Spark Project MSSD ENSAI Hugo BREHIER

January 6, 2020

1 Data Preparation

First, I initialize a connection to spark via the library pyspark

[1]:
```
import pyspark
```

[2]:
```
sc_conf = pyspark.SparkConf()
sc_conf.set('spark.executor.memory', '22g')
sc_conf.set('spark.driver.memory', '22g')
```

[2]:
```
<pyspark.conf.SparkConf at 0x7fe6f45a12e8>
```

[3]:
```
sc = pyspark.SparkContext(conf=sc_conf)
```

[4]:
```
sc.getConf().getAll()
```

[4]:
```
[('spark.app.id', 'local-1578306800830'),
 ('spark.executor.memory', '22g'),
 ('spark.rdd.compress', 'True'),
 ('spark.serializer.objectStreamReset', '100'),
 ('spark.driver.memory', '22g'),
 ('spark.master', 'local[*]'),
 ('spark.executor.id', 'driver'),
 ('spark.submit.deployMode', 'client'),
 ('spark.driver.port', '39665'),
 ('spark.ui.showConsoleProgress', 'true'),
 ('spark.app.name', 'pyspark-shell'),
 ('spark.driver.host', '192.168.0.25')]
```

[5]:
```
spark = pyspark.sql.SparkSession(sc)
```

Here files are loaded in Spark. They are supposed to be in a subfolder /data/

[6]:
```
prix2014_ddf = (spark.read.csv('./data/Prix2014.csv',sep=';
 ↪',header=None,inferSchema=True))
```

A small function to rename all columns

[7]:
```
def renameCols(df, old_columns, new_columns):
    for old_col,new_col in zip(old_columns,new_columns):
        df = df.withColumnRenamed(old_col,new_col)
    return df
```

[8]:
```
old_columns = prix2014_ddf.schema.names
new_columns = ('id_pdv','cp','pop',
               'latitude','longitude','date','id_carburant','nom_carburant',
               'prix(millieuros)'
              )


prix2014_ddf = renameCols(prix2014_ddf, old_columns, new_columns)
```

[9]:
```
prix2014_ddf.show(5)
```

```
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
| id_pdv|  cp|pop| latitude|longitude|
date|id_carburant|nom_carburant|prix(millieuros)|
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
|1000001|1000|  R|4620114.0| 519791.0|2014-01-02 11:08:03|           1|
Gazole|            1304|
|1000001|1000|  R|4620114.0| 519791.0|2014-01-04 09:54:03|           1|
Gazole|            1304|
|1000001|1000|  R|4620114.0| 519791.0|2014-01-05 10:27:09|           1|
Gazole|            1304|
|1000001|1000|  R|4620114.0| 519791.0|2014-01-06 09:07:51|           1|
Gazole|            1304|
|1000001|1000|  R|4620114.0| 519791.0|2014-01-07 09:23:56|           1|
Gazole|            1304|
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
only showing top 5 rows
```

[10]:
```
prix2015_ddf = (spark.read.csv('./data/Prix2015.csv',sep=';
 ↪',header=None,inferSchema=True))
old_columns = prix2015_ddf.schema.names
new_columns = ('id_pdv','cp','pop',
               'latitude','longitude','date','id_carburant','nom_carburant',
               'prix(millieuros)'
              )


prix2015_ddf = renameCols(prix2015_ddf, old_columns, new_columns)
prix2015_ddf.show(5)
```

```
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
| id_pdv|  cp|pop| latitude|longitude|
```

```
date|id_carburant|nom_carburant|prix(millieuros)|
+-------+----+---+---------+---------+------------------+-----------+---------
----+----------------+
|1000001|1000|  R|4620114.0| 519791.0|2015-01-02 11:01:45|          1|
Gazole|            1141|
|1000001|1000|  R|4620114.0| 519791.0|2015-01-03 09:01:42|          1|
Gazole|            1141|
|1000001|1000|  R|4620114.0| 519791.0|2015-01-07 10:01:44|          1|
Gazole|            1141|
|1000001|1000|  R|4620114.0| 519791.0|2015-01-08 10:01:06|          1|
Gazole|            1115|
|1000001|1000|  R|4620114.0| 519791.0|2015-01-09 10:01:19|          1|
Gazole|            1115|
+-------+----+---+---------+---------+------------------+-----------+---------
----+----------------+
only showing top 5 rows
```

[11]:
```
prix2016_ddf = (spark.read.csv('./data/Prix2016.csv',sep=';
 ↪',header=None,inferSchema=True))
old_columns = prix2016_ddf.schema.names
new_columns = ('id_pdv','cp','pop',
               'latitude','longitude','date','id_carburant','nom_carburant',
               'prix(millieuros)'
              )


prix2016_ddf = renameCols(prix2016_ddf, old_columns, new_columns)
prix2016_ddf.show(5)
```

```
+-------+----+---+---------+---------+------------------+-----------+---------
----+----------------+
| id_pdv|  cp|pop| latitude|longitude|
date|id_carburant|nom_carburant|prix(millieuros)|
+-------+----+---+---------+---------+------------------+-----------+---------
----+----------------+
|1000001|1000|  R|4620114.0| 519791.0|2016-01-02 09:01:58|          1|
Gazole|            1026|
|1000001|1000|  R|4620114.0| 519791.0|2016-01-04 10:01:35|          1|
Gazole|            1026|
|1000001|1000|  R|4620114.0| 519791.0|2016-01-04 12:01:15|          1|
Gazole|            1026|
|1000001|1000|  R|4620114.0| 519791.0|2016-01-05 09:01:12|          1|
Gazole|            1026|
|1000001|1000|  R|4620114.0| 519791.0|2016-01-07 08:01:13|          1|
Gazole|            1026|
+-------+----+---+---------+---------+------------------+-----------+---------
```

```
----+----------------+
only showing top 5 rows
```

[12]:
```
prix2017_ddf = (spark.read.csv('./data/Prix2017.csv',sep=';
 ↪',header=None,inferSchema=True))
old_columns = prix2017_ddf.schema.names
new_columns = ('id_pdv','cp','pop',
               'latitude','longitude','date','id_carburant','nom_carburant',
               'prix(millieuros)'
              )


prix2017_ddf = renameCols(prix2017_ddf, old_columns, new_columns)
prix2017_ddf.show(5)
```

```
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
| id_pdv|  cp|pop| latitude|longitude|
date|id_carburant|nom_carburant|prix(millieuros)|
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
|1000001|1000|  R|4620114.0| 519791.0|2017-01-02 09:37:03|           1|
Gazole|            1258|
|1000001|1000|  R|4620114.0| 519791.0|2017-01-03 09:54:58|           1|
Gazole|            1258|
|1000001|1000|  R|4620114.0| 519791.0|2017-01-06 12:33:57|           1|
Gazole|            1258|
|1000001|1000|  R|4620114.0| 519791.0|2017-01-09 08:59:53|           1|
Gazole|            1258|
|1000001|1000|  R|4620114.0| 519791.0|2017-01-10 10:38:39|           1|
Gazole|            1258|
+-------+----+---+---------+---------+------------------+------------+---------
----+----------------+
only showing top 5 rows
```

The files are merged in one dataframe

[13]:
```
import functools

def unionAll(dfs):
    return functools.reduce(lambda df1,df2: df1.union(df2.select(df1.columns)),␣
 ↪dfs)
```

[14]:
```
prixall_ddf = unionAll([prix2014_ddf,prix2015_ddf,prix2016_ddf,prix2017_ddf])
```

[15]:
```
prixall_ddf.count()
```

[15]:
```
13063734
```

Some features are created based on the date. Week_index and day are not yearly-periodic contrary to month and week

[16]:
```
from pyspark.sql.functions import year,month,weekofyear,hour,dayofyear

prixall_ddf = prixall_ddf.withColumn("year", year("date"))
prixall_ddf = prixall_ddf.withColumn("month", month("date"))
prixall_ddf = prixall_ddf.withColumn("week", weekofyear("date"))
prixall_ddf = prixall_ddf.withColumn("week_index", weekofyear("date") +␣
 ↪52*(prixall_ddf.year-2014))
prixall_ddf = prixall_ddf.withColumn("day", dayofyear("date")+ 365* (prixall_ddf.
 ↪year-2014))
#prixall_ddf = prixall_ddf.withColumn("hour", hour("date"))
prixall_ddf = prixall_ddf.drop('date')
prixall_ddf.show(5)
```

```
+-------+----+---+---------+---------+------------+-------------+---------------
-+----+-----+----+----------+---+
| id_pdv|  cp|pop| latitude|longitude|id_carburant|nom_carburant|prix(millieuros
)|year|month|week|week_index|day|
+-------+----+---+---------+---------+------------+-------------+---------------
-+----+-----+----+----------+---+
|1000001|1000|  R|4620114.0| 519791.0|           1|       Gazole|
1304|2014|    1|   1|         1|  2|
|1000001|1000|  R|4620114.0| 519791.0|           1|       Gazole|
1304|2014|    1|   1|         1|  4|
|1000001|1000|  R|4620114.0| 519791.0|           1|       Gazole|
1304|2014|    1|   1|         1|  5|
|1000001|1000|  R|4620114.0| 519791.0|           1|       Gazole|
1304|2014|    1|   2|         2|  6|
|1000001|1000|  R|4620114.0| 519791.0|           1|       Gazole|
1304|2014|    1|   2|         2|  7|
+-------+----+---+---------+---------+------------+-------------+---------------
-+----+-----+----+----------+---+
only showing top 5 rows
```

From https://www.prix-carburants.gouv.fr/rubrique/opendata/ Question 4 :

[17]:
```
prixall_ddf = prixall_ddf.withColumn("latitude",prixall_ddf.latitude / 100000 )
prixall_ddf = prixall_ddf.withColumn("longitude",prixall_ddf.longitude / 100000 )
prixall_ddf.show(5)
```

```
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+
| id_pdv|  cp|pop|latitude|longitude|id_carburant|nom_carburant|prix(millieuros)
|year|month|week|week_index|day|
```

```
+-------+----+---+--------+---------+-----------+-------------+---------------
+----+-----+----+----------+---+
|1000001|1000|  R|46.20114|  5.19791|          1|       Gazole|
1304|2014|    1|   1|         1|  2|
|1000001|1000|  R|46.20114|  5.19791|          1|       Gazole|
1304|2014|    1|   1|         1|  4|
|1000001|1000|  R|46.20114|  5.19791|          1|       Gazole|
1304|2014|    1|   1|         1|  5|
|1000001|1000|  R|46.20114|  5.19791|          1|       Gazole|
1304|2014|    1|   2|         2|  6|
|1000001|1000|  R|46.20114|  5.19791|          1|       Gazole|
1304|2014|    1|   2|         2|  7|
+-------+----+---+--------+---------+-----------+-------------+---------------
+----+-----+----+----------+---+
only showing top 5 rows
```

Here we can register the dataframe to execute some raw SQL queries

[18]:
```
prixall_ddf.registerTempTable('prixall_table')
```

[19]:
```
sqlContext = pyspark.SQLContext(sc)
```

[20]:
```
sqlContext.sql('select distinct(week) from prixall_table order by week DESC').
 ↪show(5)
```

```
+----+
|week|
+----+
|  53|
|  52|
|  51|
|  50|
|  49|
+----+
only showing top 5 rows
```

[21]:
```
sqlContext.sql('select distinct(week_index) from prixall_table order by␣
 ↪week_index DESC').show(5)
```

```
+----------+
|week_index|
+----------+
|       208|
|       207|
|       206|
|       205|
|       204|
```

```
+----------+
only showing top 5 rows
```

Next, we compute some average in order to get the price_index

[22]:
```
import pyspark.sql.functions as F
```

[23]:
```
prixall_ddf_tmp = (prixall_ddf
                .groupby('week','id_carburant')
                .agg(F.avg('prix(millieuros)').alias('prix_moyen_fr'))

)
```

[24]:
```
prixall_ddf_tmp = (prixall_ddf_tmp
                    .withColumnRenamed("week", "week2")
                    .withColumnRenamed("id_carburant", "id_carburant2")
                   )
```

[25]:
```
prixall_ddf_tmp.filter(F.col('week2')<5).show()
```

```
+-----+-------------+------------------+
|week2|id_carburant2|     prix_moyen_fr|
+-----+-------------+------------------+
|    3|            1|1152.7277113373882|
|    2|            2|1365.2639077779936|
|    2|            3| 741.7595036438842|
|    1|            2|1361.8992397298268|
|    1|            1|1181.0911314644688|
|    1|            3| 732.1840094062317|
|    3|            5| 1321.444525103035|
|    2|            5|1339.5510921961177|
|    1|            4| 753.2087378640776|
|    3|            3| 750.1097638800255|
|    4|            3| 737.0813096270598|
|    2|            1|  1176.55605922038|
|    2|            4| 762.2450042617672|
|    4|            6|1407.9012798198014|
|    3|            6|1403.3414944054618|
|    3|            2|1352.4703459098935|
|    1|            5| 1340.448898249428|
|    4|            2|1356.4011389604266|
|    3|            4|  762.974478680361|
|    4|            4|  756.958502240239|
+-----+-------------+------------------+
only showing top 20 rows
```

[26]:
```
prixall_ddf = (prixall_ddf.join(prixall_ddf_tmp,
                                (prixall_ddf.week == prixall_ddf_tmp.week2)
```

```
                                    & (prixall_ddf.id_carburant ==␣
↪prixall_ddf_tmp.id_carburant2))
                    .drop('week2','id_carburant2')
                    .
↪orderBy(['id_pdv','week_index','id_carburant'],ascending=True)
                  )
```

[27]:
```
prixall_ddf.count()
```

[27]:
```
13054278
```

[28]:
```
prixall_ddf.show(5)
```

```
+-------+----+---+--------+---------+------------+-------------+---------------
+----+-----+----+----------+---+------------------+
| id_pdv|  cp|pop|latitude|longitude|id_carburant|nom_carburant|prix(millieuros)
|year|month|week|week_index|day|     prix_moyen_fr|
+-------+----+---+--------+---------+------------+-------------+---------------
+----+-----+----+----------+---+------------------+
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|         1|364|1181.0911314644688|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|         1|365|1181.0911314644688|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  4|1181.0911314644688|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  2|1181.0911314644688|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  5|1181.0911314644688|
+-------+----+---+--------+---------+------------+-------------+---------------
+----+-----+----+----------+---+------------------+
only showing top 5 rows
```

[29]:
```
prixall_ddf_tmp = (prixall_ddf
                .groupby('week','id_carburant','id_pdv')
                .agg(F.avg('prix(millieuros)').alias('prix_moyen_station'))

)

prixall_ddf_tmp = (prixall_ddf_tmp
                    .withColumnRenamed("week", "week2")
                    .withColumnRenamed("id_carburant", "id_carburant2")
                    .withColumnRenamed("id_pdv", "id_pdv2")
                   )

prixall_ddf = (prixall_ddf.join(prixall_ddf_tmp,
                                    (prixall_ddf.week == prixall_ddf_tmp.week2)
```

```
                                    & (prixall_ddf.id_carburant ==␣
↪prixall_ddf_tmp.id_carburant2)
                                    & (prixall_ddf.id_pdv == prixall_ddf_tmp.
↪id_pdv2))
                    .drop('week2','id_carburant2','id_pdv2')
                    .
↪orderBy(['id_pdv','week_index','id_carburant'],ascending=True)
                  )
```

Finally,we get our price index

[30]:
```
prixall_ddf = (prixall_ddf.withColumn('price_index',
                                      100*(((prixall_ddf.prix_moyen_station -␣
↪prixall_ddf.prix_moyen_fr) / prixall_ddf.prix_moyen_fr)+1)
                                     )
              )
```

[31]:
```
prixall_ddf.show(5)
```

```
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+-----------------+-----------------+----------
-------+
| id_pdv|  cp|pop|latitude|longitude|id_carburant|nom_carburant|prix(millieuros)
|year|month|week|week_index|day|     prix_moyen_fr|prix_moyen_station|
price_index|
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+-----------------+-----------------+----------
-------+
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|
1|364|1181.0911314644688|1129.3333333333333|95.61779808921607|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
5|1181.0911314644688|1129.3333333333333|95.61779808921607|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
4|1181.0911314644688|1129.3333333333333|95.61779808921607|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
2|1181.0911314644688|1129.3333333333333|95.61779808921607|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|
1|365|1181.0911314644688|1129.3333333333333|95.61779808921607|
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+-----------------+-----------------+----------
-------+
only showing top 5 rows
```

2 Data Visualisation

[32]:
```
prixall_ddf_tmp = (prixall_ddf
                .groupby('week_index','id_carburant')
                .agg(F.avg('prix(millieuros)').alias('prix_moyen_fr_weekindex'))

                 )

prixall_ddf_tmp = (prixall_ddf_tmp
                    .withColumnRenamed("week_index", "week_index2")
                    .withColumnRenamed("id_carburant", "id_carburant2")
                   )

prixall_ddf = (prixall_ddf.join(prixall_ddf_tmp,
                                    (prixall_ddf.week_index == prixall_ddf_tmp.
 ↪week_index2)
                                    & (prixall_ddf.id_carburant ==␣
 ↪prixall_ddf_tmp.id_carburant2))
                     .drop('week_index2','id_carburant2')
                     .
 ↪orderBy(['id_pdv','week_index','id_carburant'],ascending=True)
                   )
```

[33]:
```
prixall_ddf.show(5)
```

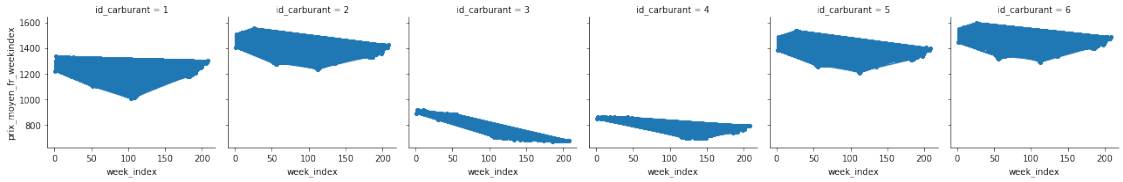
```
+-------+----+---+--------+---------+------------+-------------+---------------
+----+-----+----+----------+---+-----------------+-----------------+----------
-------+-----------------------+
| id_pdv|  cp|pop|latitude|longitude|id_carburant|nom_carburant|prix(millieuros)
|year|month|week|week_index|day|     prix_moyen_fr|prix_moyen_station|
price_index|prix_moyen_fr_weekindex|
+-------+----+---+--------+---------+------------+-------------+---------------
+----+-----+----+----------+---+-----------------+-----------------+----------
-------+-----------------------+
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|
1|364|1181.0911314644688|1129.3333333333333|95.61779808921607|
1221.7955122777307|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|
1|365|1181.0911314644688|1129.3333333333333|95.61779808921607|
1221.7955122777307|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
4|1181.0911314644688|1129.3333333333333|95.61779808921607|
1221.7955122777307|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
```

```
2|1181.0911314644688|1129.3333333333333|95.61779808921607|
1221.7955122777307|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|
5|1181.0911314644688|1129.3333333333333|95.61779808921607|
1221.7955122777307|
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+------------------+-----------------+----------
-------+-----------------------+
only showing top 5 rows
```

I use seaborn to plot the prix_moyen_fr_weekindex just computed.

[34]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

[35]:
```
prixall_pd = prixall_ddf.toPandas()
g = sns.FacetGrid(prixall_pd,col='id_carburant')
g = g.map(plt.plot, "week_index", "prix_moyen_fr_weekindex", marker=".")
```



It seems that prices vary quite frequently which creates lineplots ressembling polygons... Otherwise, we can see 2 carburants that are quite lower than the other types.

Next, I will try to get some work done on geodata. From https://www.data.gouv.fr/fr/datasets/contours-des-departements-francais-issus-d-openstreetmap/, I retrieved delimitations of french departements. I will use geopandas and the underlying shapely packages.

[36]:
```
#!pip install geopandas
#!pip install descartes
import geopandas as gpd
import descartes
```

The department data is in the form of a shapefile (.shp)

[37]:
```
shapefile = gpd.read_file("./departements-20180101-shp/departements-20180101.
 ↪shp")
shapefile
```

[37]:

	code_insee	nom	nuts3	wikipedia	\
0	974	La Réunion	FR940	fr:La Réunion	
1	11	Aude	FR811	fr:Aude (département)	
2	43	Haute-Loire	FR723	fr:Haute-Loire	

3	13	Bouches-du-Rhône	FR823	fr:Bouches-du-Rhône
4	47	Lot-et-Garonne	FR614	fr:Lot-et-Garonne
5	23	Creuse	FR632	fr:Creuse (département)
6	19	Corrèze	FR631	fr:Corrèze (département)
7	15	Cantal	FR722	fr:Cantal (département)
8	91	Essonne	FR104	fr:Essonne (département)
9	76	Seine-Maritime	FR232	fr:Seine-Maritime
10	38	Isère	FR714	fr:Isère (département)
11	2A	Corse-du-Sud	FR831	fr:Corse-du-Sud
12	2B	Haute-Corse	FR832	fr:Haute-Corse
13	63	Puy-de-Dôme	FR724	fr:Puy-de-Dôme
14	81	Tarn	FR627	fr:Tarn (département)
15	74	Haute-Savoie	FR716	fr:Haute-Savoie
16	73	Savoie	FR718	fr:Savoie (département)
17	06	Alpes-Maritimes	FR823	fr:Alpes-Maritimes
18	34	Hérault	FR813	fr:Hérault (département)
19	62	Pas-de-Calais	FR302	fr:Pas-de-Calais
20	80	Somme	FR223	fr:Somme (département)
21	972	Martinique	FR920	fr:Martinique
22	65	Hautes-Pyrénées	FR626	fr:Hautes-Pyrénées
23	12	Aveyron	FR622	fr:Aveyron (département)
24	40	Landes	FR613	fr:Landes (département)
25	64	Pyrénées-Atlantiques	FR615	fr:Pyrénées-Atlantiques
26	66	Pyrénées-Orientales	FR815	fr:Pyrénées-Orientales
27	87	Haute-Vienne	FR633	fr:Haute-Vienne
28	33	Gironde	FR612	fr:Gironde (département)
29	36	Indre	FR243	fr:Indre (département)
..	...	...	...	...
72	50	Manche	FR252	fr:Manche (département)
73	78	Yvelines	FR103	fr:Yvelines
74	27	Eure	FR231	fr:Eure (département)
75	28	Eure-et-Loir	FR242	fr:Eure-et-Loir
76	29	Finistère	FR522	fr:Finistère
77	70	Haute-Saône	FR433	fr:Haute-Saône
78	21	Côte-d'Or	FR261	fr:Côte-d'Or
79	973	Guyane	FR930	fr:Guyane
80	53	Mayenne	FR513	fr:Mayenne (département)
81	49	Maine-et-Loire	FR512	fr:Maine-et-Loire
82	41	Loir-et-Cher	FR245	fr:Loir-et-Cher
83	16	Charente	FR531	fr:Charente (département)
84	37	Indre-et-Loire	FR244	fr:Indre-et-Loire
85	17	Charente-Maritime	FR532	fr:Charente-Maritime
86	90	Territoire-de-Belfort	FR434	fr:Territoire de Belfort
87	26	Drôme	FR713	fr:Drôme (département)
88	05	Hautes-Alpes	FR822	fr:Hautes-Alpes
89	55	Meuse	FR412	fr:Meuse (département)
90	10	Aube	FR212	fr:Aube (département)

91	52	Haute-Marne	FR214	fr:Haute-Marne	
92	88	Vosges	FR414	fr:Vosges (département)	
93	976	Mayotte	None	fr:Mayotte	
94	84	Vaucluse	FR826	fr:Vaucluse (département)	
95	48	Lozère	FR814	fr:Lozère (département)	
96	04	Alpes-de-Haute-Provence	FR821	fr:Alpes-de-Haute-Provence	
97	56	Morbihan	FR524	fr:Morbihan	
98	25	Doubs	FR431	fr:Doubs (département)	
99	39	Jura	FR432	fr:Jura (département)	
100	07	Ardèche	FR712	fr:Ardèche (département)	
101	30	Gard	FR812	fr:Gard	

	surf_km2	geometry
0	2505.0	MULTIPOLYGON (((55.21643 -21.03904, 55.21652 -...
1	6343.0	POLYGON ((1.68872 43.27368, 1.69001 43.27423, ...
2	5003.0	POLYGON ((3.08206 45.28988, 3.08209 45.29031, ...
3	5247.0	MULTIPOLYGON (((4.23014 43.46047, 4.23025 43.4...
4	5385.0	POLYGON ((-0.14058 44.22648, -0.12931 44.23218...
5	5599.0	POLYGON ((1.37254 46.21672, 1.37257 46.21677, ...
6	5898.0	POLYGON ((1.22696 45.27178, 1.22705 45.27180, ...
7	5774.0	POLYGON ((2.06290 44.97664, 2.06355 44.97666, ...
8	1819.0	POLYGON ((1.91446 48.46186, 1.91557 48.46495, ...
9	6329.0	POLYGON ((0.06577 49.51269, 0.06591 49.51310, ...
10	7878.0	POLYGON ((4.74157 45.41589, 4.74158 45.41627, ...
11	4017.0	MULTIPOLYGON (((8.53996 42.23689, 8.54030 42.2...
12	4704.0	MULTIPOLYGON (((8.57438 42.38217, 8.57508 42.3...
13	8015.0	POLYGON ((2.38802 45.82583, 2.38802 45.82587, ...
14	5782.0	POLYGON ((1.53530 43.95960, 1.53551 43.95967, ...
15	4840.0	POLYGON ((5.80513 46.01485, 5.80518 46.01571, ...
16	6269.0	POLYGON ((5.62183 45.61205, 5.62200 45.61252, ...
17	4294.0	MULTIPOLYGON (((6.63330 43.78508, 6.63371 43.7...
18	6232.0	MULTIPOLYGON (((2.53955 43.34589, 2.53981 43.3...
19	6692.0	MULTIPOLYGON (((1.55562 50.40027, 1.55575 50.4...
20	6191.0	MULTIPOLYGON (((1.37983 50.06518, 1.38000 50.0...
21	1089.0	MULTIPOLYGON (((-61.22908 14.82247, -61.22895 ...
22	4519.0	MULTIPOLYGON (((-0.32765 42.91743, -0.32529 42...
23	8773.0	POLYGON ((1.83966 44.47586, 1.83969 44.47613, ...
24	9355.0	POLYGON ((-1.53338 43.53150, -1.53332 43.53158...
25	7683.0	MULTIPOLYGON (((-1.79102 43.37292, -1.79048 43...
26	4137.0	POLYGON ((1.72253 42.51769, 1.72332 42.51818, ...
27	5557.0	POLYGON ((0.62934 45.71480, 0.62991 45.71511, ...
28	10086.0	MULTIPOLYGON (((-1.26138 44.64239, -1.26122 44...
29	6898.0	POLYGON ((0.86711 46.74873, 0.87618 46.75100, ...
..	...	...
72	6040.0	MULTIPOLYGON (((-1.94877 49.71649, -1.94836 49...
73	2306.0	POLYGON ((1.44624 49.04639, 1.44945 49.04765, ...
74	6031.0	POLYGON ((0.29616 49.42987, 0.30281 49.43038, ...

```
75     5931.0  POLYGON ((0.75565 48.29990, 0.75585 48.30036, ...
76     6796.0  MULTIPOLYGON (((-4.79551 48.41438, -4.79551 48...
77     5390.0  POLYGON ((5.36695 47.46497, 5.36695 47.46510, ...
78     8802.0  MULTIPOLYGON (((4.06540 47.40725, 4.06759 47.4...
79    83543.0  MULTIPOLYGON (((-54.60278 2.33370, -54.60268 2...
80     5213.0  POLYGON ((-1.23885 47.80950, -1.23877 47.80962...
81     7172.0  MULTIPOLYGON (((-1.35422 47.30435, -1.35382 47...
82     6422.0  POLYGON ((0.58055 47.71290, 0.58239 47.71464, ...
83     5973.0  POLYGON ((-0.46314 45.75191, -0.46299 45.75206...
84     6157.0  POLYGON ((0.05272 47.19656, 0.05321 47.19721, ...
85     6906.0  MULTIPOLYGON (((-1.24366 45.77490, -1.24349 45...
86      611.0  POLYGON ((6.75646 47.72497, 6.75659 47.72507, ...
87     6558.0  POLYGON ((4.64691 44.35062, 4.64693 44.35128, ...
88     5697.0  POLYGON ((5.41839 44.42476, 5.41852 44.42493, ...
89     6235.0  POLYGON ((4.88846 48.80060, 4.88854 48.80062, ...
90     6027.0  POLYGON ((3.38364 48.47958, 3.38370 48.47963, ...
91     6256.0  POLYGON ((4.62751 48.46717, 4.62755 48.46725, ...
92     5897.0  POLYGON ((5.39361 48.39177, 5.39373 48.39181, ...
93      366.0  MULTIPOLYGON (((45.03981 -12.72228, 45.03981 -...
94     3577.0  MULTIPOLYGON (((4.64857 44.26372, 4.64867 44.2...
95     5175.0  POLYGON ((2.98226 44.64515, 2.98304 44.64546, ...
96     6993.0  POLYGON ((5.49642 44.10309, 5.49726 44.10367, ...
97     6870.0  MULTIPOLYGON (((-3.73508 48.11140, -3.73507 48...
98     5256.0  POLYGON ((5.69876 47.26464, 5.69877 47.26481, ...
99     5049.0  MULTIPOLYGON (((5.25202 46.94451, 5.25208 46.9...
100    5566.0  POLYGON ((3.86110 44.71118, 3.86110 44.71151, ...
101    5875.0  POLYGON ((3.26190 44.09335, 3.26221 44.09389, ...

[102 rows x 6 columns]
```

I change some departments with non-numeric codes, such as Corse-du-Sud with code_insee 2A.

[38]:
```
codes = shapefile.code_insee
codes.loc[11,2] = 2.1
codes.loc[12,2] = 2.2
codes.loc[63,2] = 69.1
codes.loc[64,2] = 69.2
shapefile.code_insee = codes
```

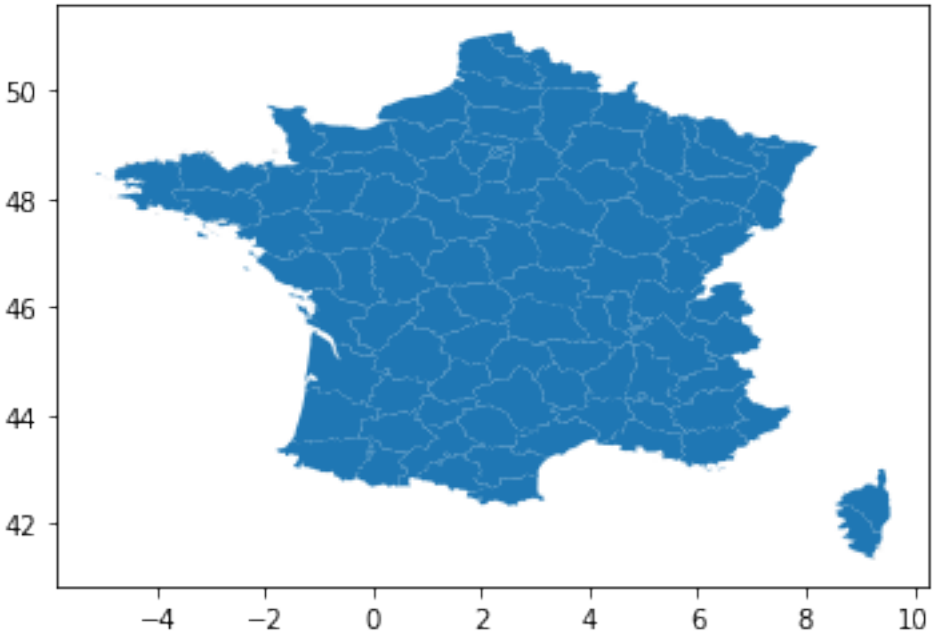
```
/home/hugoboum/anaconda3/lib/python3.7/site-
packages/pandas/core/indexing.py:190: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: http://pandas.pydata.org/pandas-
docs/stable/indexing.html#indexing-view-versus-copy
  self._setitem_with_indexer(indexer, value)
```

Then I filter metropolitan departments only to have a respectable scale

[39]:
```
shapefile = shapefile[shapefile.code_insee.astype(int) < 100]
```

[40]:
```
shapefile.plot()
```

[40]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x7fe614597160>
```



Now, I will try to get a feature in the gas price dataframe describing the department of the stations. I will then groupby this feature and compute the department average price which I will then plot on the above graph.

[41]:
```
from shapely.geometry import shape,mapping, Point, Polygon, MultiPolygon
```

[42]:
```
prixall_pd['coordinates'] = prixall_pd[['longitude', 'latitude']].values.tolist()
```

[43]:
```
prixall_pd['coordinates'] = prixall_pd['coordinates'].apply(Point)
```

[44]:
```
prixall_pd = gpd.GeoDataFrame(prixall_pd, geometry='coordinates')
```

[45]:
```
prixall_pd.head()
```

[45]:

	id_pdv	cp	pop	latitude	longitude	id_carburant	nom_carburant	\
0	1000001	1000	R	46.20114	5.19791	1	Gazole	
1	1000001	1000	R	46.20114	5.19791	1	Gazole	
2	1000001	1000	R	46.20114	5.19791	1	Gazole	
3	1000001	1000	R	46.20114	5.19791	1	Gazole	
4	1000001	1000	R	46.20114	5.19791	1	Gazole	

	prix(millieuros)	year	month	week	week_index	day	prix_moyen_fr	\
0	1304	2014	1	1	1	2	1181.091131	
1	1304	2014	1	1	1	4	1181.091131	

```
2              1304  2014      1     1           1    5    1181.091131
3              1075  2014     12     1           1  364    1181.091131
4              1075  2014     12     1           1  365    1181.091131

   prix_moyen_station  price_index  prix_moyen_fr_weekindex  \
0         1129.333333    95.617798              1221.795512
1         1129.333333    95.617798              1221.795512
2         1129.333333    95.617798              1221.795512
3         1129.333333    95.617798              1221.795512
4         1129.333333    95.617798              1221.795512

                coordinates
0  POINT (5.19791 46.20114)
1  POINT (5.19791 46.20114)
2  POINT (5.19791 46.20114)
3  POINT (5.19791 46.20114)
4  POINT (5.19791 46.20114)
```

I have prepared the coordinates of the gas stations in order to search for the department to which it belongs.

Unfortunately, the following code, which should compute it,takes too long to finish...

dep_nom = []

dep_shape = []

for pt in prixall_pd['coordinates']:

```
for poly,dpt in zip(shapefile['geometry'],shapefile['nom']):

    if(pt.within(poly)):

        dep_nom.append(dpt)

        dep_shape.append(poly)
```

prixall_pd['dep_nom'] = dep_nom

prixall_pd['dep_shape'] = dep_shape

prixall_pd.head()

We will have to move on to modelisation...

3 Modeling and forecating using SparkML

First, I will try to free some memory

[46]:
```
import gc
```

[47]:
```
del [[prixall_pd,shapefile,codes]]
gc.collect()
```

[47]: 87

Next, I created the features to be used.

[48]:
```
feature_columns = prixall_ddf.columns
feature_columns.remove('prix(millieuros)')
feature_columns.remove('nom_carburant')
feature_columns.remove('pop')
feature_columns
```

[48]:
```
['id_pdv',
 'cp',
 'latitude',
 'longitude',
 'id_carburant',
 'year',
 'month',
 'week',
 'week_index',
 'day',
 'prix_moyen_fr',
 'prix_moyen_station',
 'price_index',
 'prix_moyen_fr_weekindex']
```

I will focus on one station and one gas type

[49]:
```
mdl_ddf = prixall_ddf.filter((F.col('id_pdv') == 1000001) & (F.
↪col('id_carburant') == 1))
```

[50]:
```
from pyspark.ml.feature import VectorAssembler
assembler = VectorAssembler(inputCols=feature_columns,outputCol="features")
```

[51]:
```
mdl_ddf = assembler.transform(mdl_ddf)
```

[52]:
```
mdl_ddf.show()
```

```
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+------------------+------------------+----------
--------+----------------------+--------------------+
| id_pdv|  cp|pop|latitude|longitude|id_carburant|nom_carburant|prix(millieuros)
|year|month|week|week_index|day|     prix_moyen_fr|prix_moyen_station|
price_index|prix_moyen_fr_weekindex|            features|
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+------------------+------------------+----------
--------+----------------------+--------------------+
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  2|1181.0911314644688|1129.3333333333333|
95.61779808921607|     1221.7955122777307|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  4|1181.0911314644688|1129.3333333333333|
95.61779808921607|     1221.7955122777307|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   1|         1|  5|1181.0911314644688|1129.3333333333333|
95.61779808921607|     1221.7955122777307|[1000001.0,1000.0...|
```

```
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|         1|364|1181.0911314644688|1129.3333333333333|
95.61779808921607|     1221.7955122777307|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1075|2014|   12|   1|         1|365|1181.0911314644688|1129.3333333333333|
95.61779808921607|     1221.7955122777307|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   2|         2|  6|  1176.55605922038|1160.3684210526317|
98.6241507116563|      1333.624184460261|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   2|         2|  7|  1176.55605922038|1160.3684210526317|
98.6241507116563|      1333.624184460261|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   2|         2| 10|  1176.55605922038|1160.3684210526317|
98.6241507116563|      1333.624184460261|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   2|         2| 11|  1176.55605922038|1160.3684210526317|
98.6241507116563|      1333.624184460261|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   3|         3|
13|1152.7277113373882|1156.0666666666666|100.28965689784664|
1323.5956594071386|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   3|         3|
15|1152.7277113373882|1156.0666666666666|100.28965689784664|
1323.5956594071386|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   3|         3|
16|1152.7277113373882|1156.0666666666666|100.28965689784664|
1323.5956594071386|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   3|         3|
18|1152.7277113373882|1156.0666666666666|100.28965689784664|
1323.5956594071386|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   4|         4| 20|  1154.81429475211|
1168.5|101.18510009012553|     1323.0351362683439|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   4|         4| 22|  1154.81429475211|
1168.5|101.18510009012553|     1323.0351362683439|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   4|         4| 23|  1154.81429475211|
1168.5|101.18510009012553|     1323.0351362683439|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   4|         4| 25|  1154.81429475211|
1168.5|101.18510009012553|     1323.0351362683439|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   4|         4| 26|  1154.81429475211|
```

```
1168.5|101.18510009012553|     1323.0351362683439|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   5|         5| 27|1160.9916199016054|            1157.0|
99.65618874131549|     1324.1694572217111|[1000001.0,1000.0...|
|1000001|1000|  R|46.20114|  5.19791|           1|       Gazole|
1304|2014|    1|   5|         5| 28|1160.9916199016054|            1157.0|
99.65618874131549|     1324.1694572217111|[1000001.0,1000.0...|
+-------+----+---+--------+---------+------------+-------------+----------------
+----+-----+----+----------+---+------------------+------------------+----------
--------+----------------------+--------------------+
only showing top 20 rows
```

We can notice the last column features containing every feature to be used. Next, we split train and test sets.

[53]:
```
mdl_train, mdl_test = mdl_ddf.randomSplit([0.7, 0.3])
```

I will use a linear regression to forecast gas prices with a L1 regularization
https://spark.apache.org/docs/1.5.2/ml-linear-methods.html

[80]:
```
from pyspark.ml.regression import LinearRegression
```

[81]:
```
algo = LinearRegression(featuresCol="features",regParam=0.3, elasticNetParam=1,␣
↪labelCol="prix(millieuros)")
```

[82]:
```
model = algo.fit(mdl_train)
```

Here are the fitted coefficients

[83]:
```
print("Intercept: " + str(model.intercept))
for col,par in zip(feature_columns,model.coefficients):
    print(col + ": " + str(par))
```

```
Intercept: -88.08199267847569
id_pdv: 0.0
cp: 0.0
latitude: 0.0
longitude: 0.0
id_carburant: 0.0
year: 0.0
month: 0.0
week: -0.06207364307391711
week_index: -0.022411729779243057
day: 0.0
prix_moyen_fr: 0.0
prix_moyen_station: 0.06599688931921874
price_index: 0.0
prix_moyen_fr_weekindex: 0.9856618433687934
```

After fitting, some evaluations are possible using the test data

[84]:
```
evaluation_summary = model.evaluate(mdl_test)
```

[85]:
```
evaluation_summary.rootMeanSquaredError
```

[85]:
```
27.975697901598092
```

[86]:
```
evaluation_summary.r2
```

[86]:
```
0.8992647524490768
```

And now, we can predict on the test data

[87]:
```
predictions = model.transform(mdl_test)
```

[88]:
```
predictions.select('prix(millieuros)','prediction').show()
```

```
+----------------+------------------+
|prix(millieuros)|        prediction|
+----------------+------------------+
|            1075|1190.6432258045627|
|            1304|1190.6432258045627|
|            1304| 1292.579092574777|
|            1304|1292.7626822555574|
|            1304|1292.7626822555574|
|            1299| 1293.037289537401|
|            1285|1288.8026906493064|
|            1285| 1296.176809313951|
|            1285| 1300.552758383151|
|            1285|1300.7639589474547|
|            1535|1300.7639589474547|
|            1285| 1295.863434055244|
|            1285|1285.8429585249012|
|            1285|1285.8429585249012|
|            1285|1285.8429585249012|
|            1275| 1273.137009322462|
|            1529| 1273.137009322462|
|            1269|1274.3592363658324|
|            1265|1271.5871499714817|
|            1265|1271.5871499714817|
+----------------+------------------+
only showing top 20 rows
```

4 Conclusion

It would take a bit too long to grid-search a perfect L1 penalization parameter, so we'll stop at that.

Thank you for you attention and Happy new year !