Machine Learning Engineer Nanodegree - Capstone Proposal

Title: Downlink Data traffic prediction for new 4G cell deployments

Mattia Bonazzi
May 22nd, 2019

Table of Contents

I. Definition

Project Overview

In this paper I will use Machine Learning algorithms to try to predict how the amount of Downlink Data traffic, processes by an antenna, evolves after the installation of 4G technology.
Telecommunication sector is a multibillion-dollar industry, always evolving and with many players participating in it. In such a competitive environment, being able to offer the best solution to your

clients is mandatory to survive. To do that, Network operators need to continuously invest millions of dollars in new technologies to offer the most advanced solutions to their customers.

A network operator has thousands of towers (I will refer to them also as sites) placed across a State territory. In each of these towers, there are installed some cells from which is broadcasted the phone signal. Each tower covers a limited area, giving access to data connection to the clients inside that area.
With the rise of new technologies, Operators need to upgrade these cells, to offer 4G connectivity (and soon 5G as well) to their clients. Upgrading a cell is a high cost investment and network need to decide where to deploy their upgrades.

Problem Statement

In this project I will try to predict how the installation of a 4G cell in a certain site, will affect the amount of data traffic consumed by the users in that site and processed by the mentioned antenna. Being able to predict that could help company to answer some questions, like:

- Which are the sites that are going to increase their data traffic the most after the 4G installation? In which sites should I prioritize the installation of 4G?
- How is the 3G downlink data traffic affected by the addition of 4G? Is the 4G inclusion generating an increase on the data demand or does the demand remain constant

To show the power of Machine Learning techniques, I will try to predict the average downlink data traffic processed in an antenna in the first week after the upgrade of the cells to 4G technology, knowing the daily traffic elaborated by the same antenna in the 35 days before the installation.

Metrics

In this paper we will face a Regressor problem, hence to evaluate the performance of my Supervised Learning algorithms I will use the two following metrics:

1. The Coefficient of Determination, that is defined as:

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

   Where:
   - A score of 1.0 means that the prediction is exactly the real value
   - A score of 0.0 means that the prediction is the mean value of the Y vector
   - The score could be negative if the prediction is farer from the real value than the mean of the Y vector.

2. The Mean Square Error, defined as:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

   It measures the average squared error of our predictions. For each point, it calculates square difference between the predictions and the target and then average those values. The higher this value, the worse the model is. It is never negative, since we’re squaring the individual prediction-wise errors before summing them but would be zero for a perfect model.

II. Analysis

Data exploration

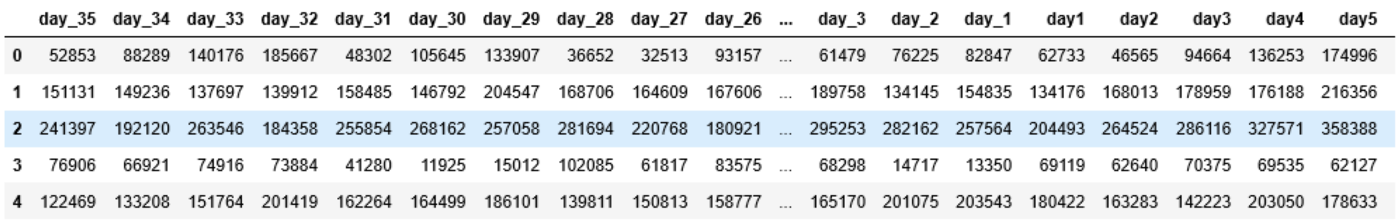
The database is made up of records of daily data traffic collected in each tower for a Spanish operator.
It's a database, with a record for each day for each site. Each row of the dataset is a different site and each column represent the total downlink data traffic processed by that antenna in that day. We recorded on a daily base the total downlink data traffic, in Mbit, processed by a cell in a day.
The data have been collected over a three years period, during which some of the cell have been upgraded from 3 G to 4G. Analysing this data we are able to see what effect had the upgrade of the cell from 3G to 4G on the amount of data traffic handled by the cell.

The data are inside a single CSV file, structured as follow:

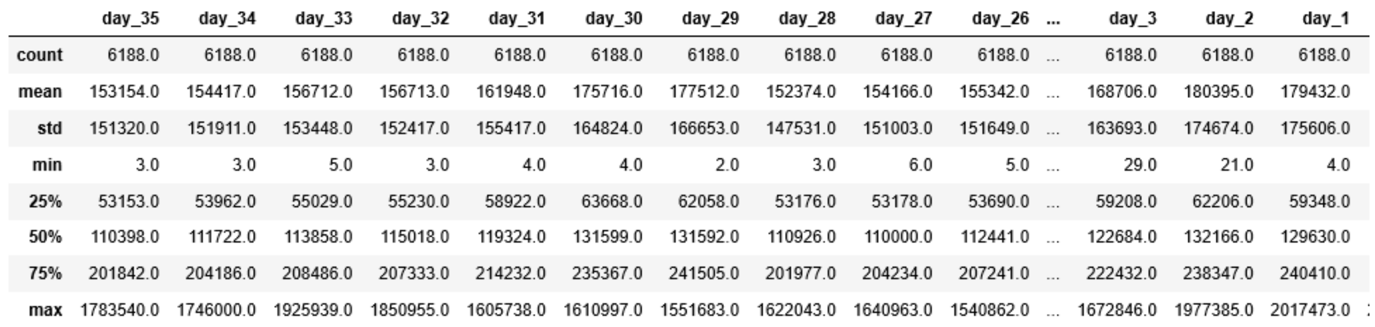
- *day_35*: Downlink data traffic processed by the cell on the 35th day before the 4G installation
- *day_34*: Downlink data traffic processed by the cell on the 34th day before the 4G installation
- *day_33*: Downlink data traffic processed by the cell on the 33th day before the 4G installation
  …
- *day_2*: Downlink data traffic processed by the cell on the 2nd day before the 4G installation
- *day_1*: Downlink data traffic processed by the cell on the 1st day before the 4G installation
- *day1*: Downlink data traffic processed by the cell on the 1st day after the 4G installation
- *day2*: Downlink data traffic processed by the cell on the 2nd day after the 4G installation
  …
- *day6*: Downlink data traffic processed by the cell on the 6th day after the 4G installation
- *day7*: Downlink data traffic processed by the cell on the 7th day after the 4G installation

Exploratory visualization

A partial snipped of the cleaned data can be seen below, along with the results of dataframe.describe method which gives us some additional meta data.

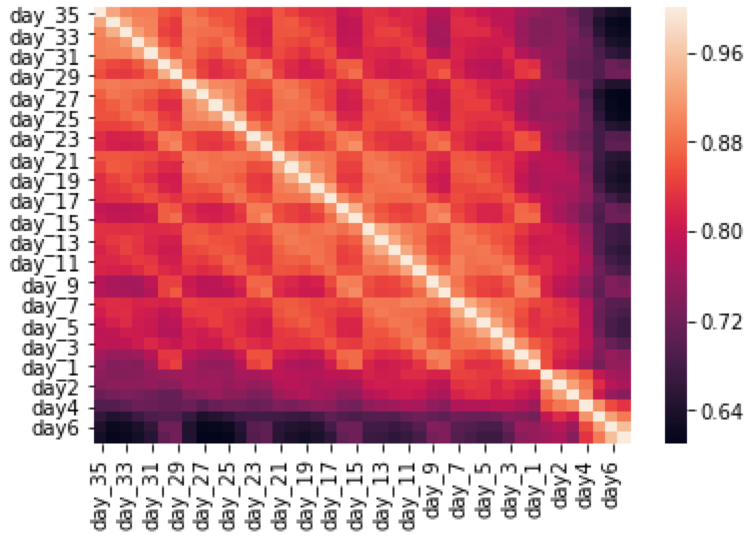


	day_35	day_34	day_33	day_32	day_31	day_30	day_29	day_28	day_27	day_26	...	day_3	day_2	day_1	day1	day2	day3	day4	day5
0	52853	88289	140176	185667	48302	105645	133907	36652	32513	93157	...	61479	76225	82847	62733	46565	94664	136253	174996
1	151131	149236	137697	139912	158485	146792	204547	168706	164609	167606	...	189758	134145	154835	134176	168013	178959	176188	216356
2	241397	192120	263546	184358	255854	268162	257058	281694	220768	180921	...	295253	282162	257564	204493	264524	286116	327571	358388
3	76906	66921	74916	73884	41280	11925	15012	102085	61817	83575	...	68298	14717	13350	69119	62640	70375	69535	62127
4	122469	133208	151764	201419	162264	164499	186101	139811	150813	158777	...	165170	201075	203543	180422	163283	142223	203050	178633



	day_35	day_34	day_33	day_32	day_31	day_30	day_29	day_28	day_27	day_26	...	day_3	day_2	day_1
count	6188.0	6188.0	6188.0	6188.0	6188.0	6188.0	6188.0	6188.0	6188.0	6188.0	...	6188.0	6188.0	6188.0
mean	153154.0	154417.0	156712.0	156713.0	161948.0	175716.0	177512.0	152374.0	154166.0	155342.0	...	168706.0	180395.0	179432.0
std	151320.0	151911.0	153448.0	152417.0	155417.0	164824.0	166653.0	147531.0	151003.0	151649.0	...	163693.0	174674.0	175606.0
min	3.0	3.0	5.0	3.0	4.0	4.0	2.0	3.0	6.0	5.0	...	29.0	21.0	4.0
25%	53153.0	53962.0	55029.0	55230.0	58922.0	63668.0	62058.0	53176.0	53178.0	53690.0	...	59208.0	62206.0	59348.0
50%	110398.0	111722.0	113858.0	115018.0	119324.0	131599.0	131592.0	110926.0	110000.0	112441.0	...	122684.0	132166.0	129630.0
75%	201842.0	204186.0	208486.0	207333.0	214232.0	235367.0	241505.0	201977.0	204234.0	207241.0	...	222432.0	238347.0	240410.0
max	1783540.0	1746000.0	1925939.0	1850955.0	1605738.0	1610997.0	1551683.0	1622043.0	1640963.0	1540862.0	...	1672846.0	1977385.0	2017473.0

The database is composed of an ID column to identify the sites, one column for each of the 35 days before the installation, and one column for each of the 7 days right after the installation.
It's important to specify that the data has already been cleaned (before being saved in the CSV file); I have already eliminated the sites with missing records in the 35 days before the installation and in the 7 days that follow it.



From the correlation graph above we can see how each feature has a stronger correlation with the features immediately more close to it.

I have then split the data into features and target (or independent and dependent variables). I calculate the target variables as the average of the data traffic in the 7 days following the installation the 4G

	day_35	day_34	day_33	day_32	day_31	day_30	day_29	day_28	day_27	day_26	...	day_10	day_9	day_8	day_7	day_6	day_5	day_4	day_3
0	52853	88289	140176	185667	48302	105645	133907	36652	32513	93157	...	71548	70187	74519	84851	56083	50241	58172	61479
1	151131	149236	137697	139912	158485	146792	204547	168706	164609	167606	...	201630	212717	153572	164675	184146	171135	192642	189758
2	241397	192120	263546	184358	255854	268162	257058	281694	220768	180921	...	281263	301006	273402	309820	251000	247043	275858	295253
3	76906	66921	74916	73884	41280	11925	15012	102085	61817	83575	...	58438	15544	12709	68753	72453	58311	72382	68298
4	122469	133208	151764	201419	162264	164499	186101	139811	150813	158777	...	165427	173540	205815	144289	175807	185027	165408	165170

Features

	WeekAvg
0	115078.0
1	171005.0
2	314738.0
3	53255.0
4	173980.0

Targets

Algorithms and Techniques

Since I will try to predict the value of Average Data Traffic, I will need to use a Supervised Regressor algorithm. I will test the three-following algorithm:

1. Linear Regression model: it is a simple method. It is very easy and intuitive to use and understand. It assumes there is a straight-line relationship between dependent and independent variables. It has the risk of it oversimplifies real world problems, but it works in most of the cases. Because of the simplicity of our problem, this model could potentially give us good results
2. Random Forest Regression model: Random forest is an ensemble method, in which a regressor is constructed by combining several different Independent base regressors. The independence is theoretically enforced by training each base regressor on a training set sampled with replacement from the original training set. This technique is known as bagging, or bootstrap aggregation. In Random Forest, further randomness is introduced by identifying the best split feature from a random subset of available features. The ensemble regressor then aggregates the individual predictions to combine into a final prediction. Introduction of randomness helps achieve independence to a certain degree and it has been empirically observed that ensembles perform significantly well over individual base classifiers
   an ensemble technique, thanks to its simplicity it adapts good to different problems
3. Support Vector Regression model: it is characterized by the use of kernels, sparse solution, and VC control of the margin and the number of support vectors. SVR has been proven to be an effective tool in real-value function estimation. As a supervised-learning approach, SVR trains using a symmetrical loss function, which equally penalizes high and low misestimates. A flexible tube of minimal radius is formed symmetrically around the estimated function, such that the absolute values of errors less than a certain threshold are ignored both above and below the estimate. In this manner, points outside the tube are penalized, but those within the tube, either above or below the function, receive no penalty. One of the main advantages of SVR is that its computational complexity does not depend on the dimensionality of the input space.

Benchmark

As benchmark we can use a model that simply assume that the data volume will increase by the 50% in the first week after the installation. The benchmark model calculates the average data traffic in the 7 days before the installation and use that value increased by 50% as target result.

The benchmark model gives us the following score values:

	Coeff. of Determination (R^2)	**Mean Square Error (MSE)**
Naïve Model	0.518	1.72e+10

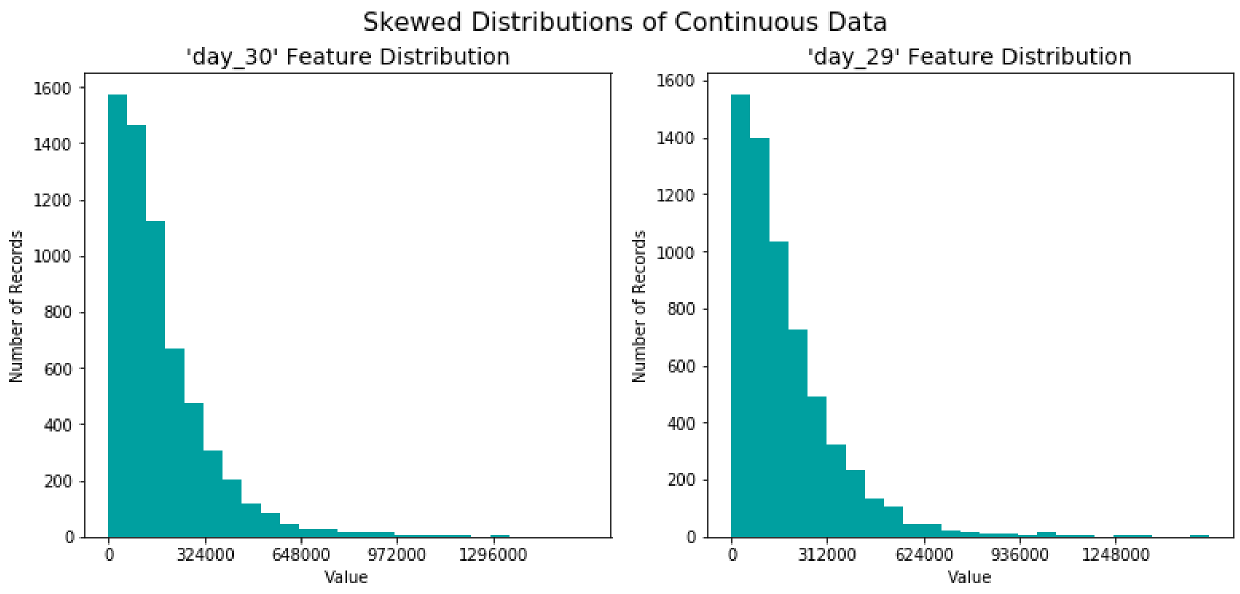
III. Methodology

Data Preprocessing

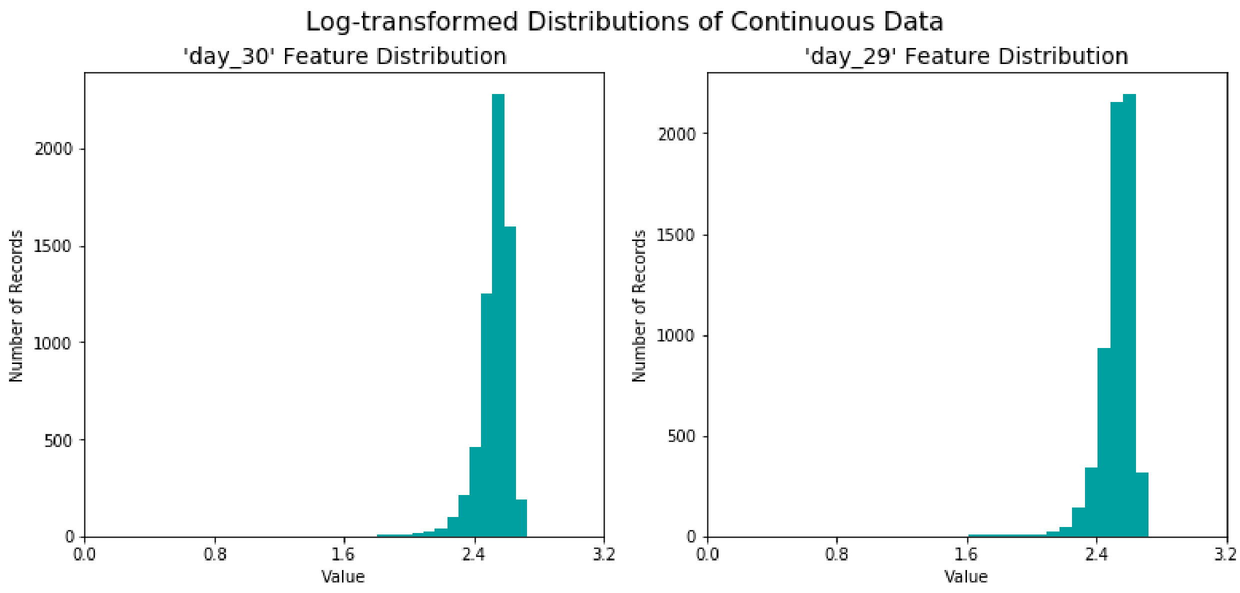
As mentioned above, for this dataset, there are no invalid or missing entries we must deal with (since I already removed it before saving the data in the CSV file), however, we can still work on some qualities about the data that can help with the outcome of the predictive algorithms.

As we can see from the graphs below, the values of the first two features tend to lie near the left side of the graph, but we also have a non-trivial number of vastly larger values that can negative affect the outcome of the algorithms. The same happen for the remaining features variables.

For highly-skewed feature distributions it is common practice to apply a logarithmic transformation on the data so that the very large and very small values do not negatively affect the performance of a learning algorithm. Using a logarithmic transformation significantly reduces the range of values caused by outliers.



Below are showed the data after performing the logarithmic transformation. The range of values has reduced, and the data are distributed differently.



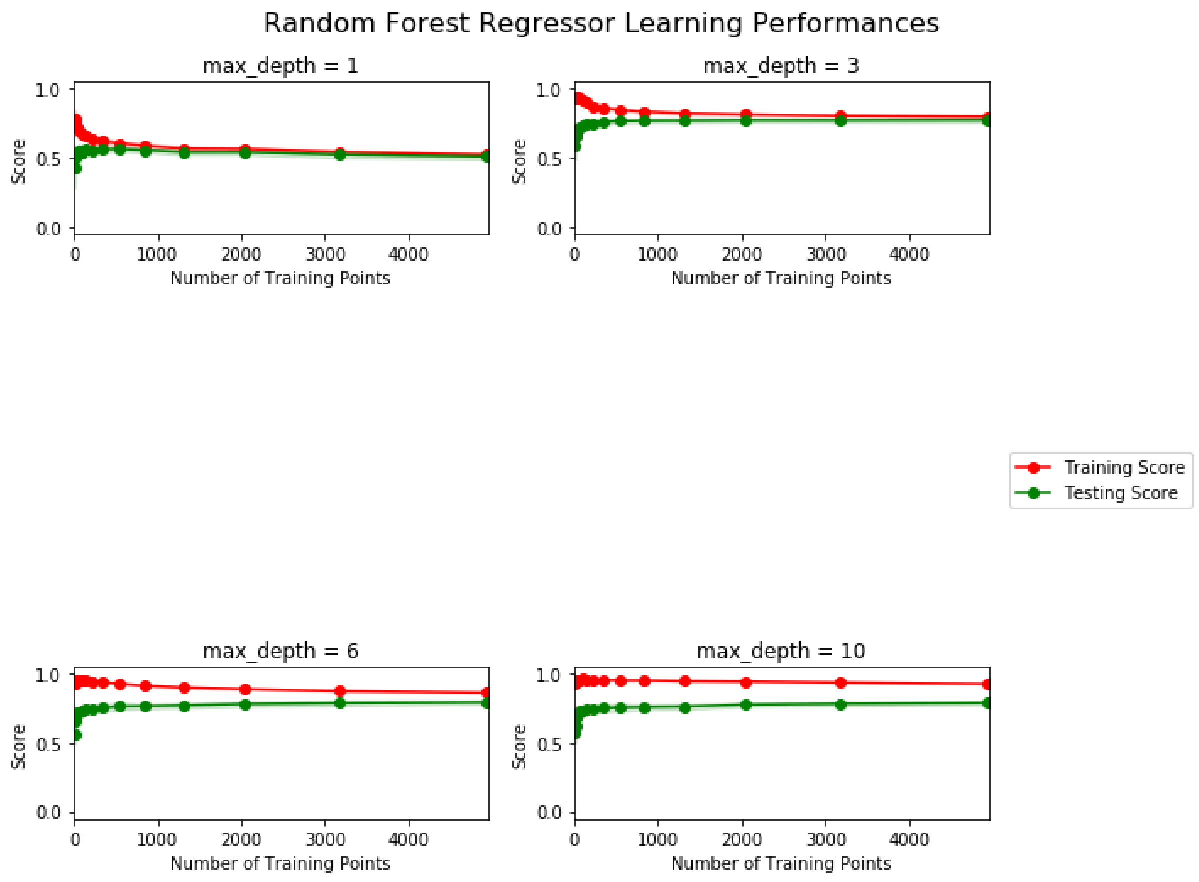
In addition to performing logarithmic transformations on features that are highly skewed, it is often good practice to perform some type of scaling on numerical features. Since applying a scaling to the data does not change the shape of each feature; however, normalization ensures that each feature is treated equally when applying supervised learners.
I have so decided to apply a normalization on all the features and normalize all the values in the range between 0 and 1.

Implementation

The first model I will test is a Random Forest Regressor. We have already spitted the data in Independent (35 days of traffic before installation) and Dependent (one-week average traffic in the 7 days after the installation) variables, we can use train_test_split from sklearn to divide the dataset into training and testing subsets. I choose an 80/20 percent split fiving me a training set consisting of 4950 records and a testing set of consisting of 1238 records.

To better understand the behaviour of our model and how it fits our model, I will first plot some graphs to observe how the model complexity and dataset size affects performance.

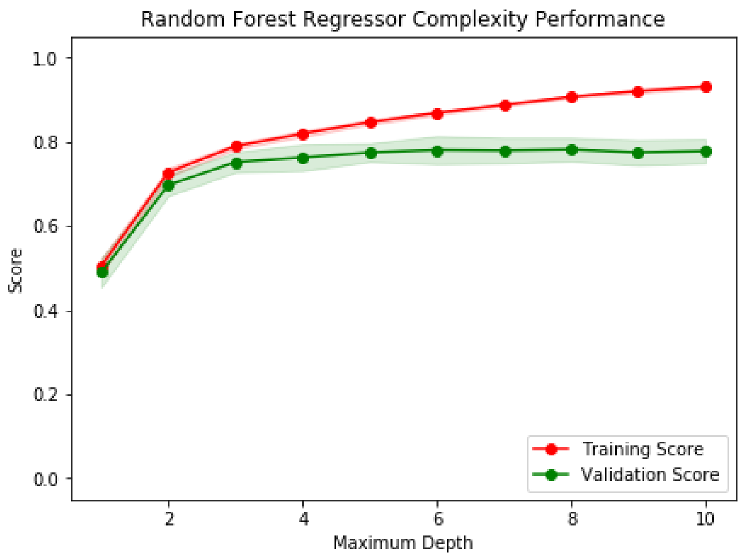
In the four graphs below, you can see the performance of a Random Forest Regressor model to vary maximum depths values and with increasing training point. The model is scored on both the training and testing sets using R2, the coefficient of determination.



We can see that as the max_depth parameter increases so too does the training score. The Testing score instead start to decrease for values of max_depth too high. This indicates that for values of max_depth too high, the model incurs in the overfitting problem, as we can see from the two curves starting to being more separated.
From the above graphs, we can also deduct that having more training points would not benefit the model. The two curves tend to settle and stay stable at that same value as the training points increase.

In the graph below, I have plotted the two complexity curves, one for training and one for validation, resulting from the Random Forest Regressor model, trained and validated on the training data using

different maximum depths. The model is scored on both the training and validation sets using the R^2 score.
We can see from the graph how good the model fit the problem; passing from suffering of high bias with low values of max_depth, to suffer of high variance with high values of max_depth.
Later in the paper, once we have chosen the best model (among those that we will test) we will need to tune at best its parameters.



To properly evaluate the performance of the models we choose, it's important to create a training and predicting pipeline that allows you to quickly and effectively train models using various sizes of training data and perform predictions on the testing data.
I have created a function called train_predict to do this. It trains and tests a model on various sample sizes and store the results. We can call this function iteratively on many different models to evaluate their performance.

I chose to test three model:

1. Linear Regression model: because of the simplicity of the problem, this model could potentially give us good results
2. Random Forest Regression model: an ensemble technique, thanks to its simplicity it adapts good to different problems
3. Support Vector Regression model: it can solve linear and non-linear problems and work well for many practical problems

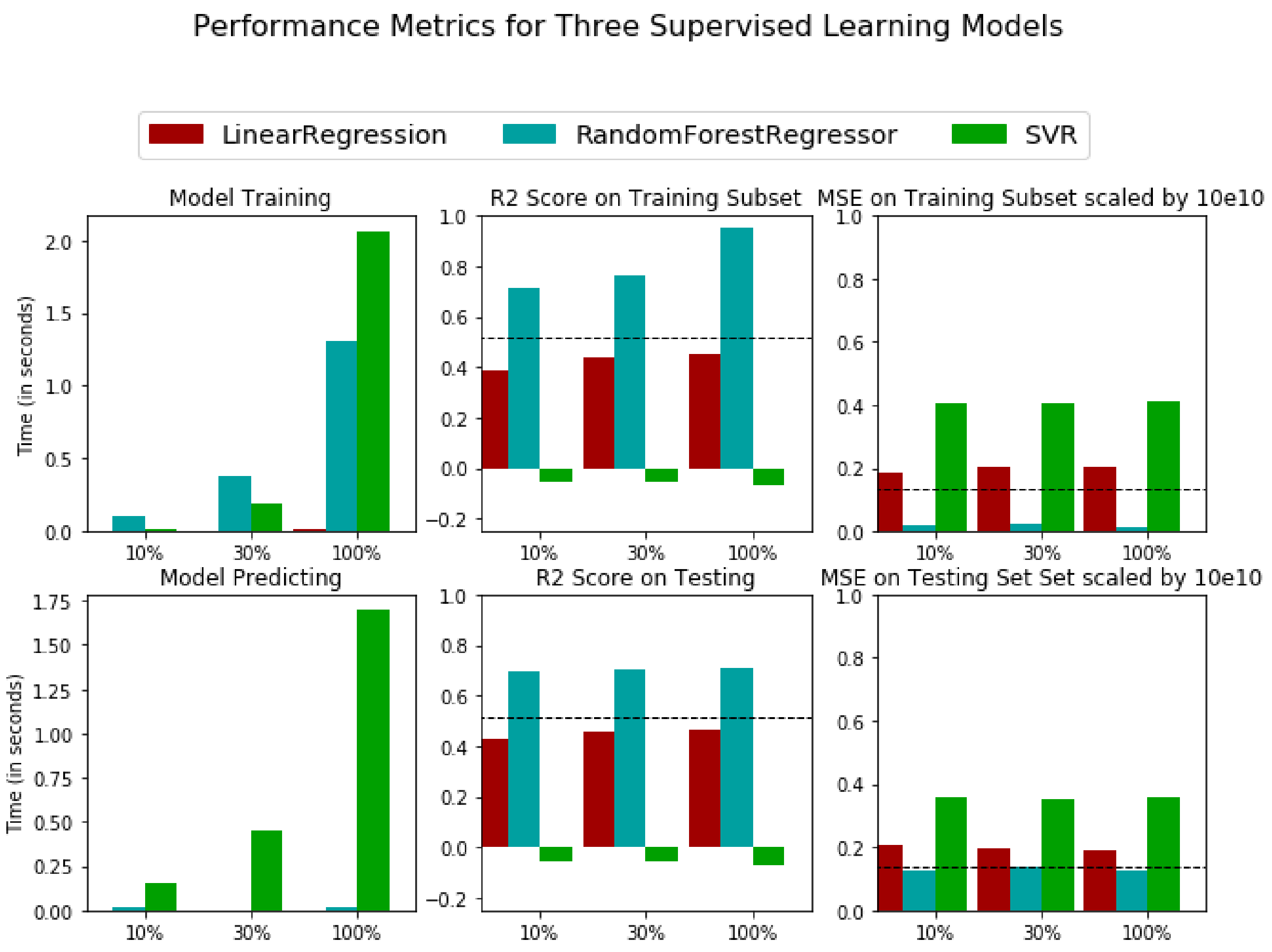
I ran the following models through the pipeline and plotted their results for R^2 score and Mean Square Error on 10%, 30% and 100% of sample training data.

The table below summarise the results of the tree models.

		R^2 score			**Mean Square Error**		
Size of the training set		**10%**	**30%**	**100%**	**10%**	**30%**	**100%**
Linear Regression	Train set	0.387	0. 439	0.450	1.84e+10	2.05e+10	2.03e+10
	Test set	0.427	0.456	0.464	2.08e+10	1.93e+10	1.92e+10
	Train set	0.770	0.761	0.952	1.61e+10	1.74e+10	2.06e+10

Random Forest	Test set	0.679	0.692	0.694	1.39e+10	1.2e+10	1.37e+10
SVR	Train set	-0.054	-0.053	-0.068	4.04e+10	4.04e+10	4.09e+10
	Test set	-0.057	-0.056	-0.071	3.55e+10	3.54e+10	3.59e+10

In the graph below, I plotted a graphic representation of the results. On the x-axe of all graph is plotted the size of the training set. Also, the MSE values have been scaled down of a factor of 10e10.



The SVR doesn't perform well at all, it gives us worst results even compared to the naïve model and it clear that it can't be used to solve our problem.
The Linear Regression model performs close to the benchmark model, it doesn't really add any value to it, since I can achieve similar results with the Naïve model.
The Random Forest Regressor model seems to be the one that gives us the best result, it achieve an higher R2 score than the Naïve model and it has a little lower SME value.

As stated above, Random forest is an ensemble method, in which a regressor is constructed by combining several different Independent base regressors. The independence is theoretically enforced by training each base regressor on a training set sampled with replacement from the original training set. This technique is known as bagging, or bootstrap aggregation. In Random Forest, further randomness is introduced by identifying the best split feature from a random subset of available features. The ensemble regressor then aggregates the individual predictions to combine into a final prediction. Introduction of randomness helps achieve independence to a certain degree and it has been empirically observed that ensembles perform significantly well over individual base classifiers.

Refinement

I can probably achieve even better performance by setting the hyperparameters of the model to different values, to make the model better fit the problem. I will use GridSearHCV to fine tune the parameters of the Random Forest Regressor model.

By running the model trough GridSearchCV to determine the optimal list of parameters we get the following results:

	Coeff. of Determination (R^2)	**Mean Square Error (MSE)**
Un-optimised Model	0.694	1.37e+10
Optimised Model	0.734	0.93e+10

We can see how the optimized model can achieve better performances.

IV. Results

Model Evaluation and Validation

Comparing the results of the optimized model and the Naïve model, we can see that we achieved reasonably good results. We have obtained a higher R2 score, so we have been able to extract more predicting information from the data. However, the MSE is just a little bit lower than the one achieved by the Naïve model.

	Coeff. of Determination (R^2)	**Mean Square Error (MSE)**
Optimised Model	0.734	0.93e+10
Benchmark	0.518	1.72e+10

Justification

We can also examine how robust is our model.
An optimal model is not necessarily a robust model. Sometimes, a model is either too complex or too simple to sufficiently generalize to new data. Sometimes, a model could use a learning algorithm that is not appropriate for the structure of the data given. Other times, the data itself could be too noisy or contain too few samples to allow a model to adequately capture the target variable.

I have so analysed the sensitivity by run the data through the optimization pipeline ten times with different training and testing sets to see how the prediction for a specific site changes with respect to the data it's trained on, and reported the results below:

Trial 1: Mbit 2,246,980
Trial 2: Mbit 2,227,971
Trial 3: Mbit 2,288,110

Trial 4: Mbit 1,699,806
Trial 5: Mbit 2,156,055
Trial 6: Mbit 2,189,784
Trial 7: Mbit 2,116,438
Trial 8: Mbit 2,254,026
Trial 9: Mbit 2,258,795
Trial 10: Mbit 1,452,530

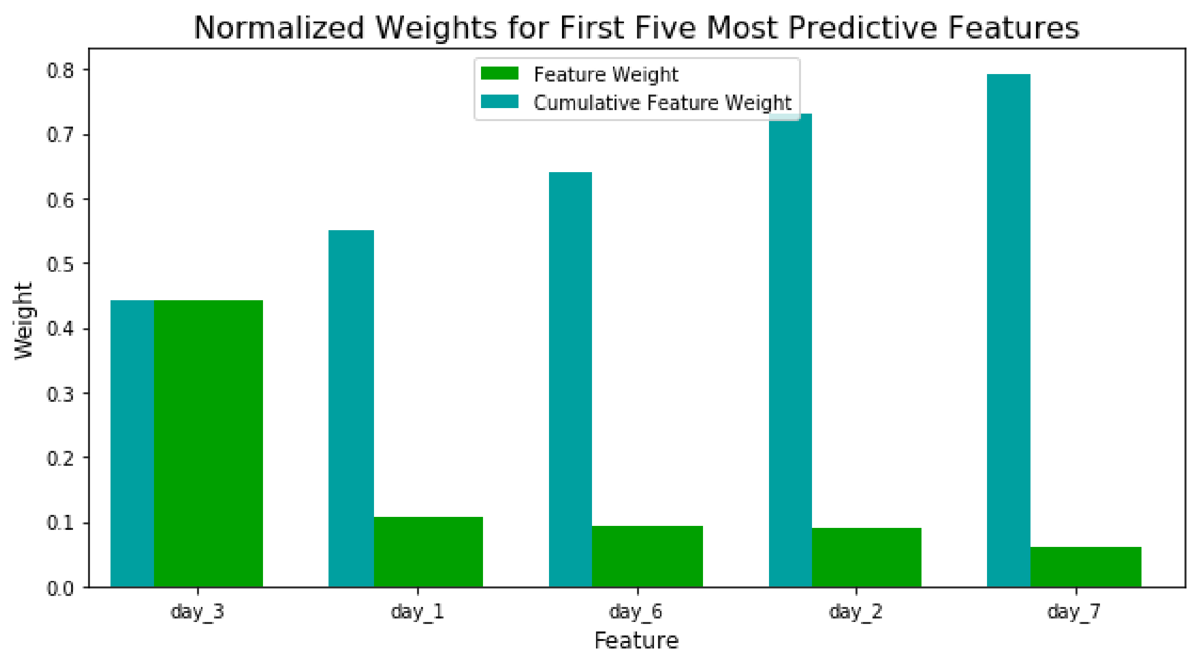
Range in Data Volume: Mbit 835,580

The model robustness seems acceptable.

V. Conclusion

Free-Form Visualization

As last analysis, we can use the feature_importances_ method of the Random Forest model to estimate which ones are the features that contribute the most to the prediction.



From the graph above its clear that the most important features are represented by the data traffic in the week just before the installation of the 4G antenna (day_3, day_1, day_6, day_2, day_7; that represent respectively the data traffic processed by the antenna in the 3rd,1st,6th,2nd and 7th day before the installation of the 4G cell). This is intuitive, since the days closer to event, are the ones that can give us more information about the evolution of the near future data traffic.

Reflection

Considering the R2 score and the MSE achieved from our best model, we have obtained reasonably good results, and the models tested seem to fit the problem.
The performances of our refined model are way above the benchmark model.
The model seems also to be quite robust, so we can feel satisfied with the results obtained.

I found it interesting that the Linear Regression model gives us results not really better than the Naïve model.

Improvement

As future development I could try enriching the dataset, including more past and future days, and:

- test other regression models with multiple target variables to try to predict the daily traffic of the following 30 days after the installation
- predict the average traffic in a further future (like the average data traffic in the month after the 4G installation, instead of in the 7 days)