Mathematics in Machine Learning
Project Report

Hourly Interstate 94 Westbound traffic volume analysis

Carlo Peluso S280203

January 2022

Master's Degree in Data Science and Engineering
01TXGSM Mathematics in Machine Learning Course

Contents

List of Figures

List of Tables

1 Introduction

The aim of this project is to analyse the Metro Interstate Traffic Volume dataset - downloaded from the UCI repository - which was designed to predict the hourly Interstate 94 Westbound traffic volume by means of weather and holidays information.

Predictions made through different regressors are also produced and few metrics are exploited to evaluate the performances of each model.

The following sections will discuss about Data Exploration and Data Cleaning (section 2), Data Processing (section 3), Model Selection and Tuning (section 4).

2 Data Exploration and Data Cleaning

In this section we will explore the data and extract information from them, by analyzing the attributes by themselves and - at times - jointly. Moreover, during the analysis, we will search for inconsistencies, missing values and we will handle them in the proper way.

2.1 Dataset overview

The Metro Interstate Traffic Volume dataset contains 48204 instances - with no missing values -, composed of 8 attributes and a target attribute. This dataset is mainly related to regression tasks in which the target attribute - the hourly traffic volume at Interstate 94 Westbound - is the value of interest to be predicted.

The attributes of the dataset are the following:

Attributes		
Attribute Name	Data Type	Notes
holiday	Categorical	Specifies the holiday name. If the date is not an holiday, is None.
temp	float	Indicates the temperature in Kelvin.
rain_1h	float	Hourly amount of rain (in mm.)
snow_1h	float	Hourly amount of snow (in mm.)
clouds_all	integer	Indicates the percentage of sky covered by the clouds.
weather_main	Categorical	Short description of the weather.
weather_description	Categorical	Long description of the weather.
date_time	String	Date (YYYY-MM-dd hh:mm:ss) in which the data was collected.
traffic_volume	Integer	Hourly traffic volume.

Table 1

Data Set Characteristics:	Multivariate, Sequential, Time-Series	**Number of Instances:**	48204	**Area:**	N/A
Attribute Characteristics:	Integer, Real	**Number of Attributes:**	9	**Date Donated**	2019-05-07
Associated Tasks:	Regression	**Missing Values?**	N/A	**Number of Web Hits:**	79315

Figure 1: UCI Repository basic information about the dataset

2.2 Dates

The **date_time** data attribute indicates the date in which the observations were sampled, for two time intervals ranging from 2012/10 to 2014/08 and from 2015/06 to 2018/09. As shown in the table above, the format of the **date_time** attribute is YYYY-MM-dd hh:mm:ss, but the precision of the attribute is restricted to the hours digit.

Thanks to the hourly precision of this attribute, it is not necessary to perform aggregations on the data: the task is to predict the *hourly* traffic volume and we already have all the attributes defined such that their value explains their behaviour within the hour.

2.3 Holiday

The **holiday** data attribute specifies the holiday name, if the date is an holiday. Instead, if the date isn't an holiday, the value of this attribute is None.

In this dataset, 12 distinct values (including the None value) are collected for this attribute: 'Columbus Day', 'Veterans Day', 'Thanksgiving Day', 'Christmas Day', 'New Years Day', 'Washingtons Birthday', 'Memorial Day', 'Independence Day', 'State Fair', 'Labor Day', 'Martin Luther King Jr Day'.
However, the occurrences of the holidays in this dataset are very rare with respect to the weekdays (i.e., None), as we can see from the following chart (Figure 2):

Figure 2: Count plot of the data occurrences for each holiday

Due to the rarity of those labels, I decided to apply a simple binning technique to this attribute. Instead of keeping all the holiday names, I reformatted the values as follows:

$$\text{attribute value} = \begin{cases} 1, & \text{if date is an holiday} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

However, since the 99.87% of the rows are weekdays and the remaining 0.12% of the rows refers to an holiday observation, the attribute remains strongly imbalanced.

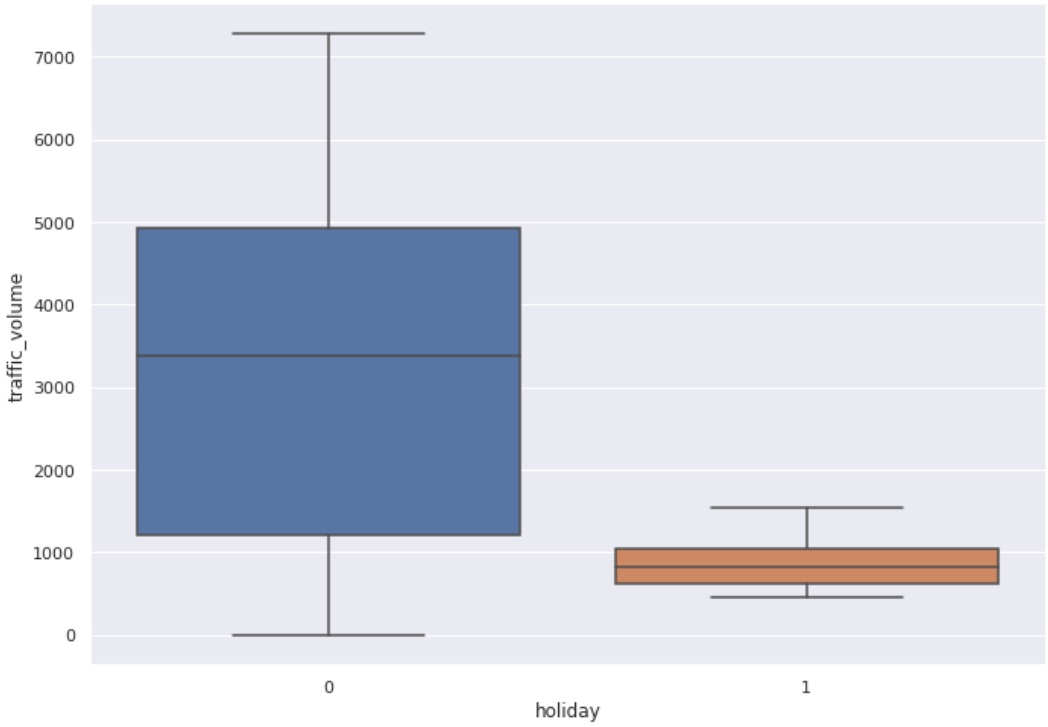

Figure 3: The traffic volume distributions over class holiday = 0 and holiday = 1

As we can see from the box plot above (Figure 3), it emerges that the distribution of the hourly traffic volume for class 0 (i.e., not-holidays) is very different from the distribution for class 1 (i.e., holidays). Hence, the holiday attribute highlights the fact that when predicting the hourly traffic volume for a holiday, this value should be underestimated.

2.4 Temperature

The **temp** data attribute indicates the temperature in Kelvin registered during the related day and hour. In this project, those values where remapped into the Celsius scale for the sake of simplicity.

Taking a look at the distribution of the temperature values (Figure 4), we can see that there are some outliers located near the absolute zero value ($-273.15°$). Clearly, these are inconsistencies inside the data that must be

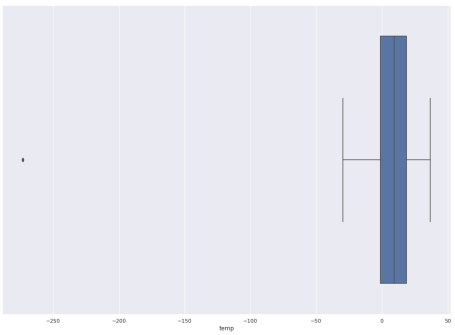

Figure 4: The original temperature's distribution.

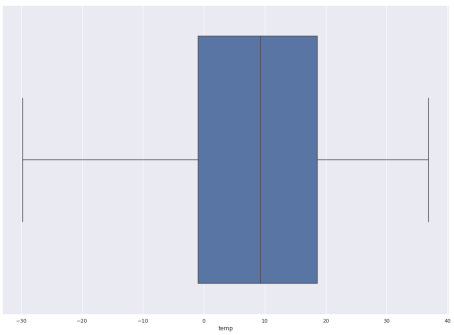

Figure 5: The temperature's distribution without outliers.

removed.

Once removed those inconsistencies, found in 10 rows of the dataset, the distribution of the temperatures is way more reasonable (Figure 5) and acceptable.

Moreover, looking at the density of the cleaned distribution (Figure 6), we can see that the values are ranging from $-30°$ to $30°$ (which is quite reasonable) with peaks around $0°$ and $20°$.

Finally, in order to confirm the consistency of this attribute, we can take a look at the evolution of the temperatures during time (Figure 7). From this graph, we can see the typical seasonal nature of the temperatures, which confirms the correctness of the data.

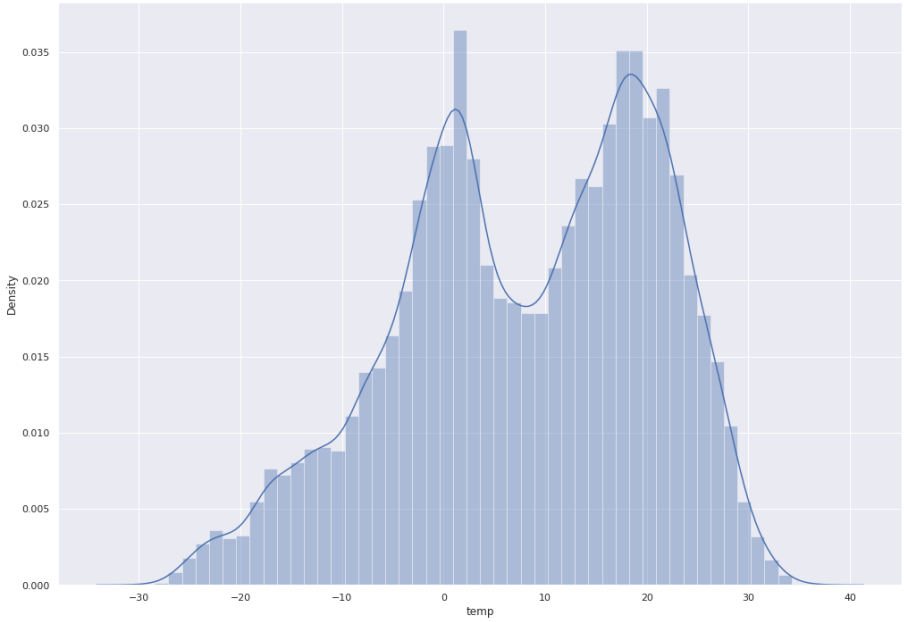

Figure 6: The density of the temperature's values distribution after removing the outliers.

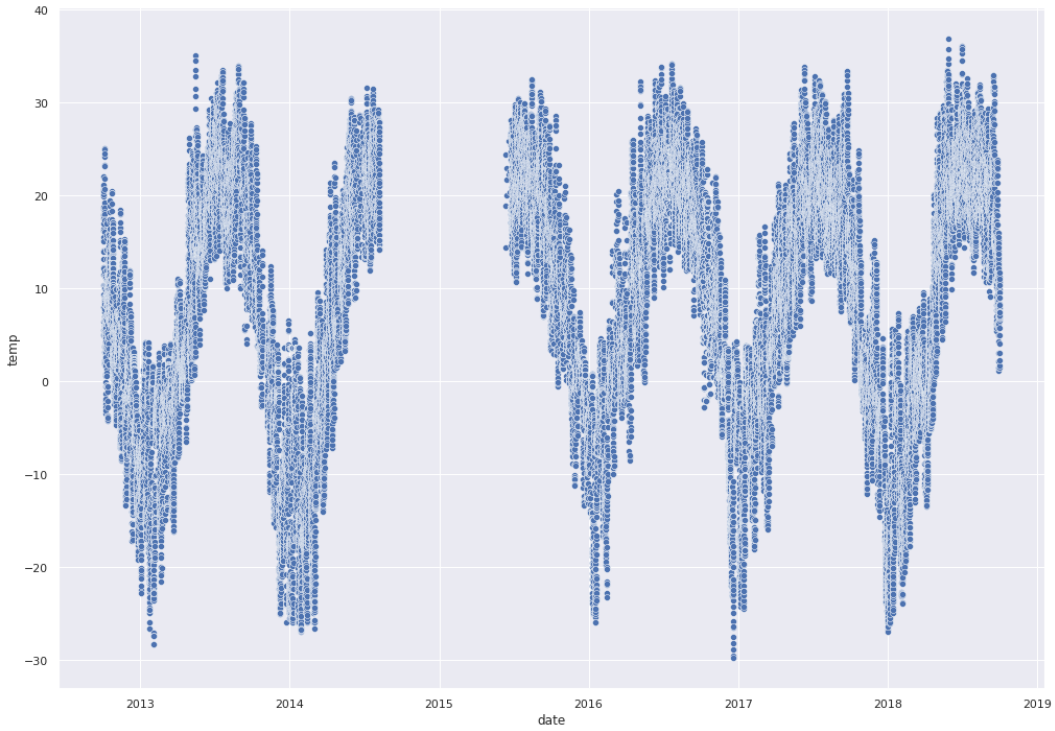

Figure 7: The evolution of the temperatures during time.

2.5 Weather

The **weather_main** and **weather_description** categorical attributes give information about the weather. As we can see from Table 2, these attributes are related through a $1 : N$ relationship, hence a particular value of weather_description (e.g., "scattered clouds") could only be related to a particular value of weather_main (in this example, "Clouds").

It immediately stands out that there is a problem in the weather_description attribute when the weather_main attribute is equal to "Clear". In fact, the weather descriptions associated to this value are redundant (['sky is clear', 'Sky is Clear']), since they do not differ in meaning but only in casing.
Lowering every record of the weather_description attribute, we end up with consistent values for this column (Table 3).

weather_main	**weather_description**
Clouds	['scattered clouds', 'broken clouds', 'overcast clouds', 'few clouds']
Clear	['sky is clear', 'Sky is Clear']
Mist	['mist']
Rain	['light rain', 'proximity shower rain', 'moderate rain', 'heavy intensity rain', 'freezing rain', 'light intensity shower rain', 'very heavy rain']
Snow	['heavy snow', 'snow', 'shower snow', 'light rain and snow', 'light snow', 'light shower snow', 'sleet']
Drizzle	['light intensity drizzle', 'drizzle', 'heavy intensity drizzle', 'shower drizzle']
Haze	['haze']
Thunderstorm	['proximity thunderstorm', 'thunderstorm with light rain', 'proximity thunderstorm with rain', 'thunderstorm with heavy rain', 'thunderstorm with rain', 'proximity thunderstorm with drizzle', 'thunderstorm', 'thunderstorm with light drizzle', 'thunderstorm with drizzle']
Fog	['fog']
Smoke	['smoke']
Squall	['SQUALLS']

Table 2:
Possible values of the weather_main and weather_description attributes.

weather_main	**weather_description**
Clouds	['scattered clouds', 'broken clouds', 'overcast clouds', 'few clouds']
Clear	['sky is clear']
Mist	['mist']
Rain	['light rain', 'proximity shower rain', 'moderate rain', 'heavy intensity rain', 'freezing rain', 'light intensity shower rain', 'very heavy rain']
Snow	['heavy snow', 'snow', 'shower snow', 'light rain and snow', 'light snow', 'light shower snow', 'sleet']
Drizzle	['light intensity drizzle', 'drizzle', 'heavy intensity drizzle', 'shower drizzle']
Haze	['haze']
Thunderstorm	['proximity thunderstorm', 'thunderstorm with light rain', 'proximity thunderstorm with rain', 'thunderstorm with heavy rain', 'thunderstorm with rain', 'proximity thunderstorm with drizzle', 'thunderstorm', 'thunderstorm with light drizzle', 'thunderstorm with drizzle']
Fog	['fog']
Smoke	['smoke']
Squall	['squalls']

Table 3:
Cleaned values of the weather_main and weather_description attributes.

Then, we can analyze the number of times each weather_main category is present on the dataset (Figure 8): reasonably, the categories "Clouds" and "Clear" are the most present inside the dataset, since they are the most frequent weather phenomena, whereas the categories "Smoke" and "Squall" are the rarest weather phenomena, appearing 20 and 4 times respectively.

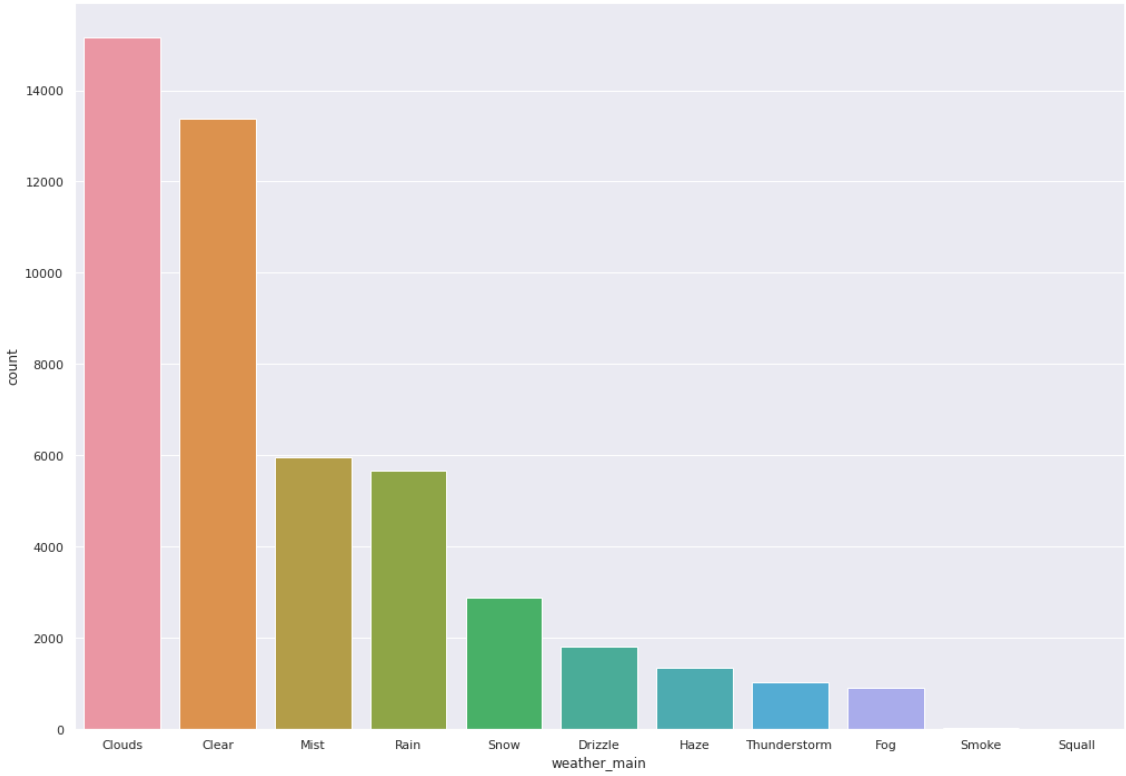

Figure 8: Count plot of the data occurrences for each weather_main category.

In order to validate the consistency of the weather_main attribute, we will evaluate the trends resulting from the "Rain" and "Snow" classes during time. The class "Snow", in particular, strictly depends on date - occurrences of "Snow" in summer months like June, July and August could be a sign of data inconsistency, also with respect to the temperatures distribution seen before.

Figure 9 and Figure 10 display, respectively, the trend of "Rain" and "Snow" during time. The values were retrieved by aggregating the occurrences of the classes monthly and summing them up. Please notice that the values between 2014/08 and 2015/06 are interpolated, since they doesn't appear in the dataset.

As we can see from the plots, the "Rain" occurrences are distributed almost equally during the majority of the central months of the year. During the last and first months of the year, instead, the occurrences reach always a minimum: this behaviour is explained by the evolution of the "Snow" occurrences. In fact, as we expected, they are concentrated during the last and first months of the year.

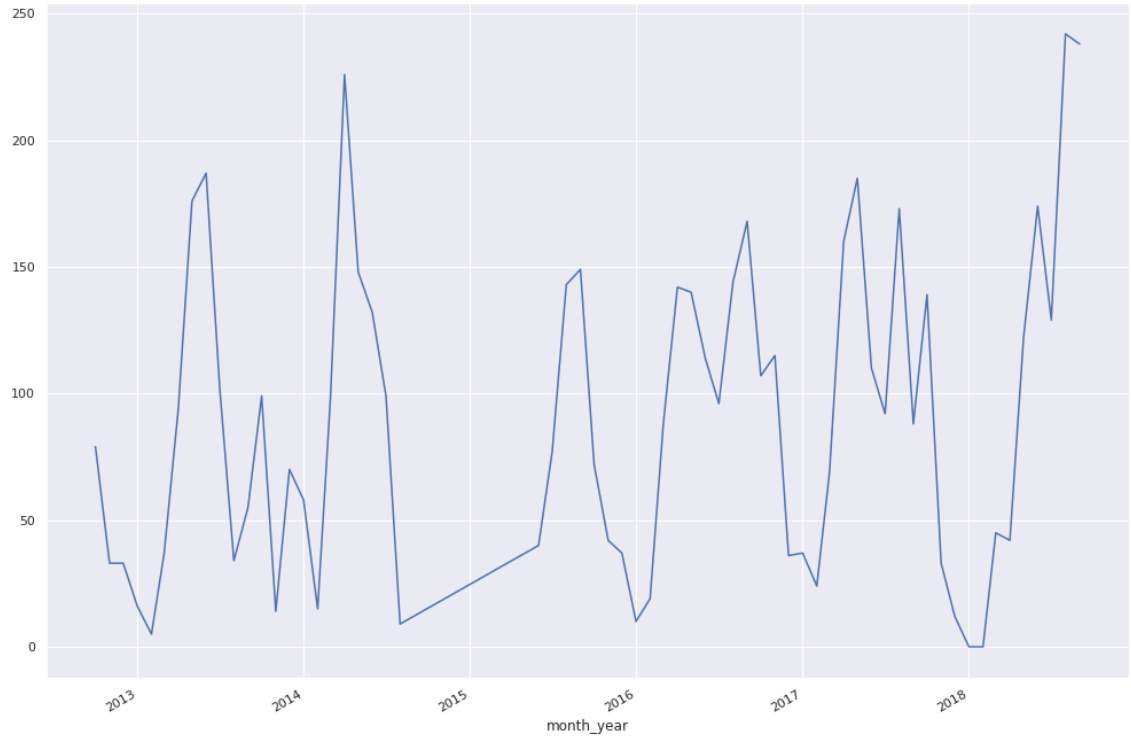

Figure 9: The monthly evolution of rain occurrences during time.

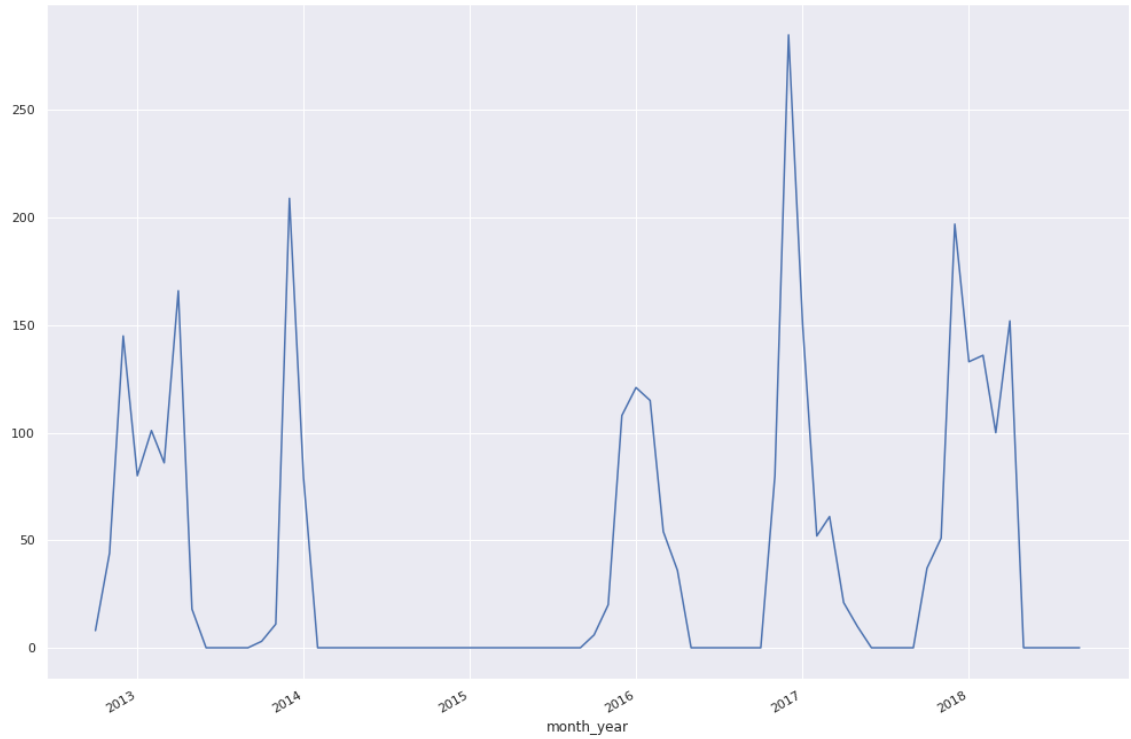

Figure 10: The monthly evolution of snow occurrences during time.

Analyzing the weather_description number of occurrences for each category (Figure 11), we must keep in mind that some weather_main categories are related to only one weather_description category: for example, "Clear" is only related to the "sky is clear" category. Hence, "sky is clear" results to be the most present category within the weather_description attribute because it is present as much as "Clear", the second most occurring weather_main attribute value.
For this reason, "mist" is the second attribute that occurs more between the weather_description categories, for example.

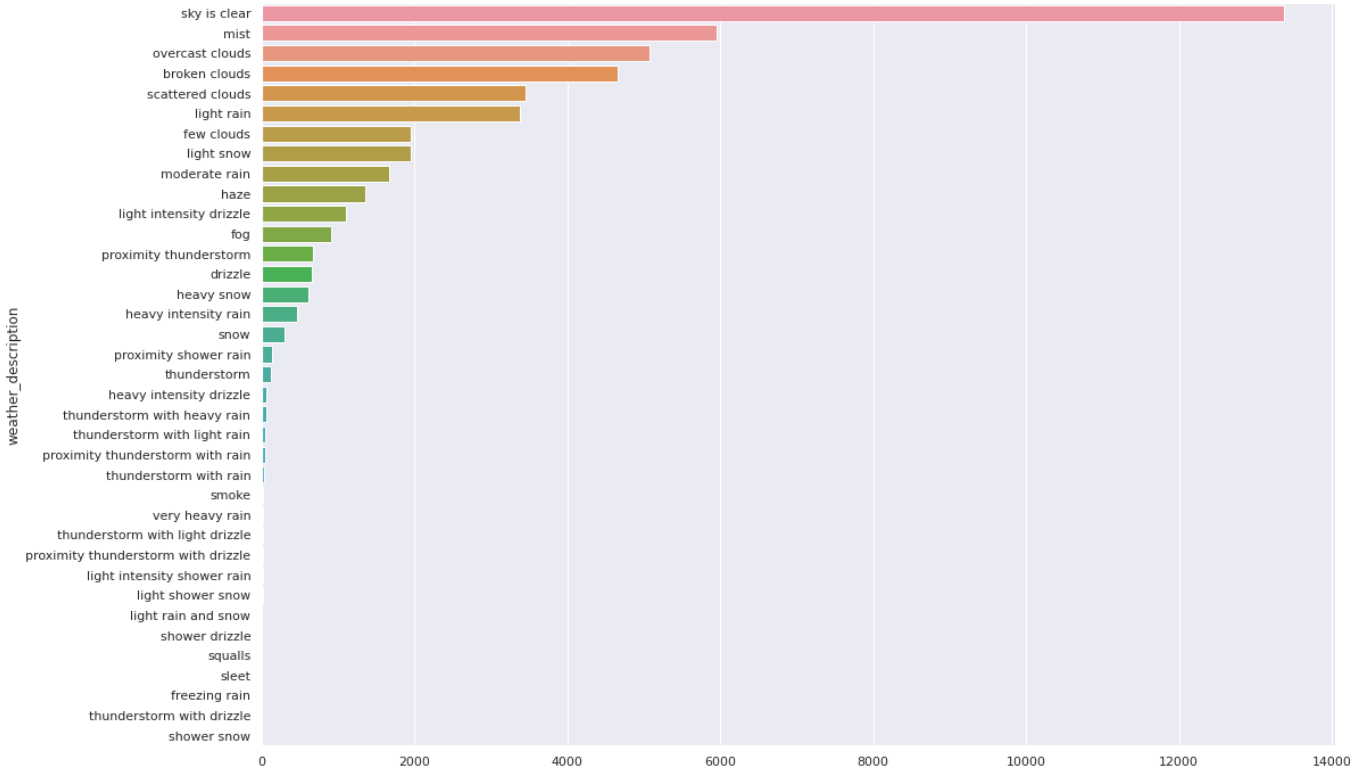

Figure 11: Count plot of the data occurrences for each weather_description category.

2.6 Rain and Snow

2.6.1 Rain

The attribute **rain_1h** indicates the hourly amount of rain (in mm.).

By verifying the original distribution of this attribute (Figure 12) we can immediately identify some didn't expected outliers in the data.

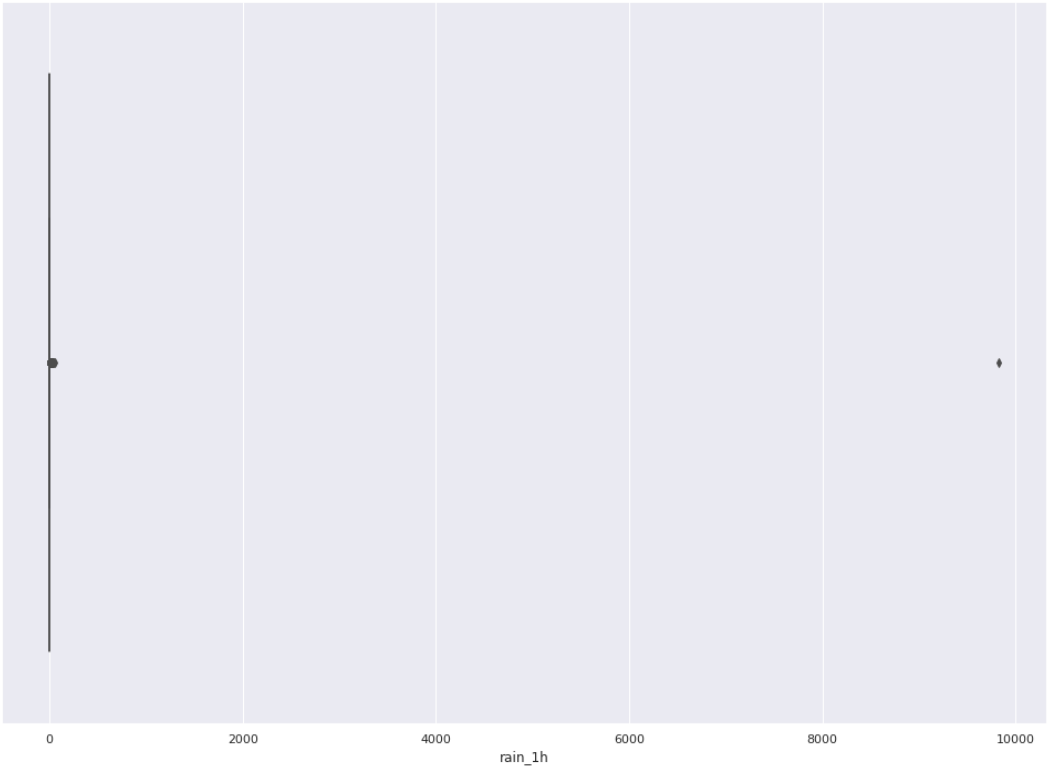

Figure 12: The original hourly rain distribution.

In order to investigate the nature of these outliers, we can perform a query on the dataset by filtering on the rows with weather_description equal to "very heavy rain" and then sorting the values by "rain_1h".
It emerges that the single outlier is at least 176 times larger than the other values (Figure 13): for this reason, the row is considered inconsistent and it was removed.

Once removed the outlier, we can analyze the cleaned distribution of the rain_1h attribute (Figures 14, 15), that immediately appears highly left-skewed.
In particular, from the distribution emerge the following descriptive statistics (Table 4) that confirms the skewed nature of this attribute.

Statistics	Value
mean	0.130510
std	1.004116
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	55.630000

Table 4:
Possible values of the weather_main and weather_description attributes.

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	date	day_of_the_week	hour	year	month	month_year
7133	0	18.61	16.38	0.0	76	Rain	very heavy rain	635	2013-06-22 05:00:00	1	5	2013	6	2013-06-01
25481	0	21.77	16.51	0.0	32	Rain	very heavy rain	5724	2016-08-04 07:00:00	0	7	2016	8	2016-08-01
14853	0	25.06	18.03	0.0	0	Rain	very heavy rain	6443	2014-06-02 16:00:00	3	16	2014	6	2014-06-01
25639	0	19.55	18.42	0.0	80	Rain	very heavy rain	1904	2016-08-10 21:00:00	6	21	2016	8	2016-08-01
10734	0	-10.50	18.80	0.0	64	Rain	very heavy rain	2755	2013-12-16 19:00:00	2	19	2013	12	2013-12-01
7664	0	19.67	19.90	0.0	20	Rain	very heavy rain	542	2013-07-14 05:00:00	0	5	2013	7	2013-07-01
25961	0	20.92	20.07	0.0	40	Rain	very heavy rain	1520	2016-08-23 22:00:00	5	22	2016	8	2016-08-01
16923	0	21.97	20.24	0.0	90	Rain	very heavy rain	4302	2015-07-28 07:00:00	2	7	2015	7	2015-07-01
7670	0	19.72	21.42	0.0	90	Rain	very heavy rain	1745	2013-07-14 07:00:00	0	7	2013	7	2013-07-01
16300	0	21.72	23.80	0.0	90	Rain	very heavy rain	346	2015-07-06 03:00:00	1	3	2015	7	2015-07-01
7667	0	19.50	25.32	0.0	8	Rain	very heavy rain	958	2013-07-14 06:00:00	0	6	2013	7	2013-07-01
17437	0	21.34	25.46	0.0	90	Rain	very heavy rain	2118	2015-08-16 21:00:00	4	21	2015	8	2015-08-01
16504	0	18.87	27.57	0.0	90	Rain	very heavy rain	492	2015-07-13 00:00:00	1	0	2015	7	2015-07-01
10806	0	-8.65	28.70	0.0	64	Rain	very heavy rain	1190	2013-12-19 23:00:00	5	23	2013	12	2013-12-01
25779	0	24.56	31.75	0.0	0	Rain	very heavy rain	4913	2016-08-16 17:00:00	5	17	2016	8	2016-08-01
7179	0	22.67	44.45	0.0	76	Rain	very heavy rain	4802	2013-06-24 11:00:00	3	11	2013	6	2013-06-01
8247	0	15.95	55.63	0.0	68	Rain	very heavy rain	315	2013-08-07 02:00:00	0	2	2013	8	2013-08-01
24872	0	28.96	9831.30	0.0	75	Rain	very heavy rain	5535	2016-07-11 17:00:00	0	17	2016	7	2016-07-01

Figure 13: rain_1h outliers analysis.

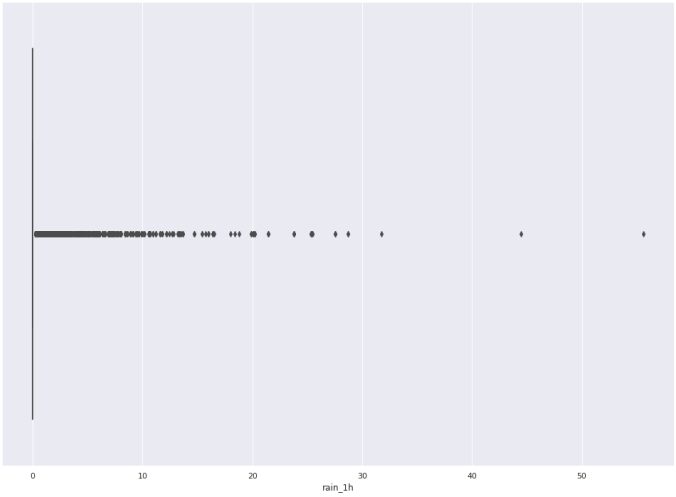

Figure 14:
The hourly rain distribution without outliers.

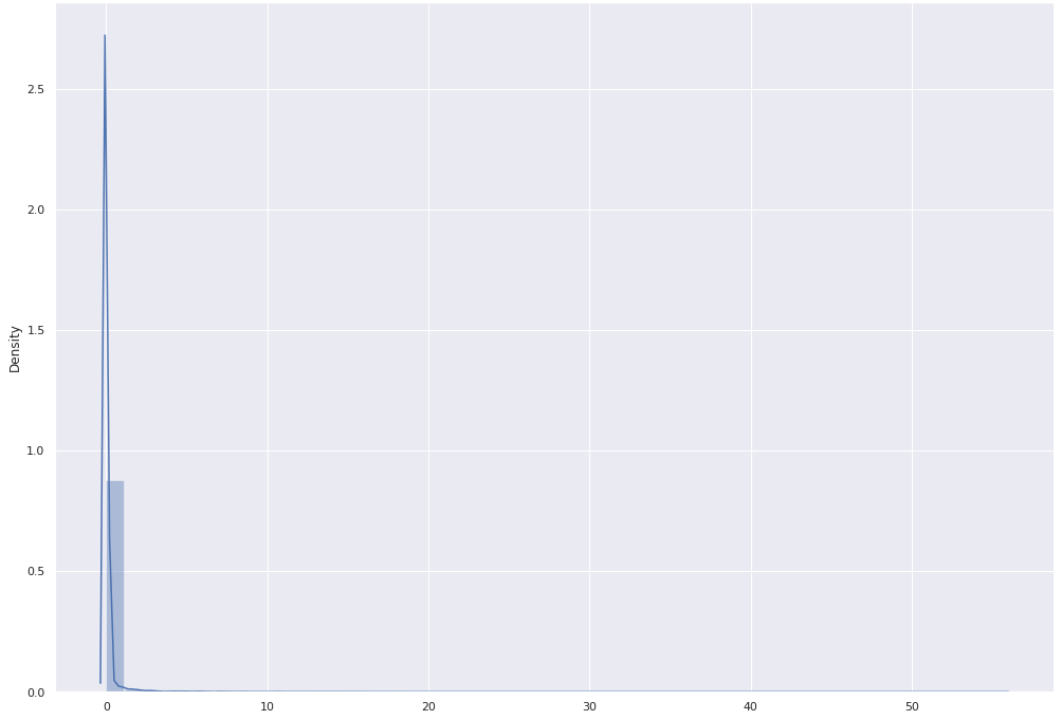

Figure 15:
The density of the rain's values distribution without outliers.

Furthermore, analyzing this attribute in conjunction with the weather_main attribute, we can highlight some severe inconsistencies: in fact, if we select all the rows matching the conditions

$$\text{rain\_1h} == 0 \text{ AND } \text{weather\_main} == \text{"Rain"} \tag{2}$$

we can find the rows flagged as weather_main == "Rain" but with 0 as rain_1h, which is a clear inconsistency. Grouping the results by weather_description, the following results are produced (Table 5):

weather_description	**Count**
light rain	2228
moderate rain	1029
heavy intensity rain	226
proximity shower rain	116
light intensity shower rain	8
freezing rain	2

Table 5:
Grouped results of the query rain_1h == 0 AND weather_main == "Rain"

Having verified the consistency of the attributes weather_main and weather_description, from this results we can suppose that the data for the column rain_1h were not collected in a proper way.

2.6.2 Snow

The attribute **snow_1h** indicates the hourly amount of snow (in mm.).

By verifying the distribution of this attribute (Figure 16) and its density (Figure 17), we can immediately appreciate the left-skewed nature of this attribute.

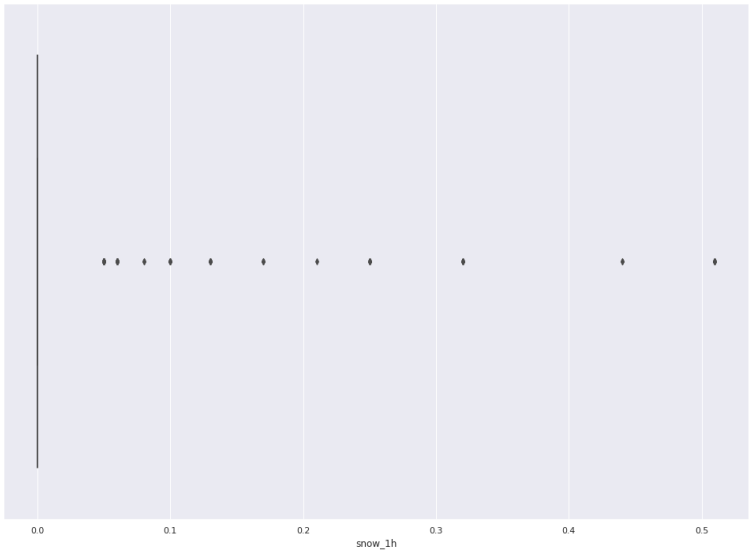

Figure 16:
The hourly snow distribution.

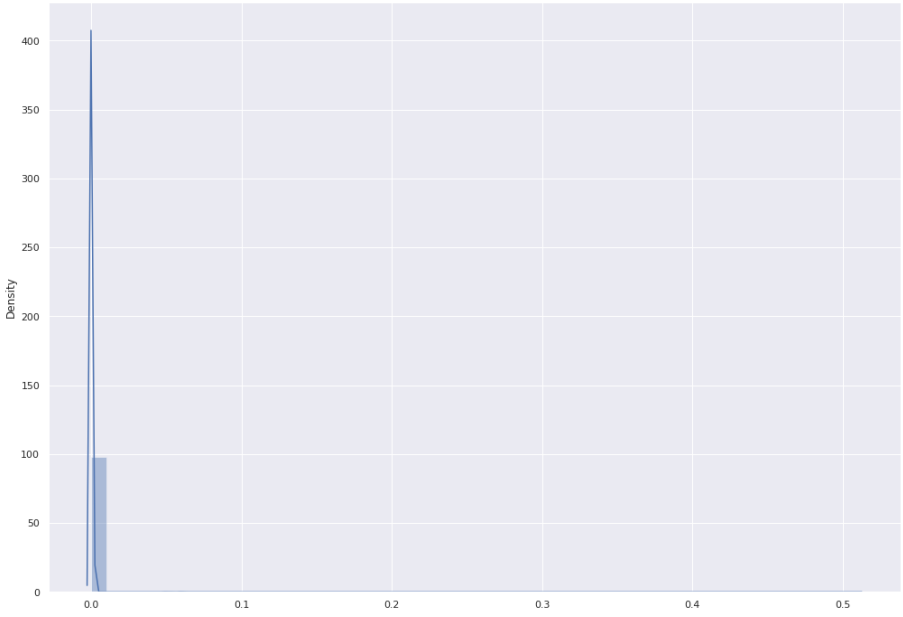

Figure 17:
The density of the snow's values distribution.

Performing the same analysis made for the attribute rain_1h, hence analyzing this attribute in conjunction with the weather_main attribute, we can again highlight some severe inconsistencies: in fact, if we select all the rows matching the conditions

$$\text{snow\_1h} == 0 \text{ AND weather\_main} == \text{"Snow"} \tag{3}$$

we can find the rows flagged as weather_main == "Snow" but with 0 as snow_1h, which is a clear inconsistency. Grouping the results by weather_description, the following results are produced (Table 6):

weather_description	**Count**
light snow	1919
heavy snow	616
snow	288
light shower snow	11
light rain and snow	6
sleet	3
shower snow	1

Table 6:
Grouped results of the query snow_1h == 0 AND weather_main == "Snow"

Again, from this results we can suppose that the data for the column snow_1h were not collected in a proper way.

2.7 Clouds

The attribute **clouds_all** indicates the percentage of sky covered by the clouds in the given hour.

At a first glance, the values of this attribute seem reasonably distributed (Figure 18):

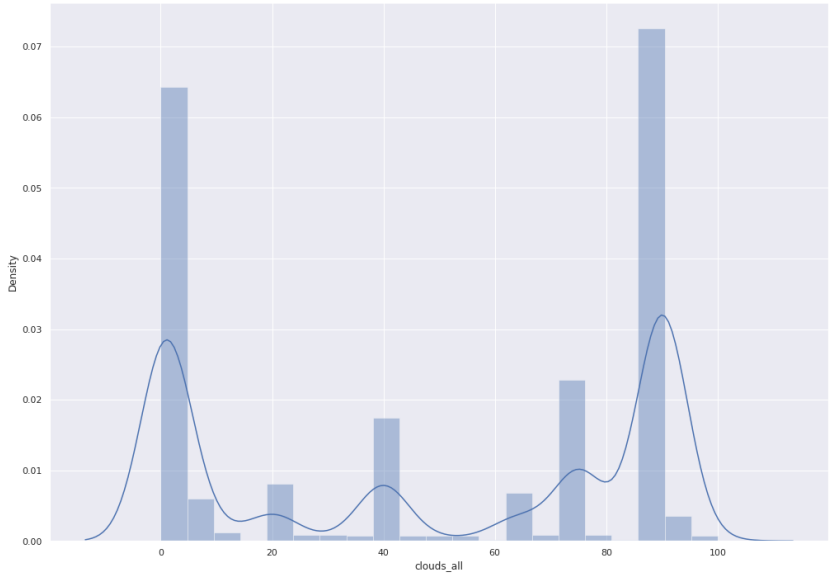

Figure 18:
The density of the clouds' values distribution.

Moreover, in order to verify the consistency of this attribute with respect to weather_main, we can ascertain that there aren't rows in which this attribute is 0 and weather_main is "Clouds" (Figure 19):

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	date	day_of_the_week	hour	year	month	month_year
3292	0	-28.33	0.0	0.0	11	Clouds	few clouds	354	2013-02-02 03:00:00	2	3	2013	2	2013-02-01
3293	0	-28.33	0.0	0.0	11	Clouds	few clouds	417	2013-02-02 04:00:00	2	4	2013	2	2013-02-01
3294	0	-28.33	0.0	0.0	11	Clouds	few clouds	678	2013-02-02 05:00:00	2	5	2013	2	2013-02-01
3117	0	-25.99	0.0	0.0	11	Clouds	few clouds	248	2013-01-23 02:00:00	2	2	2013	1	2013-01-01
3116	0	-25.99	0.0	0.0	11	Clouds	few clouds	346	2013-01-23 01:00:00	2	1	2013	1	2013-01-01
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
3250	0	-13.15	0.0	0.0	100	Clouds	overcast clouds	5330	2013-01-31 08:00:00	3	8	2013	1	2013-01-01
3272	0	-23.24	0.0	0.0	100	Clouds	overcast clouds	6681	2013-02-01 07:00:00	1	7	2013	2	2013-02-01
3273	0	-23.24	0.0	0.0	100	Clouds	overcast clouds	5948	2013-02-01 08:00:00	1	8	2013	2	2013-02-01
3254	0	-11.62	0.0	0.0	100	Clouds	overcast clouds	4962	2013-01-31 13:00:00	3	13	2013	1	2013-01-01
1810	0	-2.05	0.0	0.0	100	Clouds	overcast clouds	3137	2012-12-08 19:00:00	6	19	2012	12	2012-12-01

Figure 19:
Rows of the dataset having weather_main == "Clouds", sorted by clouds_all.

Furthermore, comparing this attribute with different weather_main values, some inconsistencies arise: for instance, querying the dataset by filtering for rows having weather_main == "Clear" and clouds_all > 10, 45 rows are returned (Figure 20): this is an inconsistency on the data and those rows must be removed.

	holiday	temp	rain_1h	snow_1h	clouds_all	weather_main	weather_description	traffic_volume	date	day_of_the_week	hour	year	month	month_year
927	0	1.26	0.0	0.0	98	Clear	sky is clear	356	2012-11-06 03:00:00	4	3	2012	11	2012-11-01
928	0	1.26	0.0	0.0	98	Clear	sky is clear	788	2012-11-06 04:00:00	4	4	2012	11	2012-11-01
929	0	1.26	0.0	0.0	98	Clear	sky is clear	2565	2012-11-06 05:00:00	4	5	2012	11	2012-11-01
930	0	1.26	0.0	0.0	98	Clear	sky is clear	5330	2012-11-06 06:00:00	4	6	2012	11	2012-11-01
1129	0	9.59	0.0	0.0	98	Clear	sky is clear	5242	2012-11-12 06:00:00	3	6	2012	11	2012-11-01
1128	0	9.59	0.0	0.0	98	Clear	sky is clear	2467	2012-11-12 05:00:00	3	5	2012	11	2012-11-01
1127	0	9.59	0.0	0.0	98	Clear	sky is clear	761	2012-11-12 04:00:00	3	4	2012	11	2012-11-01
1126	0	9.59	0.0	0.0	98	Clear	sky is clear	341	2012-11-12 03:00:00	3	3	2012	11	2012-11-01
925	0	1.26	0.0	0.0	98	Clear	sky is clear	313	2012-11-06 01:00:00	4	1	2012	11	2012-11-01
1125	0	9.59	0.0	0.0	98	Clear	sky is clear	263	2012-11-12 02:00:00	3	2	2012	11	2012-11-01
1124	0	9.59	0.0	0.0	98	Clear	sky is clear	352	2012-11-12 01:00:00	3	1	2012	11	2012-11-01
1130	0	0.75	0.0	0.0	99	Clear	sky is clear	5390	2012-11-12 07:00:00	3	7	2012	11	2012-11-01
1132	0	0.75	0.0	0.0	99	Clear	sky is clear	2096	2012-11-12 09:00:00	3	9	2012	11	2012-11-01
1131	0	0.75	0.0	0.0	99	Clear	sky is clear	2029	2012-11-12 08:00:00	3	8	2012	11	2012-11-01
934	0	1.76	0.0	0.0	99	Clear	sky is clear	4612	2012-11-06 10:00:00	4	10	2012	11	2012-11-01
933	0	1.76	0.0	0.0	99	Clear	sky is clear	5720	2012-11-06 09:00:00	4	9	2012	11	2012-11-01
932	0	1.76	0.0	0.0	99	Clear	sky is clear	6182	2012-11-06 08:00:00	4	8	2012	11	2012-11-01
931	0	1.76	0.0	0.0	99	Clear	sky is clear	6181	2012-11-06 07:00:00	4	7	2012	11	2012-11-01
184	0	6.52	0.0	0.0	99	Clear	sky is clear	4632	2012-10-10 10:00:00	1	10	2012	10	2012-10-01
313	0	8.57	0.0	0.0	100	Clear	sky is clear	4823	2012-10-15 12:00:00	6	12	2012	10	2012-10-01

Figure 20:
Rows of the dataset having weather_main == "Clear" and clouds_all > 10, tail.

3 Data Processing

In the previous sections the attributes of the Metro Interstate Traffic Volume dataset were analyzed and - where it was necessary - cleaned.

Some attributes resulted to be inconsistent and / or with skewed distributions, whereas other attributes resulted to be imbalanced but meaningful.

In this section we are going to prepare the data for the training phase, handling the categorical attributes, scaling the data and applying and evaluating some dimensionality reduction techniques.

3.1 Feature expansion

The date_time attribute appearing in the dataset is a concise but very relevant information we have at our disposal.

In order to extract as much information as possible from this attribute, I decided to expand this raw attribute into 4 more derived attributes: 'day_of_the_week', 'hour', 'year', 'month'.

This process is useful for several reasons: firstly because we have now 4 more attributes that can potentially be correlated with our target variable (i.e. the hourly traffic volume). Moreover, because now we can provide to the regressor 4 well defined meaningful attributes: in fact, we expect that the day of the week in which an observation is made has an impact on the traffic volume, for instance. Same thing for the "hour" and "month" attributes.

3.2 Handling categorical attributes

As shown in the previous section, 3 attributes of the dataset are categorical (holiday, weather_main, weather_description).
In particular, the attribute "weather_main" has 11 unique values and the attribute "weather_description" has 37 unique values.
Since the algorithms we will use cannot handle string data types, we will convert those categorical values into integers. We are not going to convert also the "holiday" attribute, because it is already a boolean attribute. Let's look at the new mappings:

Integer	**weather_main Category**
0	Clouds
1	Clear
2	Rain
3	Drizzle
4	Mist
5	Haze
6	Fog
7	Thunderstorm
8	Snow
9	Squall
10	Smoke

Table 7: Mapping of the weather_main attribute.

Integer	**weather_description Category**
0	scattered clouds
1	broken clouds
2	overcast clouds
3	sky is clear
4	few clouds
5	light rain
6	light intensity drizzle
7	mist
8	haze
9	fog
10	proximity shower rain
11	drizzle
12	moderate rain
13	heavy intensity rain
14	proximity thunderstorm
15	thunderstorm with light rain
16	proximity thunderstorm with rain
17	heavy snow
18	heavy intensity drizzle
19	snow
20	thunderstorm with heavy rain
21	freezing rain
22	shower snow
23	light rain and snow
24	light intensity shower rain
25	squalls
26	thunderstorm with rain
27	proximity thunderstorm with drizzle
28	thunderstorm
29	very heavy rain
30	thunderstorm with light drizzle
31	light snow
32	thunderstorm with drizzle
33	smoke
34	shower drizzle
35	light shower snow
36	sleet

Table 8: Mapping of the weather_description attribute.

3.3 Feature selection

3.3.1 Pearson Correlation Matrix

Once the dataset is cleaned and every attribute is numeric, we can inspect the correlations between attributes through the Pearson Correlation Matrix:

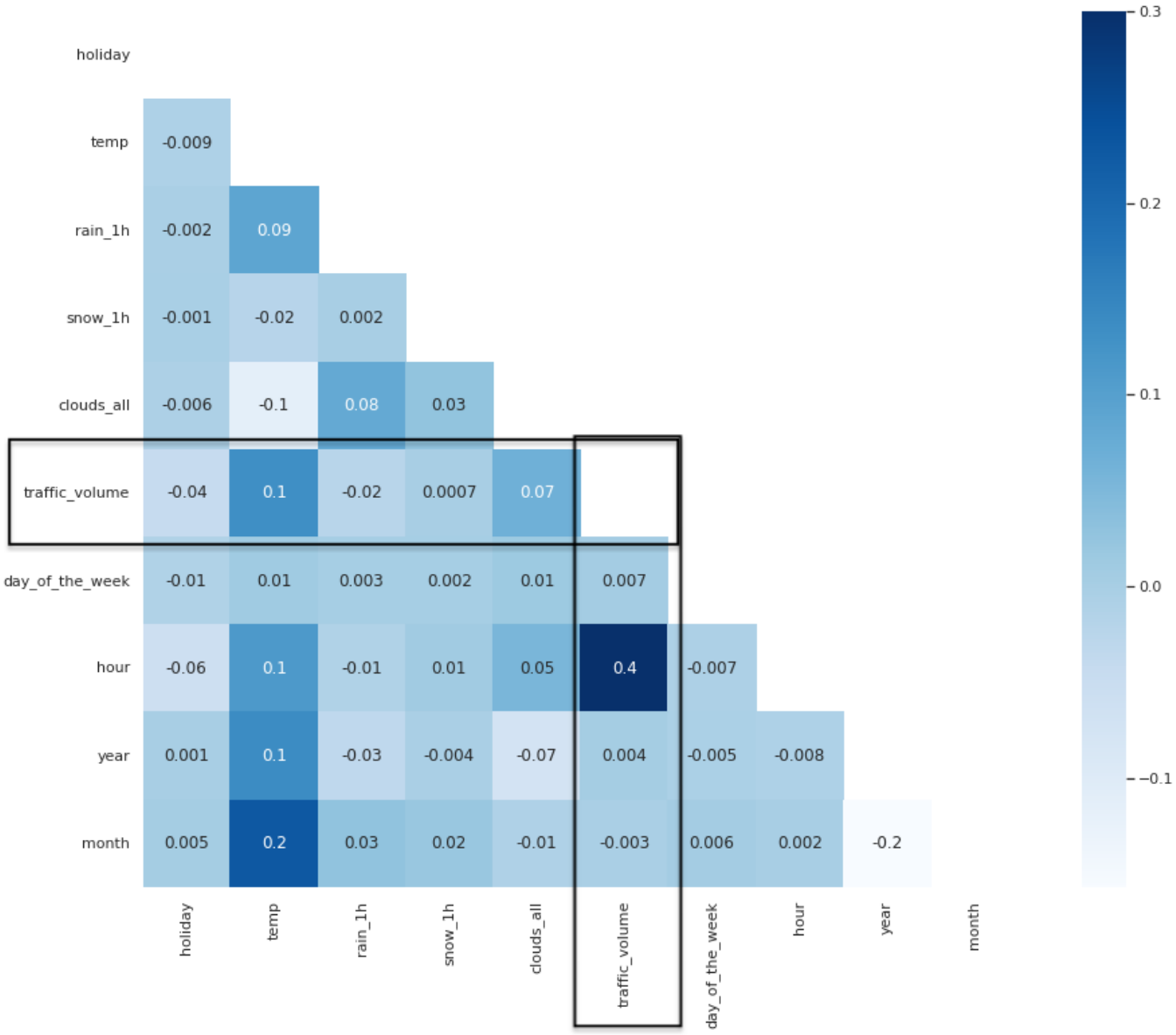

Figure 21:
Pearson Correlation Matrix

By means of the Pearson Correlation Matrix, we can derive some information: first of all, we can see that the attributes are not so correlated - not considering the temperature, that is obviously correlated with the month and the hour.
The target attribute (traffic_volume), however, is strongly correlated with the hour, as we expected.

This is reasonable, since we expect that the volume in the same day but at different hours (e.g. 2 a.m. and 2 p.m.) is very different.

3.3.2 Feature importances

In order to ascertain the meaningfulness of the dataset's attributes, it is possible to calculate the feature importances through a fitted model.
The feature importances simply represent the "importance" of each feature, in which an high score suggest that the specific feature will have a larger effect on the model that is being used to predict a target variable.
In order to calculate the feature importances, I trained a RandomForestRegressor (from the scikit-learn Python package). These are the results:

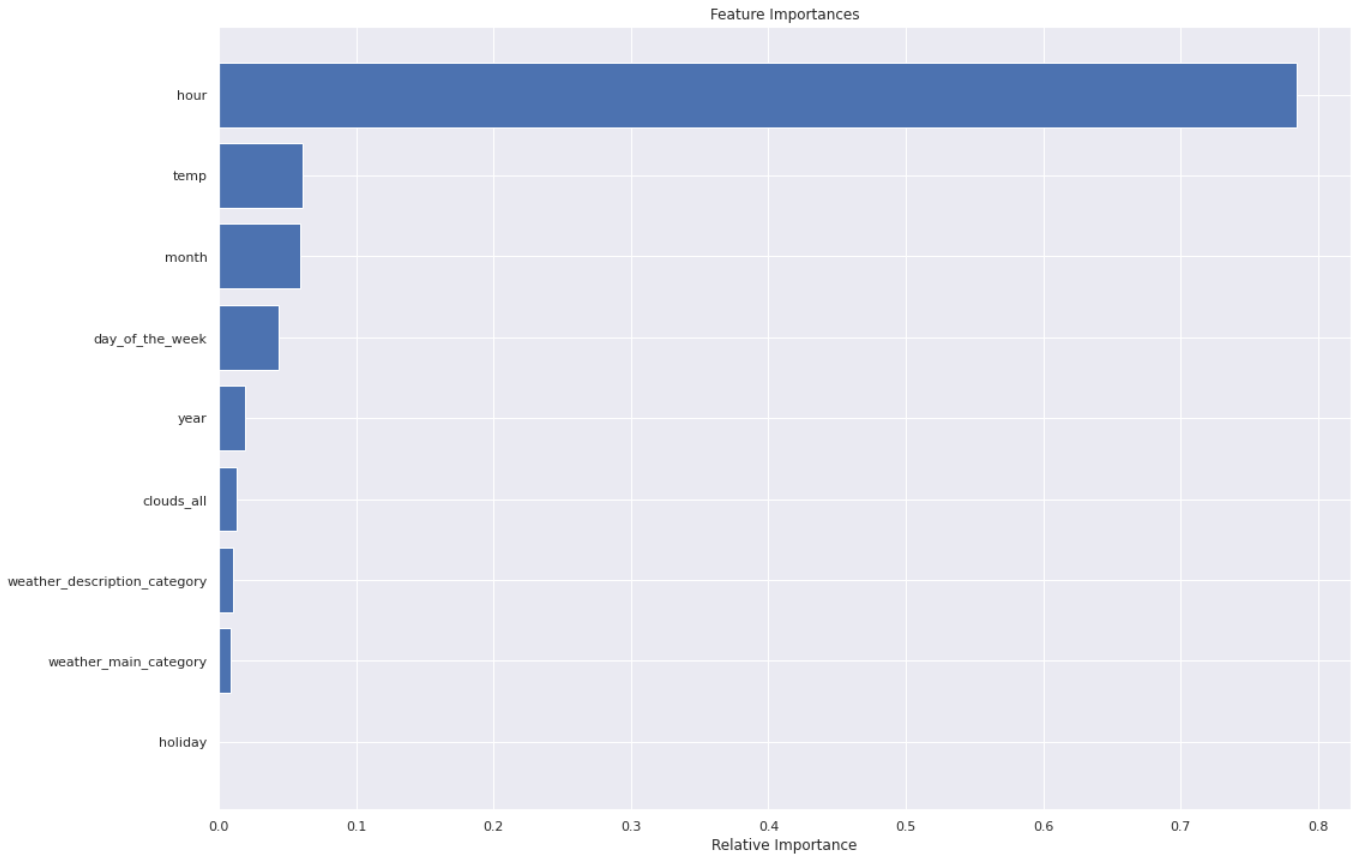

Figure 22:
Feature importances calculated through a RandomForestRegressor

These results strongly confirms (as already predicted by the Person Matrix Correlation) the meaningfulness of the hour attribute and assigns a relevant importance also to the temperature, month and day of the week.

Instead, weather related attributes are considered not very relevant. Also the holiday attribute is not relevant, but we have seen that it is very imbalanced (almost every day is not an holiday).

3.3.3 Results

Once verified the correlation between attributes and the feature importances, it is possible to draw some conclusions:

- The hour, temperature, month, day of the week and year attributes are relevant and necessary for make good predictions, so these features will be kept in the dataset.
- The clouds percentage and weather categorical descriptions are not so relevant, but they bring their contribution. These features will be kept in the dataset.
- The rain and snow attributes are nor correlated neither important. Their distribution is highly skewed and their values are quite inconsistent, so these features will be dropped from the dataset.

3.4 Data standardization

Standardize the data is an useful processing to help machine learning algorithms to learn from the data and also to manage the data in a easier way.

With standardization, it is intended centering the mean of each features around 0 with variance equal to 1.

Mathematically, for each feature and sample, the StandardScaler (from scikit-learn package) will compute:

$$z = \frac{x - \mu}{\sigma} \tag{4}$$

where

- μ is the mean of the training samples
- σ is the standard deviation of the training samples

Please notice that μ and σ are learnt from the training samples and the standardization is just applied on the testing samples.

3.5 PCA

PCA - formally, Principal Component Analysis - is a widely used dimensionality reduction technique, which detects the principal components (the directions that maximizes the variance of the projected data) and uses them to perform a change of basis.

Once computed the principal components (orthogonal one to the other), they are sorted from the one that encodes the most of the variance to the lowest. Notice that since every component is orthogonal to the others, they are all uncorrelated.

When evaluating the PCA, a common rule is to keep a number of components (from the PCA) that explains at least the 85% (for more conservative choices, the 90%) of the variance. In order to do so, we can take a look at the trend of the cumulative explained variance (Figure 23):

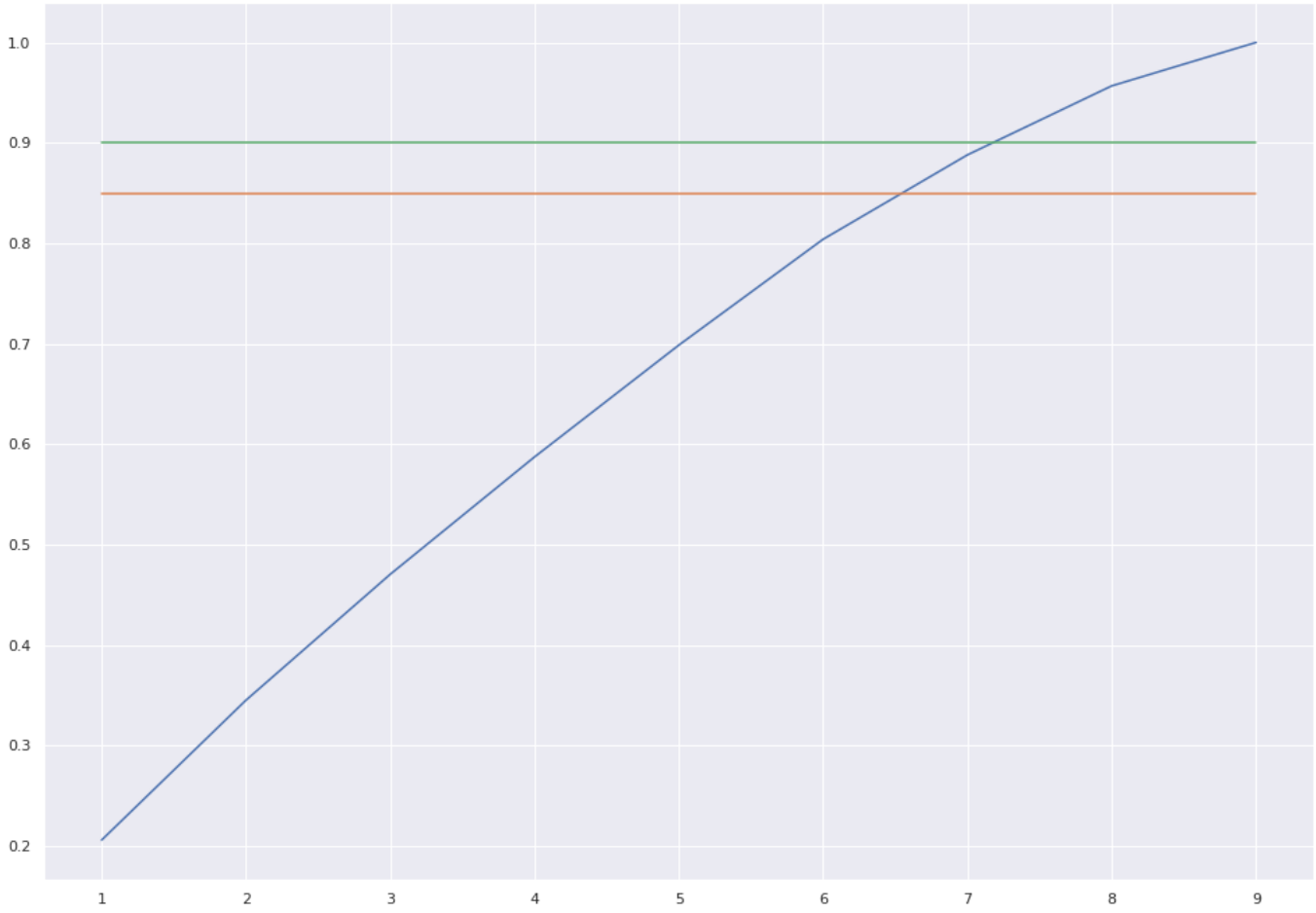

Figure 23:
PCA: Cumulative Explained Variance trend

As we can see, in order to explain the 90% of the variance, we should select 8 components. For this reason, since we have 9 features, it is not very useful to apply the PCA to this dataset.

4 Model Selection and Tuning

In the previous section we applied several common strategies to process our data and we understood which features are more relevant and useful in order to predict the hourly traffic volume, which is our goal.

In this section, we will

- Analyze the metrics used for evaluation
- Inspect the cross-validation technique applied
- Compare the results of 5 different regressors

4.1 Metrics

In order to evaluate the performances of the regressor taken in consideration in this project, it was used 2 metrics:

- MSE (Mean Squared Error)
- R^2 Coefficient of Determination

4.1.1 Mean Squared Error

The Mean Squared Error assesses the quality of a predictor by measuring the average of the squares of the errors.

The errors are defined as

$$y_i - \hat{y_i} \tag{5}$$

where

- y_i is the target value
- $\hat{y_i}$ is the predicted value

for the i^{th} set of features $\mathbf{x}_i$.

From eq. 5, we can square those errors

$$(y_i - \hat{y_i})^2 \tag{6}$$

and therefore calculate the MSE for all the predicted values as

$$MSE = \frac{1}{N}\sum_i (y_i - \hat{y_i})^2 \tag{7}$$

The Mean Squared Error is always positive, and since it is a risk function, the goal is to minimize it (which corresponds to minimizing the prediction errors).

4.1.2 Coefficient of determination: $\mathbf{R}^2$

The Coefficient of Determination (also known as R^2 score) is the proportion of the variation in the dependent variable $\mathbf{y}$ that is predictable from the independent variable(s) $\mathbf{X}$.

We define the residuals as

$$e_i = y_i - \hat{y}_i \tag{8}$$

where

- y_i is the target value
- $\hat{y}_i$ is the predicted value

for the i^{th} set of features $\mathbf{x}_i$.

Moreover,

$$\bar{y} = \frac{1}{N}\sum_{i}^{N} y_i \tag{9}$$

is the sample mean of $\mathbf{y}$.

From eq. 8 it is possible to calculate the residual sum of squares SS_{res}:

$$SS_{res} = \sum_i (e_i)^2 \tag{10}$$

and from eq. 9 it is possible to calculate the total sum of squares SS_{tot}:

$$SS_{tot} = \sum_i (y_i - \bar{y})^2 \tag{11}$$

Finally, the coefficient of determination R^2 is defined as

$$R^2 = 1 - \frac{SS_{res}}{SS_{tot}} \tag{12}$$

which has maximum value equal to 1, when $SS_{res} = 0$, i.e. the fitted model makes perfect predictions of the target values $y_i, \forall i$.

4.2 K-Fold cross validation

The K-Fold cross validation technique is a procedure to evaluate a model by testing it with different portions of the data.

Operatively,

1. The dataset is divided into K folds.
2. $K-1$ folds are used to train the model and 1 fold is used to evaluate the model with the previously introduced metrics.
3. The process is repeated K times, until all the folds are used for testing purposes.

For this project, I choose a number of folds $K = 5$ and I evaluated 5 different models (Figures 24 25 26 27 28, Table 9):

Regressor	**Average testing coefficient of determination R^2**
SVR	0.18
KNearestNeighborsRegressor	0.63
Quadratic Regression	0.64
DecisionTreeRegressor	0.69
RandomForestRegressor	0.83

Table 9:
K-Fold cross validation results.

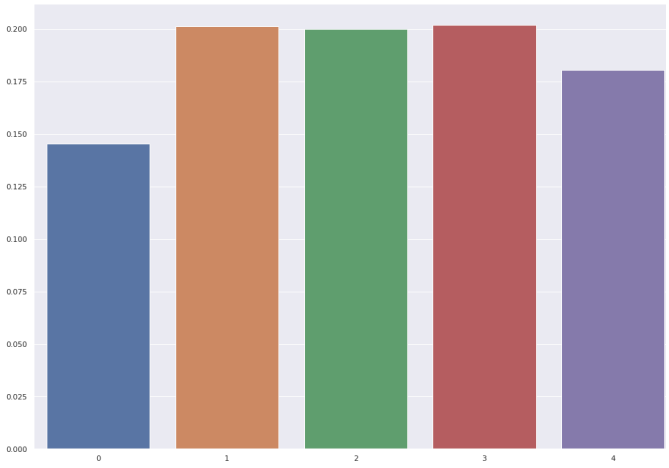

Figure 24: SVR model - K-Fold results.

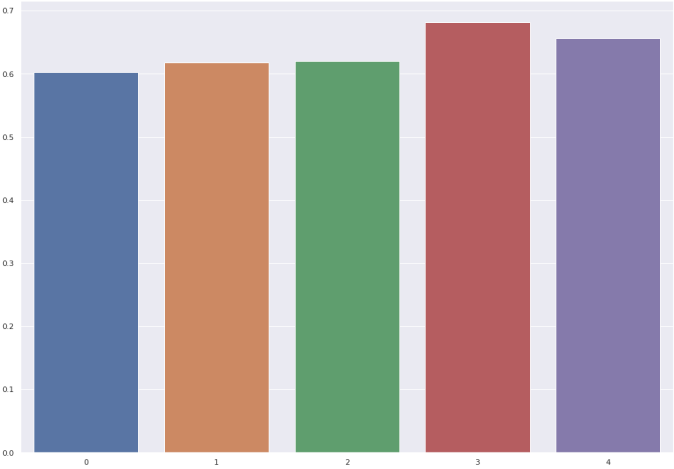

Figure 25: KNearestNeighbors model - K-Fold results.

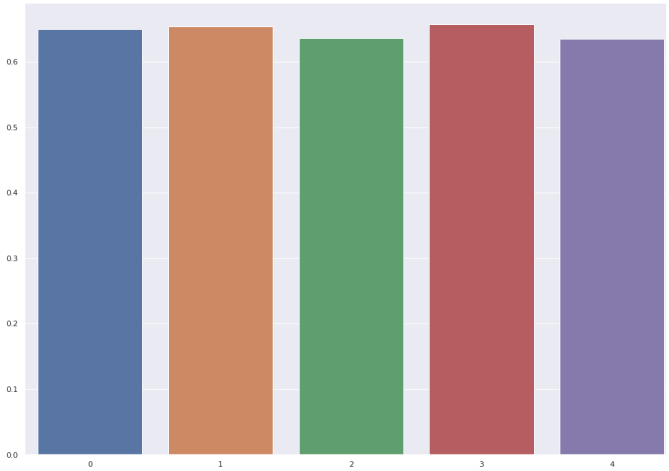

Figure 26: Quadratic Regression model - K-Fold results.

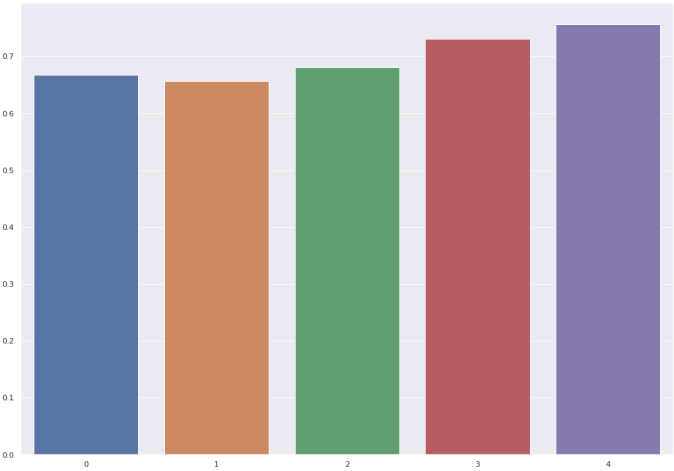

Figure 27: Decision Tree model - K-Fold results.

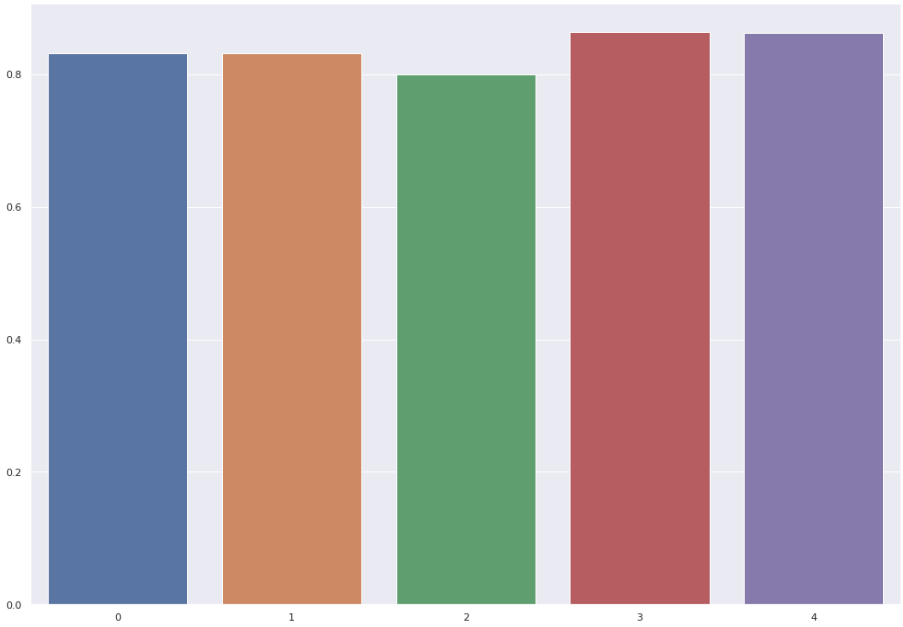

Figure 28: RandomForest Regression model - K-Fold results.

4.3 SVR

Support Vector Machines are widely used for classification problems (especially binary classification problems). The core idea of Support Vector Machines is that they map the training data maximizing the gap between a data point belonging to a class label and the other(s). In order to classify the data, it is chosen the hyperplane that maximizes the distance from it to the nearest data point on each side.

Support Vector Regressors uses the same principles as the SVM for classification: to minimize error, individualizing the hyperplane which maximizes the margin, keeping in mind that part of the error is tolerated (Figure 29).

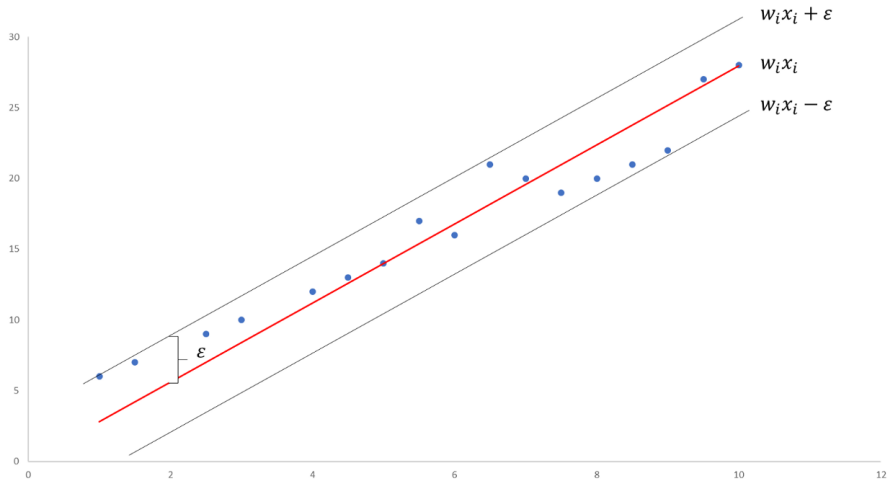

Figure 29: Illustrative example of Linear SVR.

This is different with respect to other models that try to minimize between target and predicted value: instead, SVRs tries to fit the best hyperplane within a threshold value ϵ.
As with Support Vector Machines, with Support Vector Regressors we can address also non-linear problems by means of the kernel trick, through which the data is re-mapped (through a non-linear mapping function) into an higher multidimensional space (Figure 30).

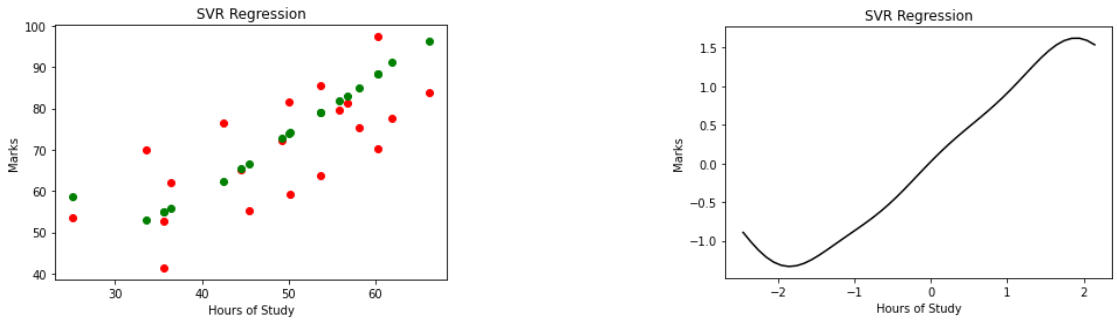

Figure 30: Illustrative example of not linear SVR.

4.4 KNearestNeighborsRegressor

The K-Nearest Neighbors algorithm is a classification / regression method that consists in predicting a class / a value by analyzing the k closest training examples in the dataset: in particular, in a regression environment, the output is the average of the values of the k nearest neighbors.

4.5 Polynomial Regression

The polynomial regression model is mathematically defined as

$$\begin{bmatrix} y_1 \\ y_2 \\ y_3 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 1 & x_1 & x_1^2 & \dots & x_1^m \\ 1 & x_2 & x_2^2 & \dots & x_2^m \\ 1 & x_3 & x_3^2 & \dots & x_3^m \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & x_n & x_n^2 & \dots & x_n^m \end{bmatrix} \begin{bmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_m \end{bmatrix} + \begin{bmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \varepsilon_3 \\ \vdots \\ \varepsilon_n \end{bmatrix}, \tag{13}$$

in which the relationship between the independent variable $\mathbf{X}$ and the dependent variable $\mathbf{y}$ is modelled as an n^{th} degree polynomial in $\mathbf{X}$.

In order to estimate the values of the coefficients $\boldsymbol{\beta}$ it is possible to apply the least square estimates method (normality is assumed for $\boldsymbol{\epsilon}$), by which an estimator for $\boldsymbol{\beta}$ can be calculated with the formula

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}'\mathbf{X})^{-1}\mathbf{X}'\mathbf{y} \tag{14}$$

In order to use the polynomial regression model to make predictions:

1. Divide the dataset into training and testing sets.
2. Estimate the value of $\hat{\boldsymbol{\beta}}$ through an estimation method
3. Once constructed the model, fed the testing data into the model and evaluate the predicted $\hat{y}$.

4.6 Decision Tree Regressor

The Decision Tree (Figure 31) is a commonly used supervised learning algorithm that consists in splitting the space of the solutions orthogonally, by making decision on the features, trying to find cut points using measures such as the Gini index or the Cross Entropy.
This model is - as the name says - as a tree, in which three types of nodes are present:

- Root node: the head node, represents the whole sample
- Interior node: represent the features of a data set. The branches deriving from that node represent the so-called decision rules.
- Leaf node: represent the output.

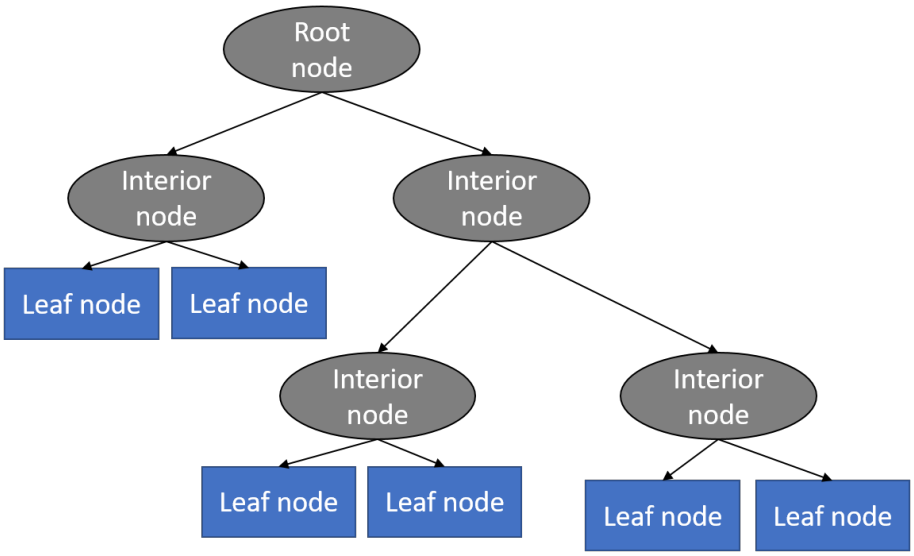

Figure 31: Illustrative example of a Decision Tree.

In this setting, a particular data point is fed through the tree: when the leaf node is reached, the prediction is made averaging the value of the dependent variables in that particular leaf.

4.7 Random Forest Regressor

Decision Trees are known for their simplicity and their easiness. However, they are likely to overfit. In order to overcome this problem, Random Forest Algorithms were introduced: they are still tree-based algorithms, but instead of using a single Decision Tree, they create a Forest of Decision Trees.

The Random Forest, essentially, produces a forest of Decision Trees (randomly created) and entrust all those trees to make predictions.

Those predictions are then merged together:

- in a classification environment, the class is selected through the mechanism of Majority Voting.
- in the regression environment, the predicted value is assigned by averaging the predictions from all the individual regression trees.

5 Conclusions

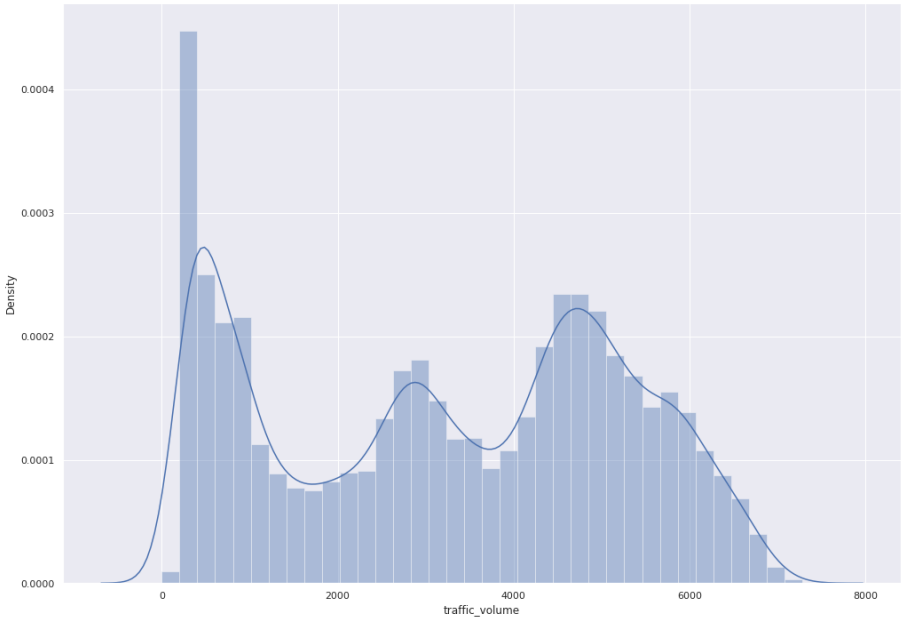

Figure 32: Traffic volume distribution plot.

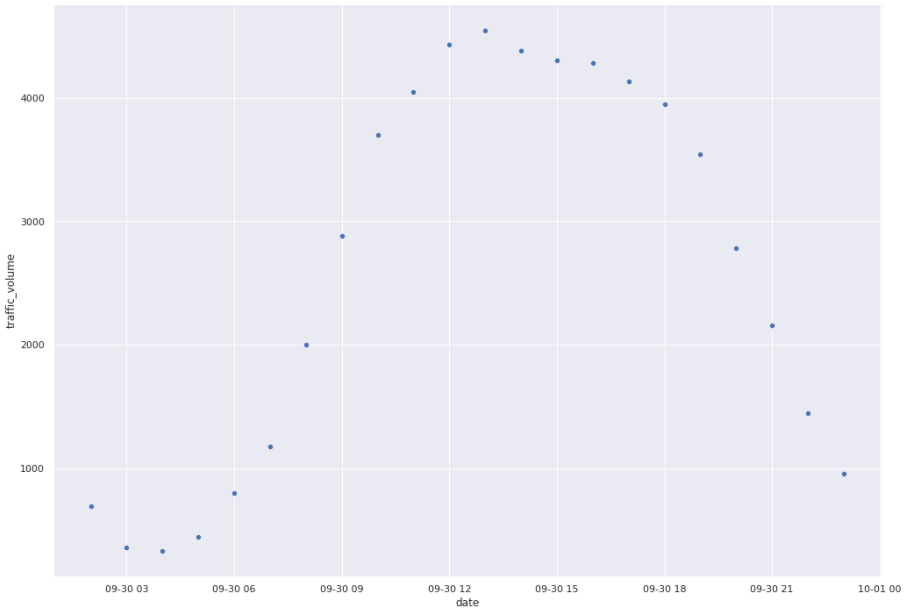

Figure 33: The trend of the traffic volume during a sampled day.

As shown in the K-Fold cross validation results for the models previously introduced (Figures 24 25 26 27 28, Table 9), we have seen that the Random Forest Regressor outperformed every other model.

In particular, the SVR (trained with linear and rbf kernels) did not work because the target variable trend was not approximable nor with a linear function neither with a non-linear exponential mapping.

In fact, as you can see from the distribution plot of the target variable (Figure 32) and in particular from the trend of the last day of observations (Figure 33), it follows the trend of a 2^{th}, 3^{th} polynomial degree function.

Due to this reason, the Quadratic Regression reached higher R^2 scores with respect to the SVR.

For what concerns the KNearestNeighborsRegressor: it reached quite good performances, but it is still a too simple algorithm. In fact, it just averages the values of the target attribute for the k most similar training examples, but does not dig into the data finding more complex relationships.

The DecisionTreeRegressor by itself immediately seemed promising - with respect to the other regressors - reaching 0.69 R^2 score on average on the testing set. However, it suffered from high overfitting: in fact, in the training set, it reached 1.00 R^2 score.

Since the Random Forest Regressor is well known for solving the overfitting problem of a single Decision Tree, I tried implementing it. It resulted to perform way better than a single Decision Tree, reaching very good results during the K-Fold cross validation.

For this reason, this model was then trained on the 70% of the dataset and tested on the remaining 30%, reaching accuracies (on the testing set) of 0.9 R^2 score, which is a very good result.