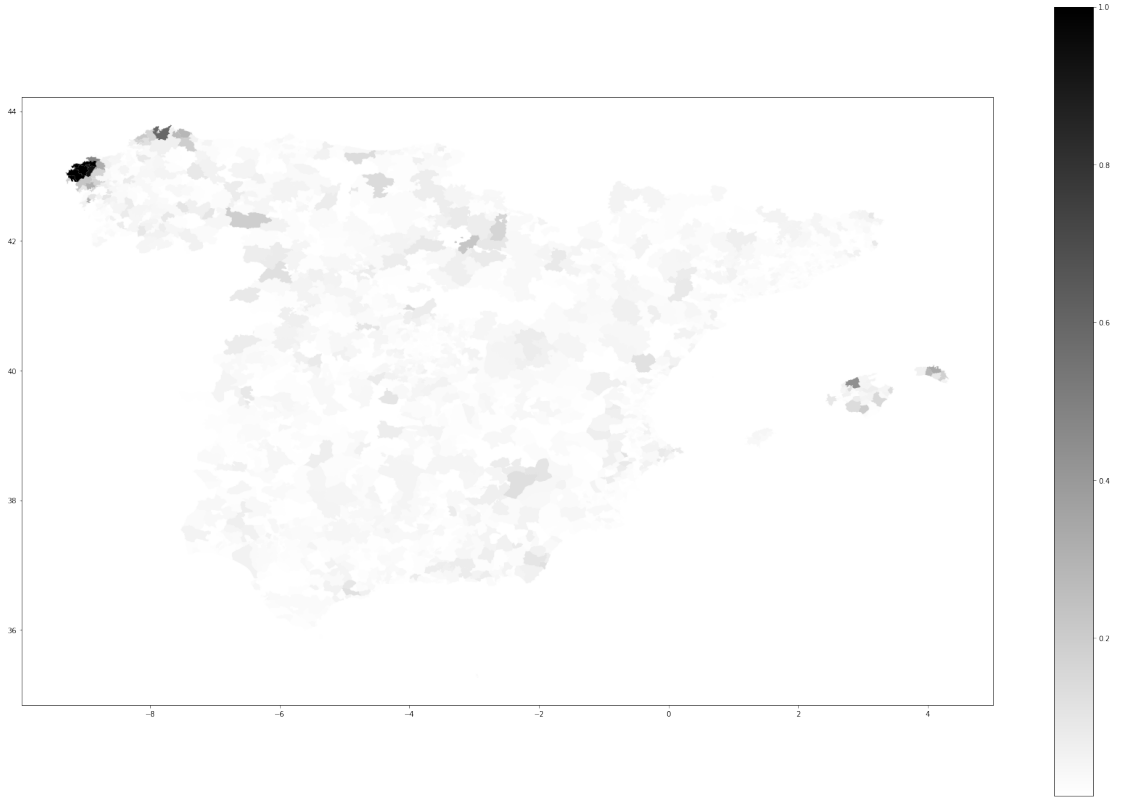
Feature importance – Higher is better



We see that only a very slight performance gain has been realized by using XGBoost instead of the random forest. The underlying use of the features did change quite a bit though. This made us belief that the XGBoost based model has more robust results. Consequently, we adopted the model above as our final prediction model to predict NO2 levels in Spain. As can be seen in the map below the model perform really well all throughout Spain. The only big outlier in the map is the Santiago the Compostela area.

3.2.8.1 Map accuracy

Map accuracy – darker is less accurate



4 Final remarks and conclusion

Our final deliverable was a model which used the best combination of 'normal' and spatial-lagged feature to predict NO2 level in Spain. We started our initial search to the best possible model with no more than a long list of potential data sources and a data set containing Spain's movement. After the first challenge to find NO2 data from all across Spain, the real challenge began. We had to select the variables which might be relevant in predicting NO2 levels. Since NO2 pollution is mostly caused by human travel, cars, or busses, we had some clue where to seek. We got relevant data from multiple sources, and although this whole process was quite lengthy in itself, it was only the start of our data science project. We had to make sure all the data was in a usable format and that the combined dataset fitted the geographical areas given by the movement data. To get everything in the correct geographical areas required a lot of data preparation.

After the initial preparation process, we ended up with a dataset containing more than 100 different features. We only had a vague idea of which ones were likely to become relevant at some point in the process. Luckily, we could rely on the expert opinion of 300.000km/s to guide us through the selection process.

We ended up with a list of 30 features to try, test, and adjust. After trying and adjusting to get the best possible model for our goal, we arrived at a model that uses eight features to predict NO2 throughout Spain. Based on the Moran's I score and multiple tryouts between different combinations of features being either spatially lagged or not. Our final model achieves an 88.8% accuracy of predicting NO2 levels throughout Spain, and we are proud of this result. We found that the percentage of space used for residential buildings and the number of homes with a surface between 61 and 90 m2 were the most potent predictors of NO2 levels. Other notable predictors were houses with a surface between 45 and 60 m2 and the number of people between 0 and 25 living per square kilometer. We could predict N02 level with the precision we were able to, mostly based on residential information, which shows how much good city planning can matter for cities' livability.

In our opinion, multiple sectors can leverage the results of our model. The first is the public sector, as urban planning matters concerning the question of pollution. By taking smart strategies to reduce traffic between places, cities might have more impact at a lower cost compared to routes taken right now. In the future, these conclusions might even get more support when we finish our GNN model. This model will give us information on what happens if we adjust particular traffic flows within the whole structure. An example can be building offices in Sant Cugat to reduce traffic inflow into Barcelona and improve Barcelona's air quality. This action contrasts with those taken nowadays, where politicians try to solve issues by taking measures where the pollution is too high. The second is the business sector, given the granularity of the predictions. ReaII

In our opinion, multiple sectors can leverage the results of our model. The first is the public sector, as urban planning matters concerning the question of pollution. By taking smart strategies to reduce traffic between places, cities might have more impact at a lower cost compared to routes taken right now. In the future, these conclusions might even get more support when we finish our GNN model. This model will give us information on what happens if we adjust particular traffic flows within the whole structure. An example can be building offices in Sant Cugat to reduce traffic inflow into Barcelona and improve Barcelona's air quality. This action contrasts with those taken nowadays, where politicians try to solve issues by taking measures where the pollution is too high.

The second is the business sector, given the granularity of the predictions. The real estate market might use our models to predict how air pollution will behave in the future. This is valuable information when making investment strategies or evaluating the current portfolio. More generally, businesses can use this information to plan their offices in the right locations to give them the possibility to market themselves as engaged businesses. This is not only what customers are demanding these days, but also of increasing interest to employees.

Thirdly, countries can use these models to check whether their more high-level pollution planning works according to their plan. In such a scenario, our predictions can be used to benchmark the areas in which particular pollution minimalizing measures have been taken to review their success. This will increase the time to market of successful ideas as it will take less time to confirm the results. Furthermore, it will enable a more rapid rollout of new ideas as a bad one will be killed sooner. All of this will save not only costs but with a bit of luck, and it might give an edge in saving the environment.

Finally, we want to thank Prof. Esteve Almirall for his outstanding support and mentorship and, of course, the excellent lunch at the Boqueria combined with a short city tour. All our conversations about data science in general, universities' role in society, and Spanish politics were gratifying. Furthermore, we had like to thank Pablo Martinez and André Resende for their availability and collaboration; we hope to see them in person one day and wish for great results to come with the GNN model. Lastly, we had like to thank all the professors at ESADE that taught us the best tools to do real Data Science.

Annexes

A. WAQI - World Air Quality Index

This notebook aimed to combine all the different datasets of different years – from 2015 to early 2020 – from the World Air Quality Index.

"The *World Air Quality Index* project is a non-profit project started in 2007. Its mission is to promote air pollution awareness for citizens and provide a unified and world-wide air quality information. The project provides transparent air quality information for more than 130 countries, covering more than 30,000 stations in 2000 major cities, via those two websites: aqicn.org and waqi.info." (WAQI, 2020)

At the beginning, we explored the WAQI-COVID CSV files; however, their content was not aligned with our goal. Later on, we had access to other data, the NO2 measurements, more useful for us. Yet, we decided to use another source for the NO2 measurements from the European Environmental Agency, which seemed more accurate. Nonetheless, we have added this part in our report as part of the process.

i. Libraries

We use the same libraries for the cleaning and exploring part of our project. Therefore, should not state otherwise, we show them only once.

```
import networkx as nx
import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import overpy
import shapely
import geojsonio as gjs
import json
import ipyleaflet as ipy
import osmnx as ox
from shapely.geometry import Point, LineString, Polygon
from descartes import PolygonPatch
from IPython.display import IFrame
ox.config(log_console=True, use_cache=True)
```

ii. Data

WAQI Data

The datasets information was commented in the first three lines of each file. We decided to combine all the files into one DataFrame, scraping Latitude and Longitude information and creating a column to populate each row with geographic coordinates.

```
'''
os.getcwd()

import os
Path = "../waqi/WAQI-1/"
filelist = os.listdir(Path)
location = []
WAQI_all = pd.DataFrame()
for i in filelist:
    if i.endswith(".csv"):
        with open(Path + i, 'r') as f:
            df = pd.read_csv(Path + i, comment = '#')
            length = len(df)
            for line in f:
                if line.startswith('#Station'):
                    second_row = line.split(' ')

                    latitude = []
                    lat = [float(second_row[-3][:-1])] * length
                    latitude.extend(lat)
                    df['Latitude'] = latitude

                    longitude = []
                    lon = [float(second_row[-1])] * length
                    longitude.extend(lon)
                    df['Longitude'] = longitude

                    WAQI_all = pd.concat([WAQI_all, df],
ignore_index=True)

WAQI_all =
WAQI_all[['date','no2','no2_min','no2_max','Latitude','Longitude']]
'''
```

	date	no2	no2_min	no2_max	geometry
0	2020-05-07	9	9	9	POINT (-5.95849 43.57532)
1	2020-05-06	7.5	2	17	POINT (-5.95849 43.57532)
2	2020-05-05	11.5	3	21	POINT (-5.95849 43.57532)
3	2020-05-04	12	3	29	POINT (-5.95849 43.57532)
4	2020-05-03	3.9	1	15	POINT (-5.95849 43.57532)

B. Spatial Lagged Features

```
######################
y_lag = ps.weights.lag_spatial(queen_w, Y)
income_lag = ps.weights.lag_spatial(queen_w, income)
pob_sale_lag = ps.weights.lag_spatial(queen_w, pob_sale)
travel_perc_lag = ps.weights.lag_spatial(queen_w, travel_perc)

tot_build_perc_lag = ps.weights.lag_spatial(queen_w,
tot_build_perc)
housing_perc_lag = ps.weights.lag_spatial(queen_w, housing_perc)

young_per_km2_lag = ps.weights.lag_spatial(queen_w, young_per_km2)
middle_per_km2_lag = ps.weights.lag_spatial(queen_w,
middle_per_km2)
old_per_km2_lag = ps.weights.lag_spatial(queen_w, old_per_km2)

less45m2_per_km2_lag = ps.weights.lag_spatial(queen_w,
less45m2_per_km2)
med_small_per_km2_lag = ps.weights.lag_spatial(queen_w,
med_small_per_km2)
large_med_per_km2_lag = ps.weights.lag_spatial(queen_w,
large_med_per_km2)
more90m2_per_km2_lag = ps.weights.lag_spatial(queen_w,
more90m2_per_km2)

lag_matrix = np.concatenate(( income_lag,
pob_sale_lag,travel_perc_lag, tot_build_perc_lag, housing_perc_lag,
young_per_km2_lag, middle_per_km2_lag, old_per_km2_lag,
less45m2_per_km2_lag, med_small_per_km2_lag,
large_med_per_km2_lag,more90m2_per_km2_lag),axis=1)
names=
['income_lag','pob_sale_lag','travel_perc_lag','tot_build_perc_lag'
,'housing_perc_lag','young_per_km2_lag','middle_per_km2_lag','old_p
er_km2_lag','less45m2_per_km2_lag','med_small_per_km2_lag','large_m
ed_per_km2_lag','more90m2_per_km2_lag']

lag_df = pd.DataFrame(data=lag_matrix, columns= names, index = None
)
lag_df.head()
middle_per_km2 = np.asarray(middle_per_km2).reshape(-
1,1).astype(float)
old_per_km2 = np.asarray(old_per_km2).reshape(-1,1).astype(float)

less45m2_per_km2 = np.asarray(less45m2_per_km2).reshape(-
1,1).astype(float)
med_small_per_km2 = np.asarray(med_small_per_km2).reshape(-
1,1).astype(float)
large_med_per_km2 = np.asarray(large_med_per_km2).reshape(-
1,1).astype(float)
more90m2_per_km2 = np.asarray(more90m2_per_km2).reshape(-
1,1).astype(float)
```

C. Graph Neural Network (Work in Progress)

i. Libraries

```
import pandas as pd
import geopandas as gpd
import numpy as np
import networkx as nx
import shapely as shp
```

ii. Network

We use networkx to create our network.

```
df_tryout = gpd.read_file('../Datasets/ultimate_move_df.geojson')
df_tryout['tot_build_perc'] = df_tryout.srf_tot/df_tryout.shape__ar
ea
df_tryout['housing_perc'] = df_tryout.srf_housin/df_tryout.shape__a
rea
df_tryout['shape__are'] = df_tryout.shape__area /1000000
df_tryout['pop_per_km2'] = df_tryout.tot_pop/df_tryout.shape__are
df_tryout['male_female_ratio'] = df_tryout.Male/df_tryout.Female
df_tryout['young_per_km2'] = df_tryout.sub_16_age / df_tryout.shape
__are
df_tryout['middle_per_km2'] = df_tryout['16_to_64_age'] / df_tryout
.shape__are
df_tryout['old_per_km2'] = df_tryout['64_more_age'] / df_tryout.sha
pe__are
df_tryout = df_tryout.drop(['shape_leng','shape_leng','NO2_avg_ut',
'sub_16_age','16_to_64_age','64_more_age','weight_urb','weight_tr',
'Male','Female','srf_housin','srf_tot'], axis = 1)
df_tryout['less45m2_per_km2'] = (df_tryout['less_30_m2'] + df_tryou
t['30_45_m2'])/df_tryout.shape__are
df_tryout['46m2_50m2_per_km2'] = df_tryout['46_60_m2']/df_tryout.sh
ape__are
df_tryout['61m2_90m2_per_km2'] = (df_tryout['61_75_m2'] + df_tryout
['76_90_m2'] )/df_tryout.shape__are
df_tryout['more90m2_per_km2'] = ( df_tryout['91_105_m2']+df_tryout[
'106_120_m2'] +  df_tryout['121_150_m2'] + df_tryout['150_180_m2']
+ df_tryout['above_180_m2'])/df_tryout.shape__area

df_tryout = df_tryout.drop(['less_30_m2' ,'30_45_m2','46_60_m2', '6
1_75_m2','76_90_m2','91_105_m2','106_120_m2','121_150_m2','150_180_
m2','O3_p932', 'O3_SOMO35','O3_S10',
                            'above_180_m2','pob_casa','pob_sale','f
irst_home', 'shape__are', 'pop_per_km2','tot_house','tot_pop','shap
e__area', 'shape__length','second_home','vacation_home'], axis = 1)
df_tryout.head()
```

	NO2_avg	Income	name_celda	geometry	tot_build_perc	housing_perc	male_female_ratio	young_per_km2	middle_per_km2
0	6.193526	9066.485226	Abanilla	POLYGON ((-1.05858 38.34556, -1.05858 38.34556...	0.007471	0.003512	1.025197	4.011302	16.957832
1	6.520018	8912.600076	Abarán	POLYGON ((-1.19809 38.28787, -1.20045 38.28562...	0.019438	0.011022	1.028571	18.924900	75.395060
2	4.628167	11013.135965	Abegondo	POLYGON ((-8.27051 43.28381, -8.27050 43.27841...	0.014269	0.009062	0.982578	7.777202	41.854405
3	4.927379	8783.054078	Abenójar y otros municipios	POLYGON ((-4.49641 38.99369, -4.48571 38.98113...	0.001459	0.000694	1.054645	0.444597	2.695122
4	16.765111	13371.511234	Abrera	POLYGON ((1.93249 41.53670, 1.93238 41.53643, ...	0.104217	0.041485	1.053090	116.400835	404.536522

Nodes

```
movements = gpd.read_file('../Datasets/movements.geojson')
nodes = movements.copy()
nodes = nodes[~nodes.nombre_celda_destino.isin(list(set(nodes['nombre_celda_destino']) - set(nodes['nombre_celda_origen'])))]
nodes_l = nodes.drop([ 'celda_destino', 'celda_origen', 'flujo', 'n_destino', 'nombre_celda_destino', 'p_pob_casa', 'pob_casa', 'pob_sale','p_pob_sale', 'pob_resid', 'geometry'], axis = 1)
nodes_l['geometry'] = movements.geometry.boundary.explode()[:,0]
nodes_l = nodes_l.drop_duplicates()
nodes_l.head()
```

	nombre_celda_origen	geometry
0	Montesinos, Los y Algorfa	POINT (-0.77723 38.04169)
8	Sant Joan de Vilatorrada y otros municipios	POINT (1.74088 41.76983)
17	Real de la Jara, El y otros municipios	POINT (-6.18708 37.77972)
19	Daimús y otros municipios	POINT (-0.14416 38.96415)
24	Formentera del Segura y otros municipios	POINT (-0.74723 38.09734)

```
df_merged = gpd.sjoin(df_tryout, nodes_l, how="inner", op='intersects')
df_merged = df_merged.drop(['name_celda', 'index_right', 'geometry'], axis = 1)
df_merged.head()
```

	NO2_avg	Income	tot_build_perc	housing_perc	male_female_ratio	young_per_km2	middle_per_km2	old_per_km2	less45m2_per_km2
0	6.193526	9066.485226	0.007471	0.003512	1.025197	4.011302	16.957832	6.345922	0.000000
1	6.520018	8912.600076	0.019438	0.011022	1.028571	18.924900	75.395060	19.577482	0.000000
2	4.628167	11013.135965	0.014269	0.009062	0.982578	7.777202	41.854405	18.047857	0.000000
3	4.927379	8783.054078	0.001459	0.000694	1.054645	0.444597	2.695122	1.306249	0.031476
4	16.765111	13371.511234	0.104217	0.041485	1.053090	116.400835	404.536522	67.547380	0.000000

Edges

```
edge = nodes[['nombre_celda_origen', 'nombre_celda_destino','flujo'
]]
edge.columns = ['source','target','weight']
edge.head()
```

	source	target	weight
0	Montesinos, Los y Algorfa	Formentera del Segura y otros municipios	107
1	Montesinos, Los y Algorfa	Almoradí	463
2	Montesinos, Los y Algorfa	Rojales	983
3	Montesinos, Los y Algorfa	San Miguel de Salinas	213
4	Montesinos, Los y Algorfa	Orihuela (distrito 05)	167

Graph

```
#Initialize the graph
G = nx.from_pandas_edgelist(edge, source='source', target= 'target'
, edge_attr= 'weight')

#See graph info
print('Graph Info:\n', nx.info(G))
```

```
Graph Info:
Type: Graph
Number of nodes: 3136
Number of edges: 16325
Average degree:  10.4114
```

```
node_attr = df_merged.set_index('nombre_celda_origen').to_dict('ind
ex')
nx.set_node_attributes(G, node_attr)

#Inspect the node features
print('\nGraph Edges weights: ', G['Rojales']['Montesinos, Los y Al
gorfa'])
print('\nGraph Nodes: ', G.nodes['Rojales'])
```

```
Graph Edges weights:  {'weight': 241}

Graph Nodes:  {'NO2_avg': 12.64, 'Income': 7342.00306065665, 'tot_buil
d_perc': 0.10225941019706217, 'housing_perc': 0.07261906943101344, 'ma
le_female_ratio': 1.0513698630136987, 'young_per_km2': 45.092589245828
22, 'middle_per_km2': 394.74198085763334, 'old_per_km2': 213.826149004
41123, 'less45m2_per_km2': 19.81892027336805, '46m2_50m2_per_km2': 13.
636871747730309, '61m2_90m2_per_km2': 97.45817675711261, 'more90m2_per
_km2': 0.0001596423119267628}
```

Bibliography

EEA. (2019). *Air pollution data - EEA*. Retrieved from European Environmental Agency: https://www.eea.europa.eu/themes/air/dc

INE. (2011). *Census Data 2011*. Retrieved from Instituto Nacional de Estadística: https://www.ine.es/censos2011_datos/cen11_datos_resultados_seccen.htm

Sharma, A. (2020). *Decision Tree vs. Random Forest – Which Algorithm Should you Use?* Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

Smith, J. (2016). *Spatial Analysis*. Retrieved from SEG wiki: https://wiki.seg.org/wiki/User:JudySmith/Spatial_analysis

Tyagi, P. (2018). *Decision Tree*. Retrieved from Medium: https://medium.com/@pytyagi/decision-tree-ac0c9e3b8258

WAQI. (2020, 09 30). *About the World Air Quality Index project*. Retrieved from World Air Quality Index: https://aqicn.org/contact/