POLITECNICO DI TORINO

Master's Degree in Computer Engineering

Master Thesis

Analysis of car sharing data and service availability prediction using data mining techniques

Assignees
prof.ssa CHIUSANO Silvia Anna
prof. CAGLIERO Luca
prof. GARZA Paolo

Candidate
Marco Delrio

Academic Year 2017-2018

Abstract

The topic of this thesis is the analysis of the car sharing service, in particular of the analysis of the availability of this kind of service in terms of space (i.e, different areas of a city) and time (i.e., different moment of time, days of the week, slots of time and so on). In order to do so I exploited several regression algorithms, that gave as their outputs the prediction of the number of cars available in a given area in a given moment of time. The analysis of the prediction error of these algorithms allowed to determine which are the situations, from both space and time point of view, in which the service is most used. Such analysis allowed also to determine which are the techniques that give best performances when predicting this kind of data. In particular three use cases are considered in this thesis:

- an italian city, for which I considered two different car sharing service providers;
- two USA cities, Seattle and Portland, for which I considered only one service provider.

The chapters of this thesis are structured as follows: chapters from 1 to 3 give an introduction of the topic faced in this thesis and an overview of the tools and the software I developed; chapters from 4 to 9 describe in details the analysis made considering the first use case, giving informations about the dataset, the performance of the used algorithms and the different granularities at which the analysis was made; chapters 10 and 11 describe instead respectively use case 2 and 3; the final chapter makes a point of the conclusions and results of all the analyses I made and shows which are some possible extensions and future implementations that can start from my work.

Contents

Chapter 1

Introduction

This chapter gives an overview of the thesis objective and what are the main topics of my work. It also contains informations about the car sharing service and what are its main issues and how they were resolved so far.

1.1 Thesis objectives and software development

The objective of this thesis is to develop a framework that is able to elaborate car sharing data and to give an evaluation of the accuracy of a regression algorithm that is run using such data as an input. All the regression algorithms I exploited were analyzed in terms of the error of their predictions and then compared between them to determine which regression technique is the best one to be used to resolve the problem of predicting the availability of the car sharing service in different areas of a given city. In my case predicting the availability of the service means predicting the number of cars that are made a disposal of the service customers in a certain area, at a certain moment of time. To reach the thesis objective it was necessary to use several tools and algorithms which are described in the following chapters.

1.2 Main issues when facing car sharing

The main issue when facing car sharing is the redistribution of the vehicles, in order to provide the highest number of available cars in areas where the highest demand is expected. Some vehicles can be rent to reach peripheral areas of the city, where the demand of vehicles is not significatively high. So it is necessary for this vehicles to be redistributed, in order to avoid keeping cars parked for long times before being rent again. This operation has an instrinsic cost for the service provider, that has to have available operators to do this job. Two operators are usually needed, one that drives the car to a different area and one that guides him (for example vocally). Characterizing the car sharing service is strictly necessary to achieve the best distribution of available cars in a city . Researches so far focused mainly on these features:

- characterization of car sharing users and usage patterns;

- spatial analysis, identifying the distribution of demand among different areas;
- temporal analysis, identifying the distribution of demand among different days and timeslots;
- external factors that can influence car sharing usage.

All these factors can be considered to achieve a better knowledge of where and when vehicles could be required in the future. Of course achieving the first point requires datasets that provide socio-demographical informations. These datasets can be very useful as they can give many informations (such as age and income) that can help identifying the characteristics of the most frequent users of the service. Knowing where people that share these characteristics live can provide informations about where the demand is expected to be high. Such datasets are often not available or their access is highly restricted, even if some researches associated to national projects managed to have access to such kind of data [1]. Spatial and temporal analysis instead can be done much easier. Datasets used to do such kind of analysis have informations such coordinates of the rent car (departure and arrival) and informations about the vehicle(fuel/battery level, vehicle ID, customer ID, interior conditions etc.). Some factors (e.g weather) can infuence the usage of the car sharing service and knowing the behaviour of these factors is helpful to predict how the rental can be potentially influenced (short-term or long-term) by them.

1.3 State of the art

Clustering is one of the most adopted technology that has been exploited so far to analyze and characterize the car sharing service. Clustering algorithms such as k-means or partitioning around medoids have been used at different granularities to identify areas and times with highest demand,travel and users features. Such analysis leads to informations about groups of days with similar usage patterns (from a spatial point of view), asymmetries in the spatio-temporal distribution of the vehicles availability and demand [2] or users behavioural classes [3] [4]. All these informations can be used to predict where demand will be present, taking into consideration the external factors mentioned above. For example classifying the different areas as business or residential can help to do a better analysis. Infact travels from a residential and a business area are expected to be more frequent in the morning, when workers use the car sharing service to reach their working places. Prediction models for car sharing usage are a field in continous evolution.

Also deep networks have been used to predict the distribution of demand by training them with spatial and temporal features of car sharing demand [5]. This means we can use for example the previous 7-day car sharing demand data to predict the demand in the eighth day by training the network properly.

Chapter 2

Used tools and algorithms

This chapter describes the tools I used to reach the thesis objective and the regression algorithms that were run using car sharing data as an input.

2.1 Eclipse IDE

The Eclipse IDE was used to write the Java code to implement the pre-processing of the datasets, in particular to:

- elaborate the input datasets in order to obtain new datasets with the information I needed to make a prediction about the availability (in terms of number of vehicles) of the car sharing service;
- compute error measures starting from the *RapidMiner* output files.

2.2 RapidMiner

RapidMiner and its library of algorithms were used to predict the number of vehicles present in a certain area and to generate the graphs showing the informations about the error measures starting from the error files given as an output of the prediction error computation part of the Java framework. In particular also some algorithms of the *Weka* extension of RapidMiner were used.

2.3 Regression algorithms

Several regression algorithms were applied to make predictions. Each one of the algorithms in some cases was subject of an optimization process, in order to tune its parameter in the best way and make it able to predict data more efficiently.

2.3.1 SVM overview

SVM is a tool that can be useful not only to solve regression problems but also to classify data and recognize data patterns [6]. Basically SVM is an algorithm that tries to find the

best way to separate data, in order have different spaces in which the input data share some features. Such separation is based on the concept of *hyperplane*, that can be considered a line able to separate the input variable space. The distance between the line and the closest data points is referred to as the *margin*. The best or optimal line that can separate the two classes is the line that has the largest margin. This is called the *Maximal-Margin hyperplane*. The hyperplane is learned from training data using an optimization procedure that maximizes the margin. The margin is calculated as the perpendicular distance from the line to only the closest points. Only these points are relevant in defining the line and in the construction of the classifier. These points are called the *support vectors*. In practice data usually can not be separated by an hyperplane, so the margin constraint is relaxed, allowing some points to violate the separating line. Such violation is measured usually referring to a threshold, that helps classifying it as allowable or not. Usually a classification or regression problem can not be solved in a two dimensional space, so the learning of the hyperplane problem is solved moving into an higher dimensional space using some linear algebra, in particular using *kernel functions*. Such functions are able to map the input data into another space and to solve the problem into another space. It is desirable to use more complex kernels as it allows lines to separate the classes that are curved or even more complex. The equation for making a prediction for a new input computes the dot product between the input (x) and each support vector (x_i) and is calculated as follows

$$f(x) = B_0 + sum(a_i * (x, x_i))$$

This equation involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_0 and a_i (for each input) must be estimated from the training data by the learning algorithm.

The kernel functions considered in this paper are:

- Linear: the dot product is rewritten as

$$K(x, y) = sum(x * y);$$

- Polynomial: instead of the dot product, a polynomial function is used, for example:

$$K(x, y) = 1 + sum(x * y)^d,$$

  where d is the degree of the function and is a parameter of the algorithm;

- RBF: a *radial basis function* is used as a kernel function, for example:

$$K(x, y) = exp(\frac{||x - y||^2}{2\sigma^2}),$$

  where σ is a parameter of the algorithm;

- Sigmoid: a sigmoid function is used as a kernel function, for example:

$$K(x, y) = tanh(\gamma * x^T y + r),$$

  where r and γ are kernel parameters.

2.3.2 W-SMOreg algorithm

The W-SMOreg algorithm is a regression algorithm based on Support Vector Machine (SVM). In particular *w-SMOreg* is based on *sequential minimal optimization* (SMO), which is an iterative algorithm for solving the regression problem using SVM. This algorithm is not present in the base library of RapidMIner but it is included in the *Weka* extension.

2.3.3 LibSVM algorithm

The *LibSVM* algorithm is one of the most spread SVM based algorithm. It is an integrated software for support vector classification, that can also support multi-class classification and regression. In this paper it was applied on the input datasets considering all the kernel mentioned above and using epsilon-SVR SVM implementation, specific for solving regression problems.

2.3.4 DecisionStump and RepTree algorithm

The *RepTree* algorithm and the *DecisionStump* algorithms are decision tree based algorithms that support regression. A decision tree is basically a tree where each node represents a splitting condition on an attribute that can split data into groups, where each group of data represents the inputs for which the condition is satisfied or not, in order to be able to classify data giving it a label. The leafs of the tree represent the decision made for the inputs, i.e. the labels that is given to that specific group of data inputs. This algorithm builds a decision/regression tree using information gain/variance (which are measures that gives information about the quality of the splitting condition) and prunes it using reduced-error pruning (with backfitting, to improve its generalization capabilities). This means that starting at the leaves, each node is replaced with most popular class and if the prediction accuracy is not affected then the change is kept [11]. The textitDecisionStump algorithm instead is a simplified decision tree, that splits data ccording to only one splitting condition.

2.3.5 Weka Multilayer Perceptron algorithm

This algorithm is basically a neural net that supports regression. A neural net is a net made up of neurons and connections. Each neuron can be considered as an object that elaborates its inputs and generates some outputs that are given as an input to other neurons using its connections. Neural nets can be used for both classification and regression problems, so the final outputs that are generated give informations about the label of the input data (i.e, its classification) or a numerical prediction, as in this specific case.

Chapter 3

Framework overview and pre-processing

This chapter focuses on the description of the framework I implemented and the step of pre-processing of the car sharing data, that was the preliminary step before running the regression algorithms on the datasets I used.

3.1 Framework Description

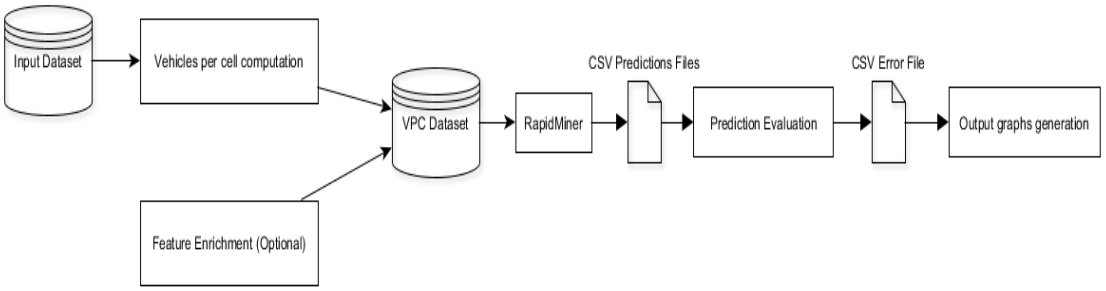


Figure 3.1: Framework Overview

The diagram above shows the main steps through which all the analisys described in this work passed. Here is a brief description of each block, in particular of the ones that correspond to a part of the framework I implemented in Java code:

- *Vehicles per cell computation*: this is the part of the framework I implemented in Java code that is in charge of computing the number of vehicles available for rental inside each cell in which each city I considered was divide. Since the input datasets had different formats for each city the implementation of this block (in terms of code) is slightly different, but the final output is quite the same. For Seattle and Portland dataset it was necessary to clean data, considering that GPS system could locate

a vehicle in different positions in two consecutive timestamps even if the vehicle is actually not moving;

- *Feature Enrichment*: this part of the framework is in charge of adding addition features to the dataset. In particular it adds some timing informations to the dataset produced by the *Vechile per cell coomputation* block. This was a necessary step in order to make an analysis of the prediction error through slots of time and days of the week;
- *Prediction Evaluation*: it computes different error measure in order to evaluate the performance of the various regression algorithms that were run on the input datasets.

3.2 Input Dataset

The input files from which this works starts are *CSV* files (Comma Separated Values) that contains informations about car trips exploiting two popular service providers (that starting from this point will be referred as *provider1* and *provider2*) service providers during a period of time that goes from April 2016 to July 2016 relative to an italian city (which will be referred from now on as *use case 1* city). Such data was generated by retrieving from the server of each service provider the position of each vehicle (sampled every 5 minutes by default) and generating from these informations a dataset containing all the car trips in *use case 1* city. This dataset has been the point of beginning of my work, as all the work I did is relative to the number of vehicles available in each cell in which the city of interest was divided. Each of the input file described above has the following attributes:

- *Vehicle_ID*: it is *string* made of letters and numbers that uniquely identifies the vehicle;
- *Start_Date*: it is a *string* that identifies the timestamp, in the format *yyyy-MM-dd hh:mm:ss*, in which the vehicle starts its travel;
- *Start_Latitude*: it is a *double* number and it is the latitude at which the vehicle starts its travel;
- *Start_Longitude*: it is *double* number and is the longitude at which the vehicle starts its travel;
- *Start_Fuel_Level*: it is an *integer* number (within a range that goes from 0 to 100) that inidcates the level of fuel of the vehicle at tthe *Start_Date*;
- *End_Date*: it is a *string* that identifies the timestamp, in the format *yyyy-MM-dd hh:mm:ss*, in which the vehicle ends its travel;
- *End_Latitude*: it is a *double* number and it is the latitude at which the vehicle ends its travel;
- *End_Longitude*: it is *double* number and is the longitude at which the vehicle ends its travel;

- *End_Fuel_Level*: it is an *integer* number (within a range that goes from 0 to 100) that inidcates the level of fuel of the vehicle at tthe *Start_Date*;
- *Duration*: it an *integer* number that indicates the duration of the car travel (i.e, the absolute difference between *Start_Date* and *End_Date*. It is measured in minutes and it is a multiple of the sampling time (i.e., 5 minutes in this case);
- *Distance*: it is an *integer* number and indicates the distance made by the car during its travel. It is measured in *metres*;
- *Fuel_Delta*: it is an *integer* number that indicates the difference between *End_Fuel_Level* and *Start_Fuel_Level*;

3.3 Spatial granularity

use case 1 city urban area was divided in cells, in order to see wHere the service is most used. Each cell is identified by a progressive numeric ID and covers an area of about $1km^2$. During the various analysis described in this thesis I had to make a selection of these cells, by choosing for the prediction algorithms I applied on the dataset only the data relative to the most dynamic cells. Applying a prediction algorithm on the data relative to a cell in which the number of various stays static for long periods of time is not meaningful in order to see how the algorithm performs, as its work is highly facilitated. It is more interesting instead to apply it on the data relative to cells in which the number of vehicles available for rental varies dinamically through time. This cells are tipically located, as I found out during the various experiments I made, in the central area of the city.

3.4 Time granularity

The time granularity has an impact on how a cell can be considered dynamic or not. It is highly rare that the number of vehicles inside a cell vary if the time at which the number of vehicles is monitored is too small. For example once a cell is left in a cell at time x is not so frequent that leaves the cell in the period of time between x and $x+$ *5 minutes*. For each cell 3 different *csv* files are generated, each one containing the number of vehicles sampled at a different timestamp:

- 5 minutes, which is the default timestamp;
- 10 minutes;
- 15 minutes.

From the code point of view the number of cars is sampled using a 5 minutes timestamp and then the samples are skipped or taken into consideration according to a number passed to the program from the command line, that indicates how many timestamps have to be grouped together to obtain the number of present vehicles. For example, if the number passed by the command line is 2, this means that in the final output file the rows are printed alternatively in the output file, skipping one row.

3.5 Initial input dataset

The input dataset is generated starting from the *provider1* travels dataset collected during May 2016. The input *csv* files contain for each timestamp (i.e, 5 min, 10 min or 15 min) the number of available vehicles in a given cell. If for a certain timestamp there is no vehicle left inside the cell or leaving from the cell, the attribute storing the number of available vehicles is kept unchanged. This attribute (i.e, *numcars*) is set to be a *label* attribute by the Windowing operator, as it is the value that has to be predicted by the regression algorithm *W-SMOreg*. During the analysis **12** cells were considered: 6 belonging to the center of the city and 6 in the peripheral area. The cells are located in the following areas:

- **cell 105**: Center;
- **cell 125**: Center;
- **cell 166**: Peripheral;
- **cell 43**: Peripheral;
- **cell 90**: Center;
- **cell 150**: Peripheral;
- **cell 71**: Peripheral;
- **cell 66**: Center;
- **cell 7**: Peripheral;
- **cell 85**: Center;
- **cell 126**: Center;
- **cell 50**: Peripheral.

The considered cells were chosen among the cells that, during the period of the collection of the data, were starting/destination cells of at least a car trip.

3.5.1 Base dataset

The dataset initially used and obtained by elaborating the dataset described at the beginning of this chapter contains the information of the number of vehicles at each timestamp, for each one of the cells listed in the previous paragraph. Each record contains the following attributes:

- *cell*: the integer value that uniquely identifies the considered cell;
- *timestamp*: the progressive integer number that identifies the timestamp;
- *day*: the integer value that identifies the day within the month in which the number of cars is sampled;
- *month*: the string that identifies the month in which the number of cars is sampled;
- *year*: the integer value that identifies the year in which the number of cars is sampled (it is always equal to 2016 in this case);
- *numcars*: the number of cars that is sampled at the moment identified by the previous four attributes. It is an integer value.

3.5.2 Enriched dataset

In order to make a temporal analysis of the performance of the regression algorithms I used it was necessary to add to the dataset described previously two more attributes:

- *timeslot*: an integer value that represents the period of time in which the timestamp is located. In this case each timeslot is 3 hours wide. So, for example, if the timeslot is equal to 1, it means that the timestamp is located in a period of time that goes from 12am to 3am;
- *week_day*: an integer value that identifies the day of the week that corresponds to the triplet *(day,month,year)*. This number goes from 1 to 7, where 1=Sunday, 2=Monday and so on.

By adding these attributes is was possible to find out how the various applied regression algorithms behave for each day of the week and each timeslot.

3.6 Pre-processing and per cell vehicle number computation

Since the input dataset is a collection of records that contain informations about car trips, two informations are needed in order to compute the number of cars present in each cell at each timestamp:

- the *start coordinates* (i.e., longitude and latitude), to obtain the vehicles that leave a specific cell at a specific timestamp,

- the *end coordinates*, to obtain the vehicles that are parked within a specific cell at a specific timestamp.

The computation in this case is made (initially) for the data concerning May 2016 and it is relative to the *provider1* service provider. The initial number of cars (i.e, the number of cars present in the first timestamp of the 1st May) is set to 0, assuming that I don't have any information about the distribution of the vehicles in the previous month. However, in order to avoid possible computation problems (i.e, negative numbers if a vehicle whose presence in the cell is not known leaves the cell), if the recorded number of vehicles present at a certain timestamp in a cell is set to 0, I check if any vehicle left the considered cell during the time between two consecutive timestamps. If it is so, these vehicles are considered as present vehicles, since before leaving those vehicles are considered available in the time interval between the considered timestamp and the previous one. The number of vehicles present in each cell is then computed adopting an incremental approach: at each timestamp the number of present vehicles is computed adding to the number of vehicles present in the previous timestamp the number of incoming vehicles in that timestamp and subtracting the number of cars that left the cell in the previous timestamp:

$$numcars = numcars_{t-1} + incoming_t - exiting_{t-1},$$

where t is the timestamp taken into consideration. Of course, as I said before, this computation formula is valid for all the records except the first one, that has no previous data to work on. Since the informations about some days are not available (e.g, there is a hole of data between May 18th and May 23rd) the number of vehicles present at the first timestamp of some days cannot be computed. To avoid starting with a number of vehicles equal to 0 in these cases I chose to set the number of vehicles to the average of the number of the vehicles present at the first timestamp in the previous days. Of course it is an approximation, so this number can lead to consider a cell with a level of availability of cars that can be more/less than its actual level of occupancy.

Chapter 4

SVM prediction

This chapter describes in details the prediction of an SVM based algorithm and the operators I exploited to make predictions about the number of available cars in a given area at a given monet of time. It also gives informations about the output files produced by the algorithm and the output of the part of the framework that evaluates the algorithm performance. The final sections show graphically the error distribution of the algorithm.

4.1 Windowing operator

Windowing operator parameters are changed in order to measure the performance of the *W-SMOreg* algorithm, using the datasets described in the first paragraph. I set the Windowing parameters to the following values:

- *window size*: 3. I set it to 3 in order to use data concerning 3 contigous timestamps (i.e, 15 minutes if the sampling time is equal to 5 minutes, 30 minutes if the sampling time is equal to 10 minutes, 45 minutes if the sampling time is equal to 15 minutes). This parameter is kept unchanged;
- *step size*: 1. This parameter is kept unchanged. It is the distance between first values;
- *horizon*: the horizon is the parameter that is changed according to the integer values passed to the program from the command line. The program that evaluates the prediction performances receives as initial parameters the starting horizon and the last horizon value to be considered (both values are included in the analysis). I chose to the consider the following horizon values: 2, 3, 4, 5, 6, 7 and 8. Increasing the horizon value means increasing the distance of the prevision in the future and , in terms of minutes, it increases as the sampling time increases.

Obviously changing the horizon means setting how far the prediction is distant in the future and this distance varies (in terms of minutes) according to the time between two consecutive timestamps:

Sampling time	Horizon	Prediction at
5 minutes	2	10 minutes
5 minutes	3	15 minutes
5 minutes	4	20 minutes
5 minutes	5	25 minutes
5 minutes	6	30 minutes
5 minutes	7	35 minutes
5 minutes	8	40 minutes
10 minutes	2	20 minutes
10 minutes	3	30 minutes
10 minutes	4	40 minutes
10 minutes	5	50 minutes
10 minutes	6	60 minutes
10 minutes	7	70 minutes (1 hour and 10 minutes)
10 minutes	8	80 minutes (1 hour and 20 minutes)
15 minutes	2	30 minutes
15 minutes	3	45 minutes
15 minutes	4	60 minutes
15 minutes	5	75 minutes (1 hour and 15 minutes)
15 minutes	6	90 minutes (1 hour and a half)
15 minutes	7	105 minutes (1 hour and 45 minutes)
15 minutes	8	120 minutes (2 hours)

Table 4.1: Prediction moment for each *(horizon,sampling time)* couple

The windowing operator is also used to create a *label* attribute, which is the attribute whose value that has to be predicted by the regression algorithm. In this case the label attribute is the *numcars* attribute. Since any attribute with *id* or *label* role is not considered by the operator during the windowing operation I had to modify/create new attributes:

- I initially chose as *id* the couple (*timestamp,day*), but in this way the windowing operator didnt' consider these attributes. This means that no *timestamp-x* or *day-x* attributes were generated in the output of the operator, with $x \in [0, (window\_size - 1)]$. I solved this problem using the *Generate Concatenation* operator, generating the attribute *day_timestamp* and setting its role to the *id* value. This attribute is simply a concatenation of the *day* and *timestamp* attributes, using the '_' as separator;
- A similar problem was present for the *numcars* attribute. Since its role was set to *label*, no *numcars-x* attributes were generated in the output, with $x \in [0, (window\_size - 1)]$. So I had to use the *Generate Copy* operator to create a copy of the *numcars* attribute, naming it *cars#* and setting its role to *regular* attribute.

4.2 The W-SMOreg algorithm

The regression algorithm I chose to use is the *W-SMOreg*, a regression algorithm based on SVM. The W-SMOreg algorithm is an algorithm included in the *Weka* extension of

Rapidminer, that is a suite of machine learning algorithms developed at the University of Wakato, New Zealand. I chose to keep the algorithm parameters unchanged:

- **C**: the *complexity*. It is a complexity parameter which SMO support vector machine uses to build the *hyperplane* to separate the data. It controls how soft the class margins are, in practice how many instances are used as 'support vectors' to draw the linear separation boundary in the transformed euclidean feature space. By default the complexity constant is set to 1.0;
- **N**: the *normalize* parameter. It is set to 0, that means normalize;
- **I**: it is a string that contains all the parameters of the optimizer class used for solving the quadratic optimization problem;
- **K**: the *kernel* to use. In this case the default kernel is the *Polynomial* kernel.

The most significant attribute (i.e, the one with the highest weight assigned by the algorithm) in predicting the number of available cars is the *cars#-0* attribute, that contains the number of available cars registered in the previous timestamp before the value specified by the horizon parameter. Its weight at each run is very close to 1. The algorithm is run once for each cell, for each of the three sampling times I chose to consider in my analysis. The computational time necessary to predict the data seems to be affected by the position and occupancy level of the cell. For cells in the peripheral area of the *use case 1* city, the algorithm seems to take much less time than for the other cells in the center of the city. To validate the algorithm I chose a *X-validation*, partitioning the data in 10 partitions. Nine of them are used to train the W-SMOreg algorithm and the remaining one is used to test the quality of the prediction.

4.3 Results file format

Each results file is relative to a specific cell and its named *results_cellX_Ymin_sampling*, where X is the cell number and Y is the sampling time. Each file contains, for each record, the predicted value of the available cars in a given cell and month:

- **timestamp-2**: the integer value of the previous third timestamp;
- **timestamp-1**: the integer value of the previous second timestamp;
- **timestamp-0**: the integer value of the previous timestamp;
- **day-2**: the integer value of the previous third day;
- **day-1**: the integer value of the previous second day;
- **day-0**: the integer value of the previous day;
- **cars#-2**: the number of cars available in the considered cell going back in the past for an amount of time equal to $horizon + 2 * timestamp$;
- **cars#-1**: the number of cars available in the considered cell going back in the past for an amoount of time equal to $horizon + 1 * timestamp$;

- **cars#-0**: the number of cars available in the considered cell going back in the past for an amount of time equal to *horizon* ;
- **label**: the number of cars available in the considered cell in the current day at the current timestamp (i.e., the value to be predicted);
- **day_timestamp**: the ID that uniquely identifies the record;
- **prediction(label)**: the prediction value (i.e., the output of the W-SMOreg regression algorithm). It is a *double* number.

4.4 Prediction error computation

Once all the results of the regression algorithm are generated for all the cells and for all the horizon values I chose to consider, the next step is to evaluate how much the regression algorithm was accurate in predicting the number of available cars in the future. In order to do this two measures were computed:

- **Mean Abs Error, MAE**: this error measure can be expressed as $|MAE = predicted\_value - actual\_value|$, where *predicted_value* is the output of the regression algorithm and *actual_value* is the actual number of cars available that can be rented in the considered cell;
- **Adjusted Mean Abs Error, A-MAE**: this error measure instead is computed as $A-MAE = |rounded\_predicted\_value - actual\_value|$, where *rounded_predicted_value* is the *predicted_value* approximated to the closest integer number, since the output of the regression algorithm is a *double* number and the number of cars present in a cell must be an integer value. This kind of approximation however can lead to a prediction that includes a car that can be actually not present.

This two measure are then written in a file where each row has the following attributes:

- **horizon**: the horizon set for the Windowing Operator(range from 2 to 8, both included);
- **cell**: the cell for which the prediction was made by the regression algorithm;
- **samplingTime**: the distance between two consecutive timestamps (5 minutes, 10 minutes or 15 minutes);
- **mean_abs_error_r**: the *A-MAE* value;
- **mean_abs_error_nr**. the *MAE* value.

These data are then used to generate, for each cell, a graph that shows three different curves (one for each possible sampling time) that represent how the performances change using different values of the horizon (i.e, how performances evolve as the distance between the window and the prediction in the future increases). The performances are best for the cells where the range of values of the available cars is less wide (for example cell 7 or 50, that are both in the peripheral are of the city) and both MAE and A-MAE are very close

to 0, independently from the sampling time. Performances decrease instead for the cells in the central area, and differ significatively for the three considered sampling times. MAE and A-MAE assume a range of values between 0 and 2.5 . The error evaluation is done inside the *predictionEvaluation* Java project, that elaborates the results of the regression algorithm in order to compute the *MAE* and *A-MAE*.

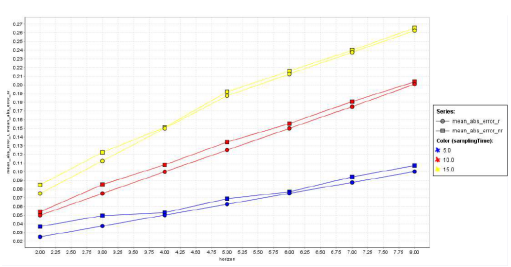
4.5 Performance evaluation output file format

The performance evaluation output file (name *prediction_performances.csv* has the following fields:

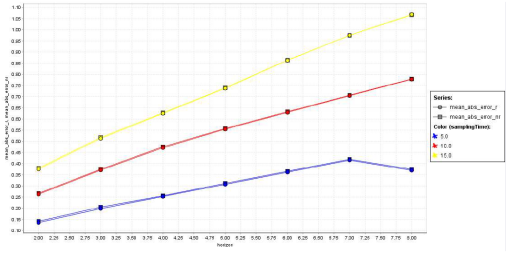
- **horizon**: horizon value used by the *Windowing operator* (range from 2 to 8, both included);
- **cell**: cell number;
- **samplingTime**: time interval between two consecutive timestamps in which the number of cars is computed;
- **mean_abs_error_r**: A-MAE error measure;
- **mean_abs_error_nr**: MAE error measure.

4.6 Prediction error graphs - provider1

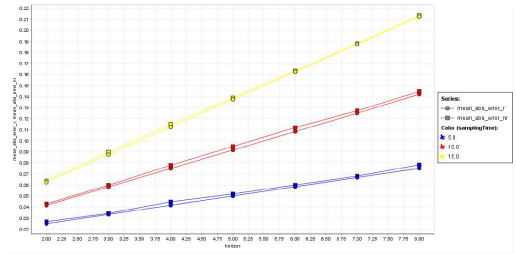
Graphs are generated from the file described in the previous paragraph. Each graph shows in the horizontal and vertical axis respectively the horizon value and MAE and A-MAE measures.



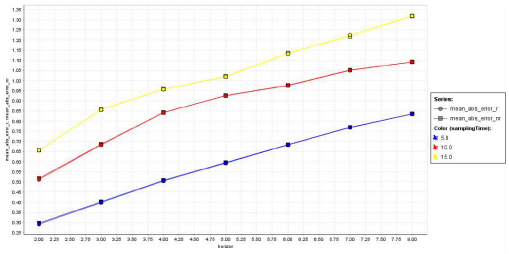
(a) *Cell 7*



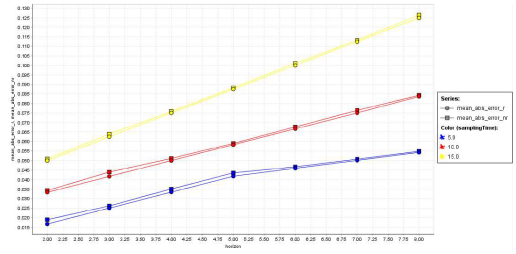
(b) *Cell 43*



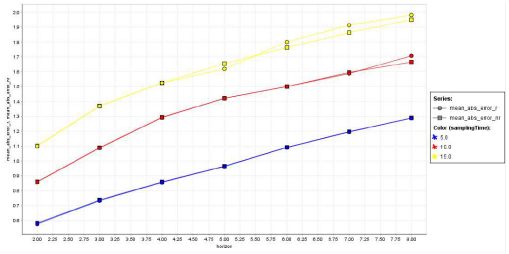
(c) *Cell 50*



(d) *Cell 66*

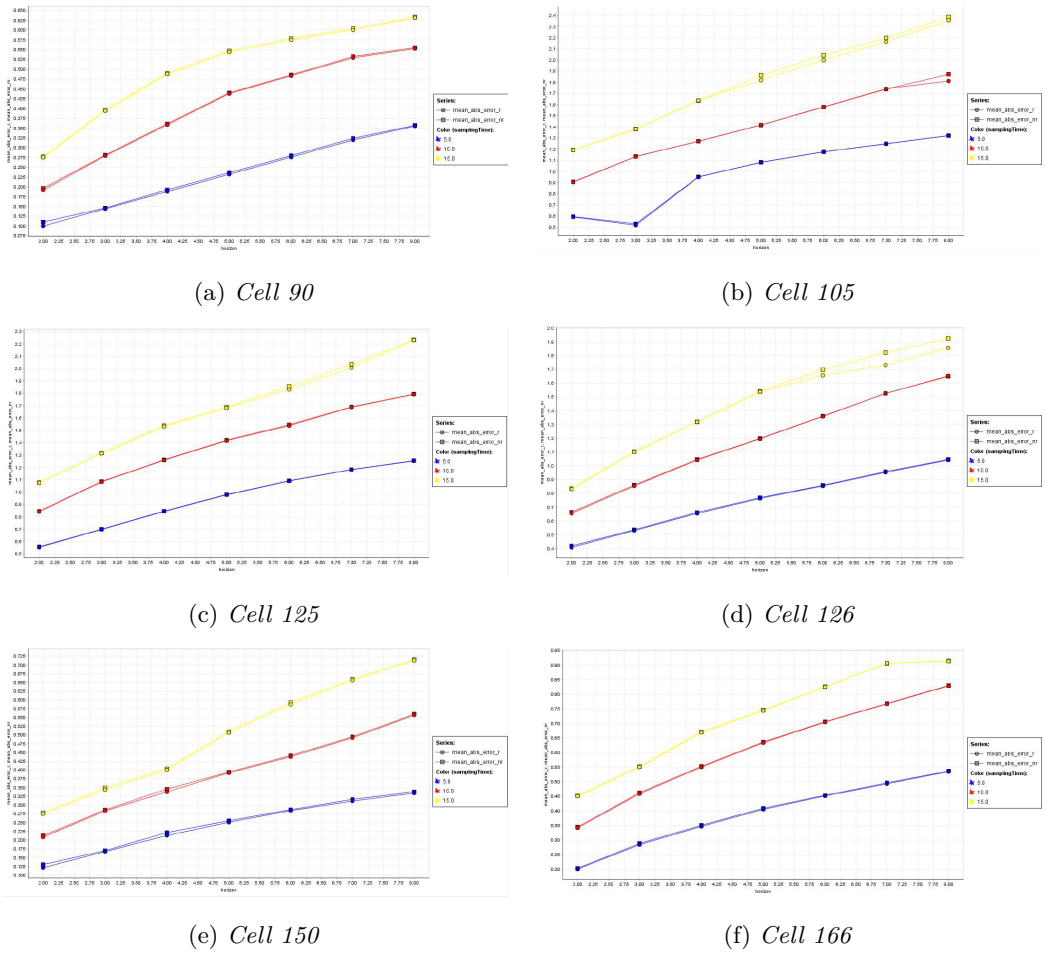


(e) *Cell 71*



(f) *Cell 85*

Figure 4.1: Prediction error graphs - provider1 (1)



(a) *Cell 90*

(b) *Cell 105*

(c) *Cell 125*

(d) *Cell 126*

(e) *Cell 150*

(f) *Cell 166*

Figure 4.2: Prediction error graphs - provider1 (2)

Starting from these graphs I did an analysis with the final scope to determine which are the best values of the horizon and sampling time to be used further:

- as the horizon value increases, as it was expected, the error increases. However the behaviour is very similar if we considered the curves that represent each sampling time. The error approximatively doubles from an horizon value equal to 2 to an horizon value equal to 8. Since the number of present vehicles in each cells does not seem to vary significatively in a period of time equal to the prediction horizon, I chose for the future analysis an horizon equal to 8. In this way the data used for prediction are at a significant distance from the data to predict and the error can be more interesting to be analyzed. It should also be consdiered that an average user of the service can be more interested in knowing the availability of vehicles not in the immediate future but in a moment located further.

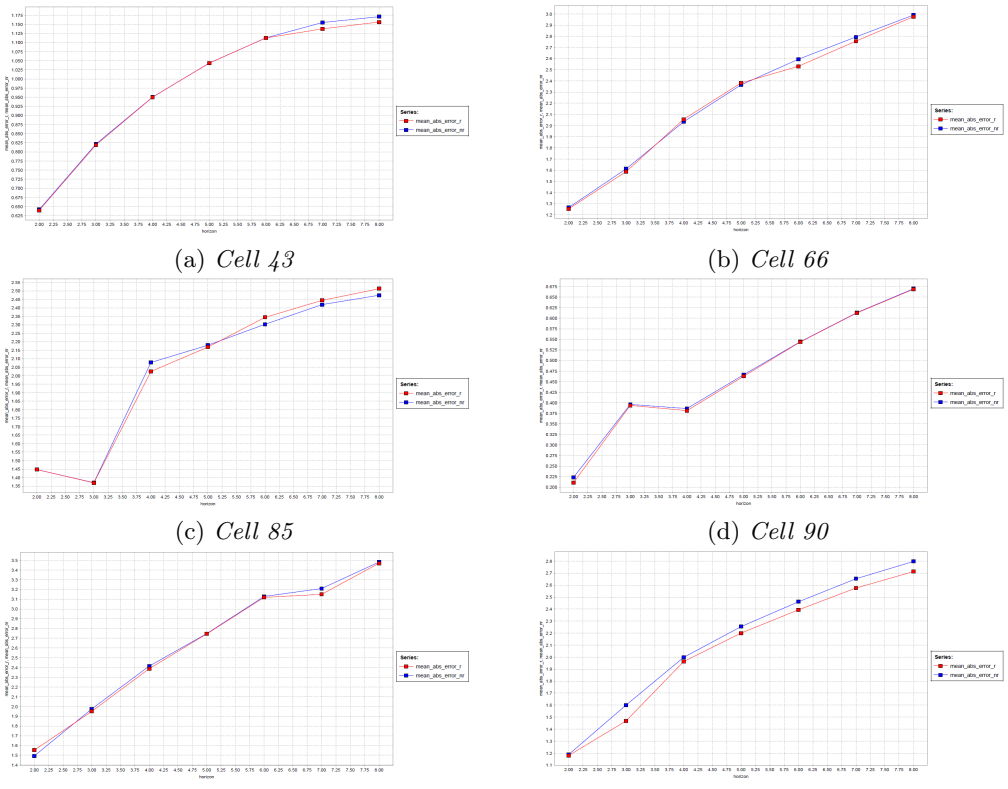
- also increasing the sampling time, as it was expected, leads to an increase of the error. However the error gap between the three curves is not significatively high enough to

prefer one of them instead of the others. In this case I prefered choosing 15 minutes as sampling time, because the number of available cars in a cell has more probability to vary in a slot of time equal to 15 minutes than in a slot equal to 10 or 5 minutes;

- the difference between the two error measures instead is not significant. In the future analysis the only error measure used will be the A-MAE measure.

4.7 Prediction error graphs - provider2

The same analysis done for the *provider1* input dataset is done using as an input the information about the tracks concerning *provider2* vehicles. The graphs below show how the error evolves as the horizon of the prediction increases. However it was not possible to do such analysis for all the cells considered for the *provider1* dataset, because the number of vehicles didn't vary significantly among the considered days. This could be imputed to the smaller number of parking machines of the provider and to the rare fruition of the service in the peripheral areas of the city. Infact all the cells for which the regression algorithm could not be applied are located quite far from the center of the *use case 1* city.



(a) *Cell 43*

(b) *Cell 66*

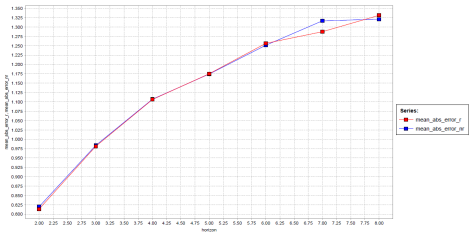
(c) *Cell 85*

(d) *Cell 90*

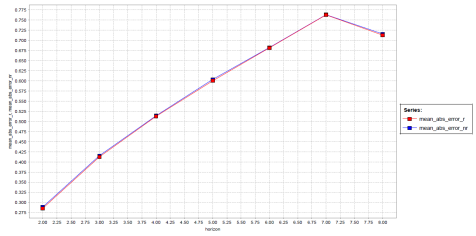
(e) *Cell 105*

(f) *Cell 125*

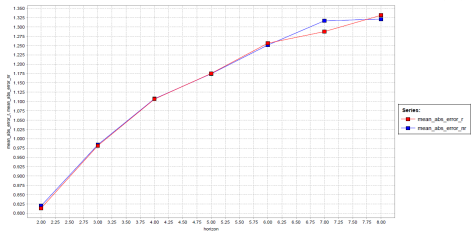
Figure 4.3: Prediction error graphs - provider2 (1)



(a) *Cell 126*



(b) *Cell 150*



(c) *Cell 166*

Figure 4.4: Prediction error graphs - provider2 (2)

The same considerations made for the *provider1* graphs set can be made for this set, taking into consideration that in this case the sampling time is equal to 15 minutes. Also in this case increasing the horizon value leads to an increase of the average error. Such behaviour happens for all the cells considered in the analysis. The error for the highest horizon value is approximatively the double of the value assumed for the lowest value.

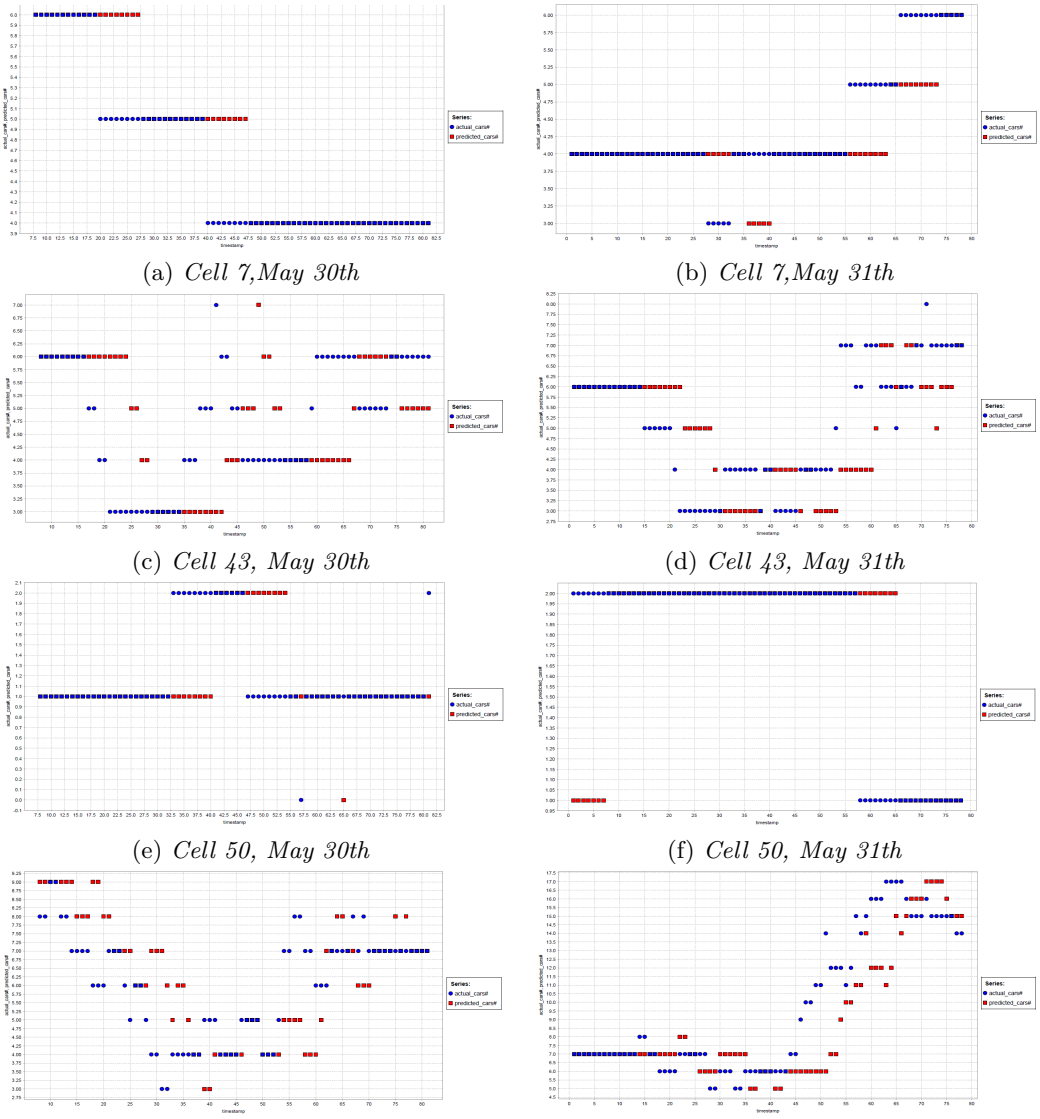
Chapter 5

Baseline prediction

This chapter describes the results, in terms of accuracy, of a prediction made considering only the number of cars present in the past, without considering any other information when making predictions. This analysis, called *Baseline* does not consider any time data, but consider as a prediction the number of cars present at a certain moment in the past in a given area.

5.1 Baseline analysis - provider1

The results obtained by applying the *W-SMOreg* regression algorithm seems to predict quite well the occupancy of the cells and the prediction is more accurate when the number of cars in the considered cell does not vary significantly during the day. In order to see how other regression algorithms behave doing this task, I implemented a simple regression algorithm that predicts the number of cars with an horizon of 2 hours (8 timestamps in this case). The data used to apply the algorithm takes as an input the CSV file concerning the number of cars in each cell sampled every 15 minutes, concerning the two-days data that are used to test the *W-SMOreg* algorithm. The idea on which the regression is based is that the number of cars in the next 2 hours is the same that is present in the current timestamp. The following graphs show for each timestamp the number of cars actually present in the cell and the predicted number of cars using this approach. These graphs also put in evidence which are the cells where the number of cars vary significantly during the day and the timeslots where the prediction is more inaccurate. This kind of approach is clearly more efficient when I have an almost constant number of cars in the cell during all the considered timeslots, as it happens for example for cell 71 during May 30th. In general, the prediction is close or equal to the actual number of cars for all the cells where I have less movement (i.e., in the cells where the service is less used).



(a) *Cell 7,May 30th*

(b) *Cell 7,May 31th*

(c) *Cell 43, May 30th*

(d) *Cell 43, May 31th*

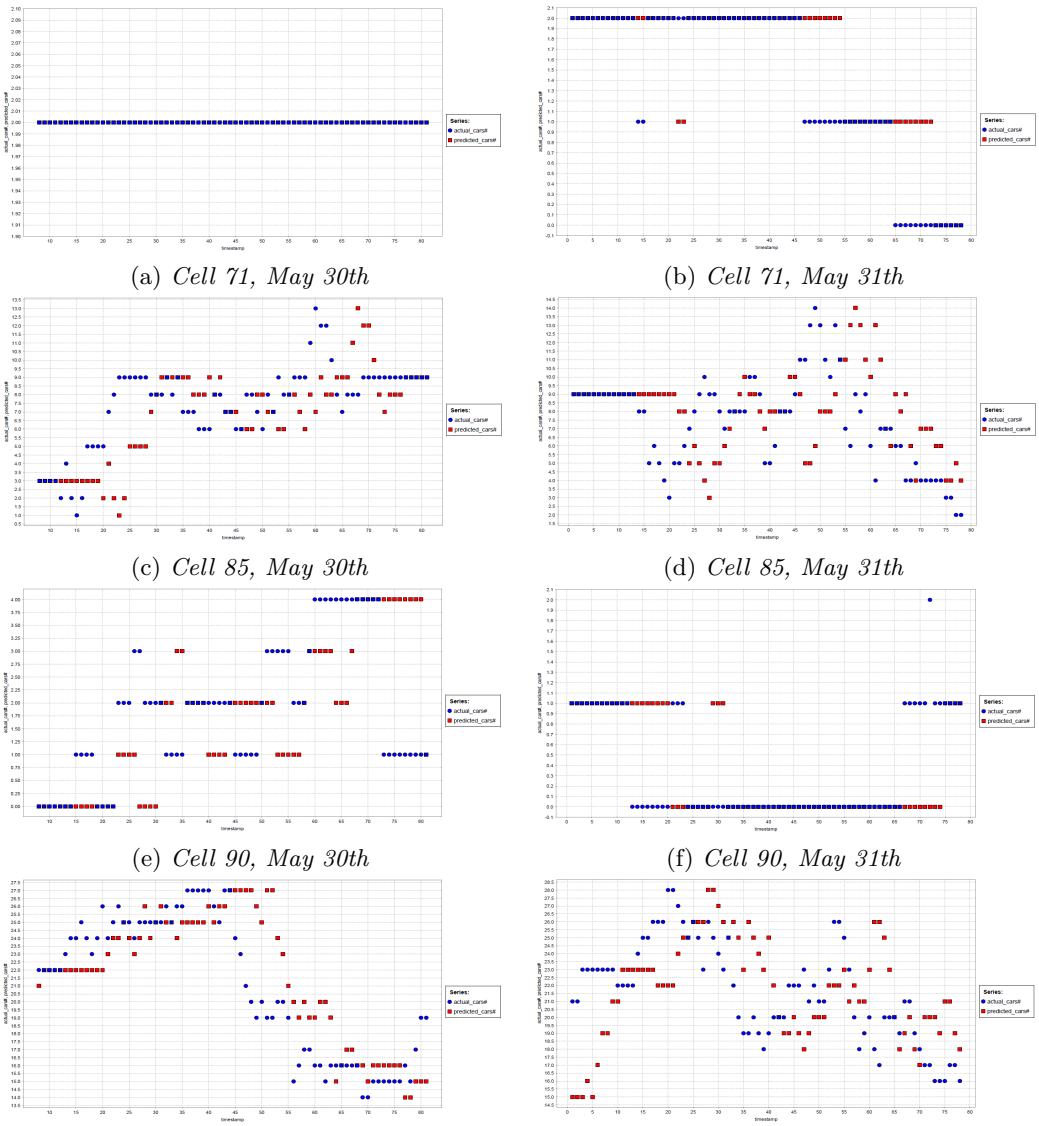
(e) *Cell 50, May 30th*

(f) *Cell 50, May 31th*

(g) *Cell 66, May 30th*

(h) *Cell 66, May 31th*

Figure 5.1: Baseline graphs - provider1 (1)



(a) *Cell 71, May 30th*

(b) *Cell 71, May 31th*

(c) *Cell 85, May 30th*

(d) *Cell 85, May 31th*

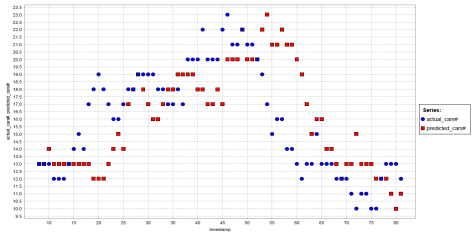
(e) *Cell 90, May 30th*

(f) *Cell 90, May 31th*

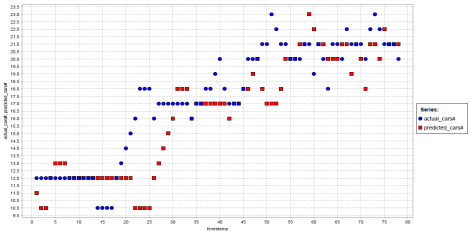
(g) *Cell 105, May 30th*

(h) *Cell 105, May 31th*

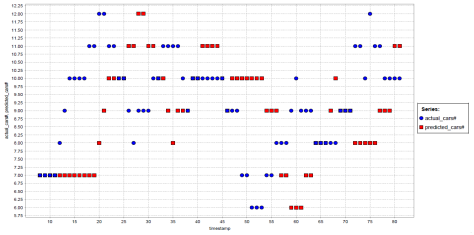
Figure 5.2: Baseline graphs - provider1 (2)



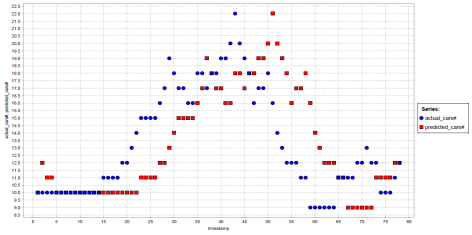
(a) *Cell 125, May 30th*



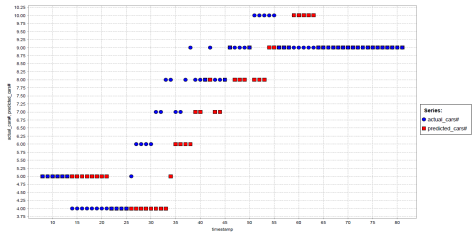
(b) *Cell 125, May 31th*



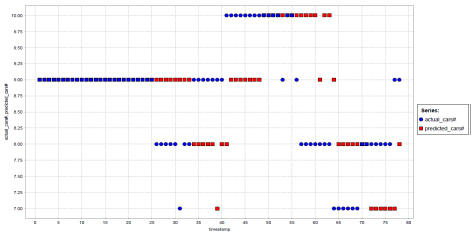
(c) *Cell 126, May 30th*



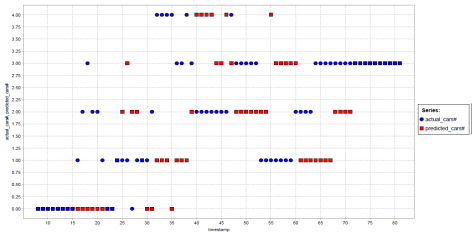
(d) *Cell 126, May 31th*



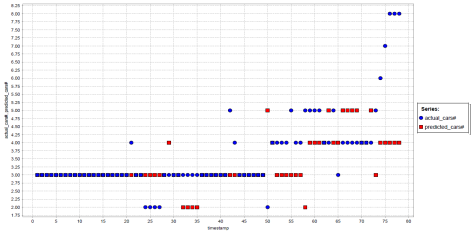
(e) *Cell 150, May 30th*



(f) *Cell 150, May 31th*



(g) *Cell 166, May 30th*



(h) *Cell 166, May 31th*

Figure 5.3: Baseline graphs - provider1 (3)

The previous graphs show that the cells with the highest service usage are cell 66,85, 105,125 and 126. Indeed these are the cells where the baseline predicted value differs significatively from the the actual number of cars, so during a period of two hours the number of available vehicles changes dinamically in this city areas. It is surely not an accident that all these cells are located in the center of the city. All the following analysis is made considering these cells as points fo reference. The following graph shows how the MAE measure evolves during time, for all the cells named previously and for both the days whose data is used to test the *W-SMO-reg* algorithm.

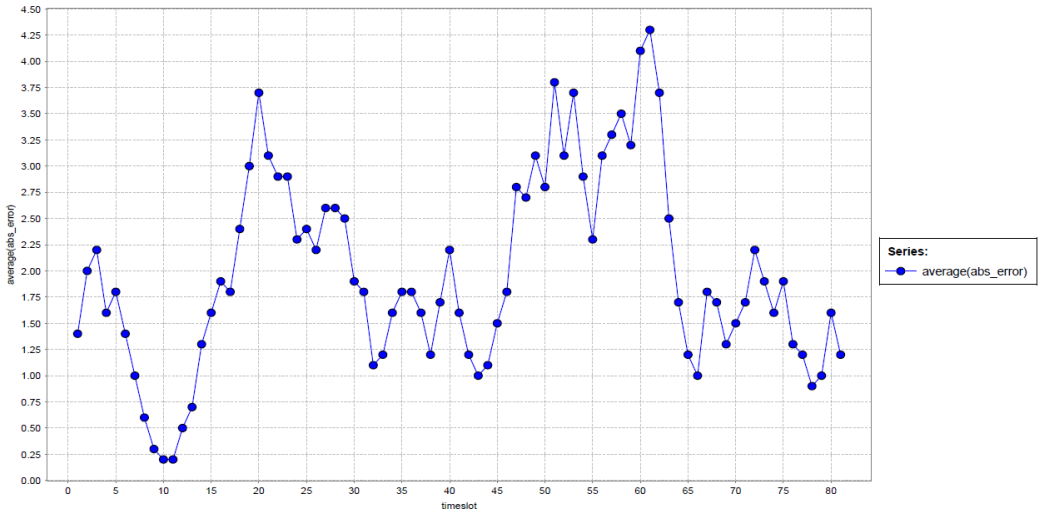


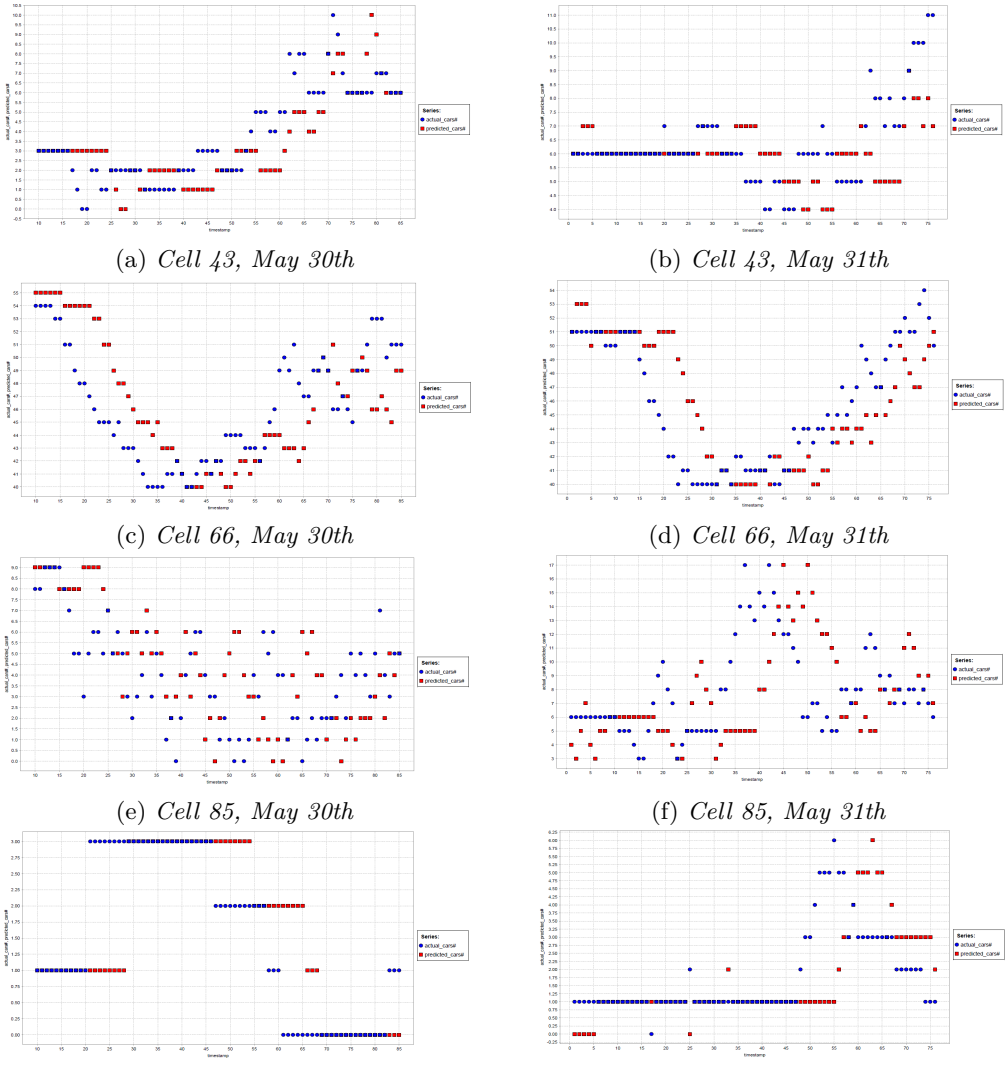
Figure 5.4: MAE evolution graph

The previously shown graph puts in evidence which are the timeslots in which the prediction based on the baseline is less accurate. In particular two periods of time seem to be affected by the most significative prediction error:

- the period between 19th and 29th timestamps (i.e., indicatively between 5am and 7:30am);
- the period between 51th and 61th timestamps (i.e., indicatively between 12:30pm and 3pm).

5.2 Baseline Analysis - provider2

The same analysis described in the previous paragraph was done for the data concerning the availble vehicles in each cell for the *provider2* service provider. The results are quite similar to the one obtained for the *provider1* provider, either in terms of accuracy of the prediction or in terms of cells with the highest traffic of incoming/exiting vehicles in the considered timeslot (i.e., 2 hours). The graphs below show the number of cars predicted using the baseline approach and the actual number of cars available for rental in each cell.



(a) *Cell 43, May 30th*

(b) *Cell 43, May 31th*

(c) *Cell 66, May 30th*

(d) *Cell 66, May 31th*

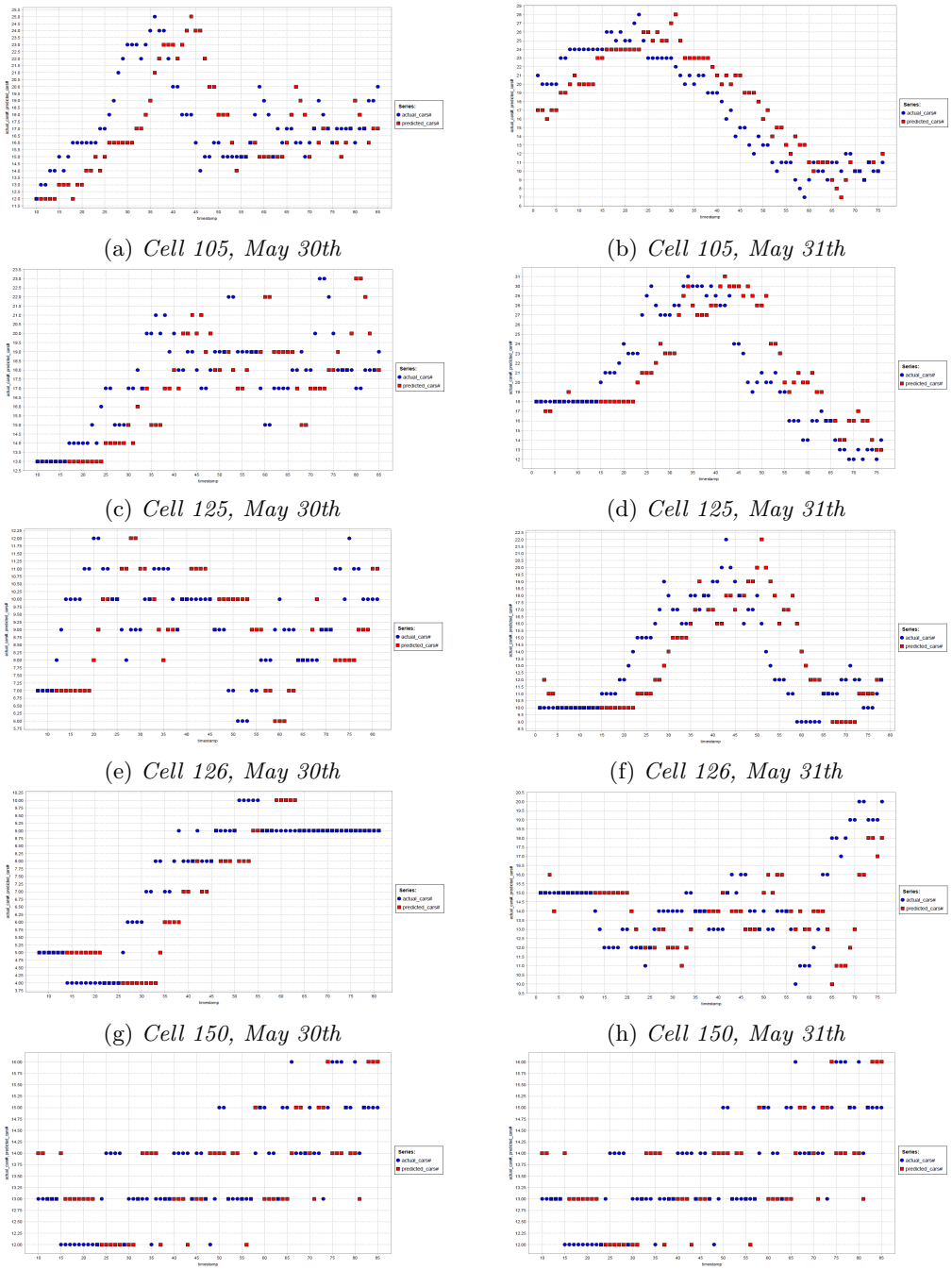
(e) *Cell 85, May 30th*

(f) *Cell 85, May 31th*

(g) *Cell 90, May 30th*

(h) *Cell 90, May 31th*

Figure 5.5: Baseline graphs - provider2 (1)



(a) *Cell 105, May 30th*

(b) *Cell 105, May 31th*

(c) *Cell 125, May 30th*

(d) *Cell 125, May 31th*

(e) *Cell 126, May 30th*

(f) *Cell 126, May 31th*

(g) *Cell 150, May 30th*

(h) *Cell 150, May 31th*

(i) *Cell 166, May 30th*

(j) *Cell 166, May 31th*

Figure 5.6: Baseline graphs - provider2 (2)

As I did for the *provider1* dataset I generated a dataset that for each timestamp gives an information about the MAE measure, considering all the cells. The timestamp for which the error is higher are quite the same that for the previously considered case.

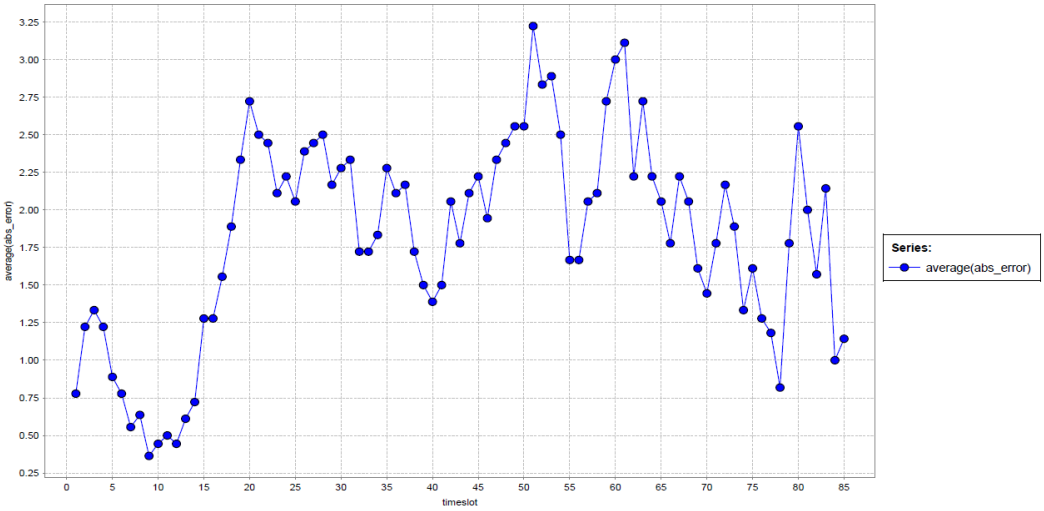


Figure 5.7: Baseline error distribution by timeslot

Chapter 6

SVM prediction using Sliding Window operator

This chapter describes another method of validation of the same regression algorithm described in chapter 4. It also shows how changing the parameters of this kind of validation changes the performance of the regression algorithm.

6.1 Sliding Window operator

Since now the *W-SMOreg* was validated using the *Cross-validation* operator, splitting data into 10 subsets. 9 of the subsets are then used to train the algorithm and the remaining one to test it. In order to see how wide has to be the training set to obtain good performances, I chose to validate the regression algorithm using the *Sliding Window* operator. This operator validates an algorithm through two windows: one relative to the training data and one relative to the test data. This two windows are moved forward of the specified step and are separated by a gap defined by the *horizon* parameter. The data on which the regression algorithm is applied is the same used previously and it is relative to the number of vehicles present in each cell, monitored every 15 minutes. The *Sliding Window* operator is configured with the following parameters. These parameters are changed to see how the size of the two windows affects the regression algorithm performances:

- **training window width (number of records)**: 96 (1-day data), 192 (2-days data), 480 (5-days data), 672 (1 week data), 960 (10-days data) amd 1344(2-weeks data).
- **training windows step size(number of records)**: -1. It means that the training window is moved forward for a number of examples equal to the test window size.
- **horizon (number of records)**: 8. It is the gap between training and test window.
- **test window width(number of records)**: 96 (1-day data), 480(2-days data), 672 (1 week data).

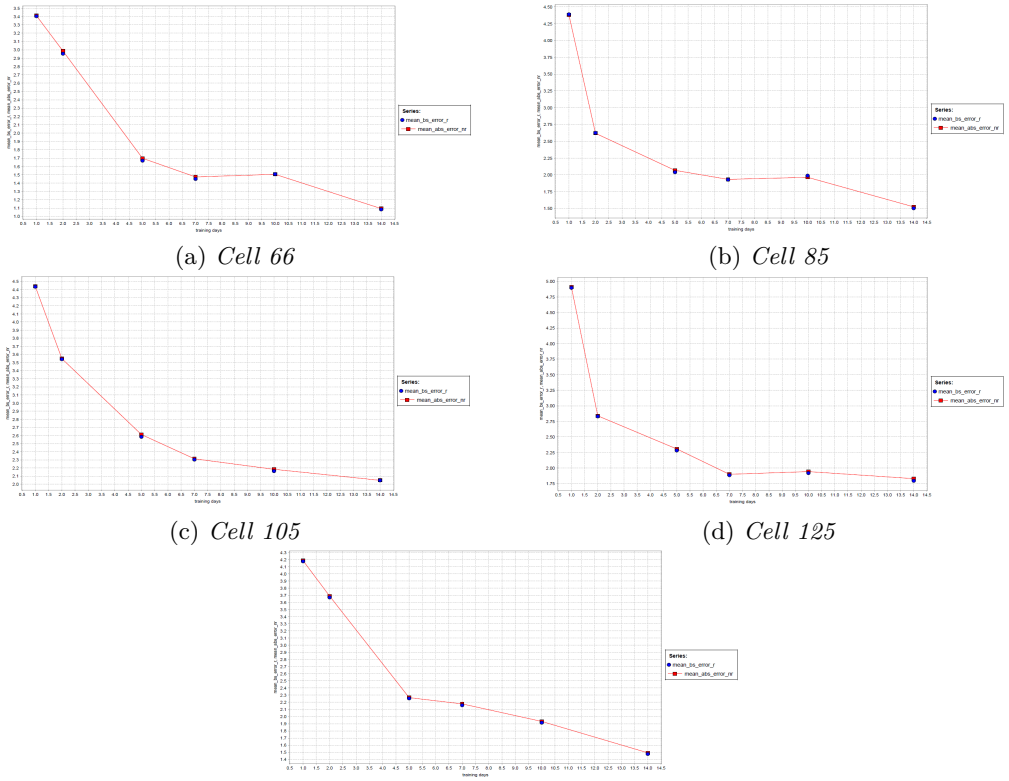
The *Sliding Window* validation is applied to the data concerning the cells with the highest traffic of incoming and exiting cars, which are:

- cell 66;
- cell 85;
- cell 105;
- cell 125;
- cell 126.

All these cells belong to the central area of the city and, according the *Baseline* analysis done in the previous chapter, are the cells in which the number of available cars changes more dinamically. The folllowing graphs show how the MAE and A-MAE error measures change according to the training window size.

6.2 Performance analysis by changing training/test window width

6.2.1 Performance by varying training window width



(a) *Cell 66*

(b) *Cell 85*

(c) *Cell 105*

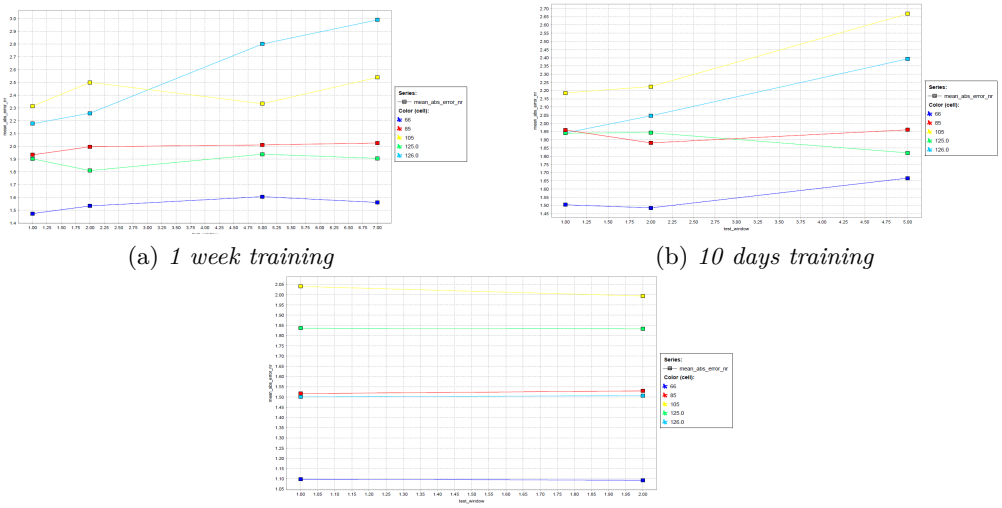
(d) *Cell 125*

(e) *Cell 126*

Figure 6.1: Sliding window performance with varying training window width

The most significant information that can be extracted from these graphs is that in most of the considered cells the error decreases as the training window size increases. This means that the *W-SMOreg* algorithm behaves better as it has more data available for the training phase. Since the smallest training window sizes did not lead to an accurate prediction I chose to go on with my analysis focusing on three window sizes: 1 week, 10 days and 2 weeks. The following step is to see how the test window size can affect the performance of the validation, showing how (for a fixed training window size) the error evolves when the test window width increases. The graphs below show such variation. Each graph concerns a constant training window width and each curve is relative to each of the cells mentioned above.

6.2.2 Performance by varying test window width



(a) *1 week training*

(b) *10 days training*

(c) *2 weeks training*

Figure 6.2: Sliding window performance with constant training window width and varying test window width

Since the analysis was done using as an input the data concerning May 2016, it was not possible to extend the test window width for some training window widths. The graphs above show the results for the following training and test window widths.

Training window width	Test window width
7 days	1 day
7 days	2 days
7 days	5 days
7 days	7 days
10 days	1 day
10 days	2 days
10 days	5 days
10 days	7 days
14 days	1 day
14 days	2 days

Table 6.1: Training window and test window values

Chapter 7

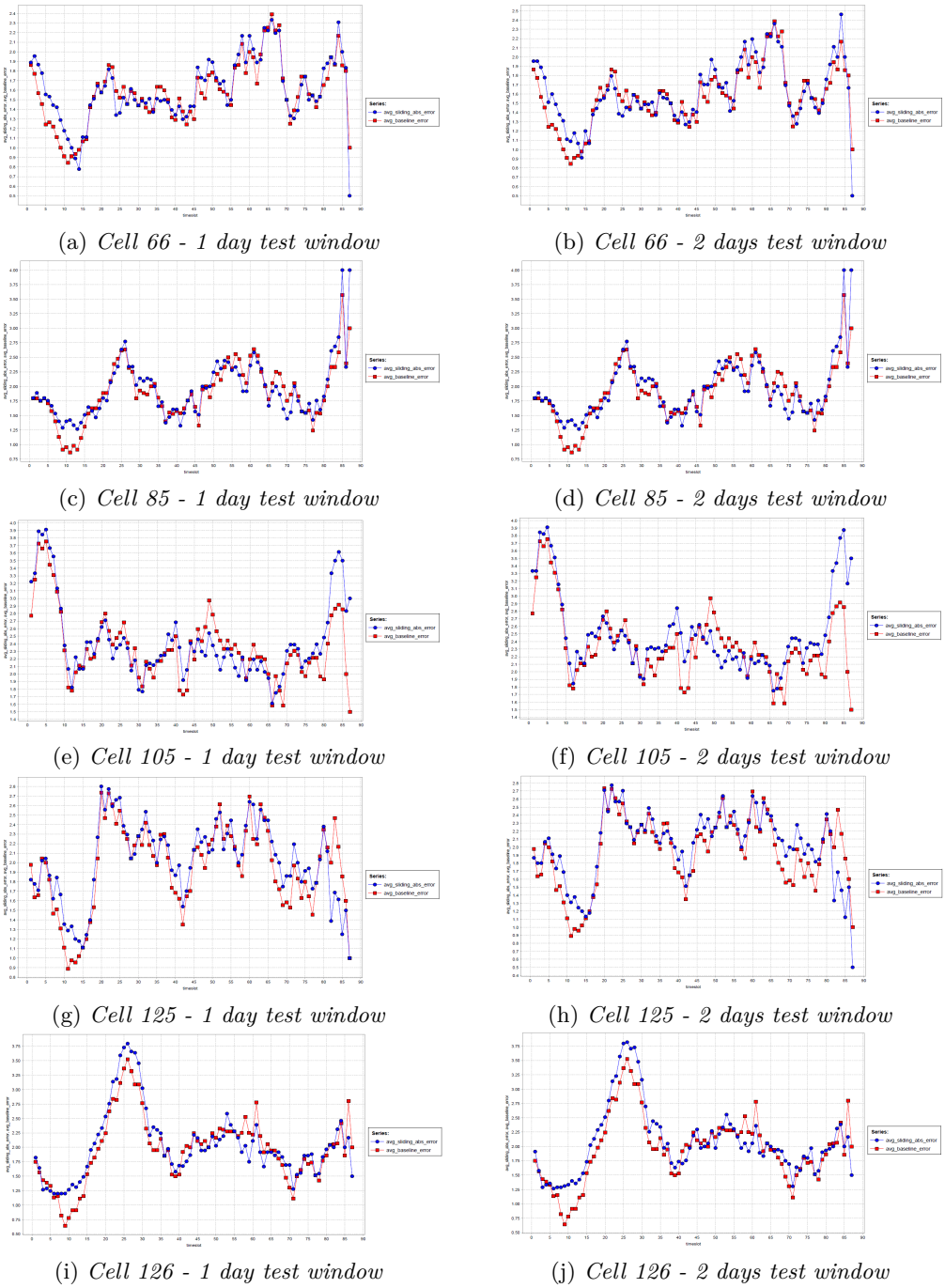
Baseline and SVM comparison

This chapter makes a graphical comparison of the same SVM based regression algorthm described in the previous chapters and the *Baseline* tecnique.

7.1 Baseline and SVM comparison graphs

The graphs below show a comparison between the prediction performance of the *W-SMOreg* algorithm with Polynomial Kernel and the Baseline algorithm. Data were grouped by timestamp, in order to see which were the timeslots where the prediction performance was worst. These graphs show that performances between the two algorithms are very similar. Probably the accuracy of the SVM-based algorithm was affected by two factors:

- the approximation of the predicted number of vehicles;
- the width of the window generated by the *Windowing* operator, that is currently set to 3 (i.e, 45 minutes).



(a) *Cell 66 - 1 day test window*

(b) *Cell 66 - 2 days test window*

(c) *Cell 85 - 1 day test window*

(d) *Cell 85 - 2 days test window*

(e) *Cell 105 - 1 day test window*

(f) *Cell 105 - 2 days test window*

(g) *Cell 125 - 1 day test window*

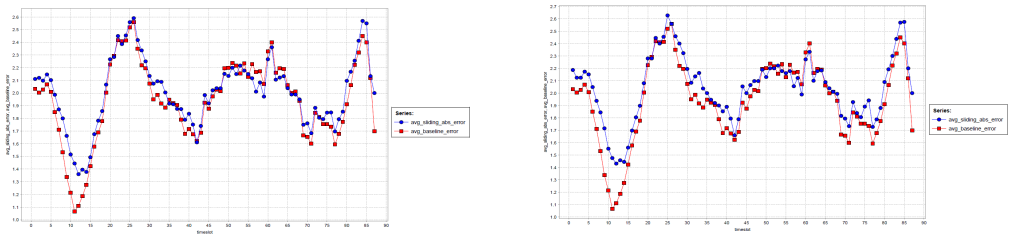
(h) *Cell 125 - 2 days test window*

(i) *Cell 126 - 1 day test window*

(j) *Cell 126 - 2 days test window*

Figure 7.1: SVM performance compared to Baseline performance - per cell

These results are obtained using as input a larger dataset than the one cosidered before. Such dataset concerns a period that goes from the second half of April 2016 to the end of June 2016. The graphs below instead show the average prediction errors considering all the cells.



(a) *1 day test window* (b) *2 days test window*

Figure 7.2: SVM performance compared to Baseline performance - all cells

All these graphs show that the baseline and the *W-SMOreg* algorithm have very similar accuracy when predicting the number of available cars in the future. In particular the *W-SMOreg* does not have a significant improvement or decrease of accuracy when extending the test window size. So even if the SVM considers more informations when making a prediction it can not give better results than an algorithm that uses a sort of naive approach. This could happen because the *W-SMOreg* algorithm uses data that are too far from the moment for which the prediction is made. The *Baseline* approach instead has a different prediction horizon, as it considers for making a prediction only data that are very close to the moment of prediction.

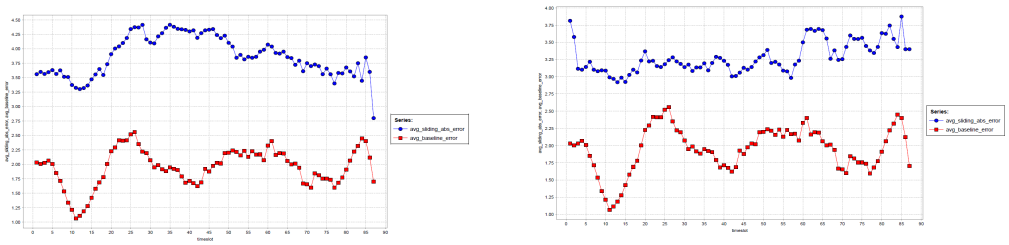
Chapter 8

Other algorithms prediction

This chapter describes the performance of some regression algorithms based on different techniques from the one considered so far.

The same analysis that is described in the previous paragraph was done using other prediction algorithms: *W-DecisionStump*, *W-MultilayerPerceptron*, and *W-RepTree* algorithm, keeping the default parameters. In both cases the prediction perfomance was worse than the Baseline algorithm performance. Both graphs consider the prediction error considering all cells.

Baseline and the other regression algorithms start from different points of time. Infact every different implementation of the baseline skips a number of records from the input file equal to the prediction horizon. The input files of the other algorithms are instead all aligned together (i.e, the prediction output files start from the same day and timestamp). Since the baseline and the prediction results are not aligned, I had to align them by using the *Filtering Examples By Range* operator. In this way I can filter the examples by range specifying the indexes of the examples in the dataset, where the index is the number of the row in the input file. The *MultilayerPerceptron* algorithm is based on backpropagation and exploits it to classify instances. The use of this algorithm leads to the building of a network, in which all the nodes are usually sigmoids. In this case, since the class is numeric, the output nodes become unthresholded linear units.



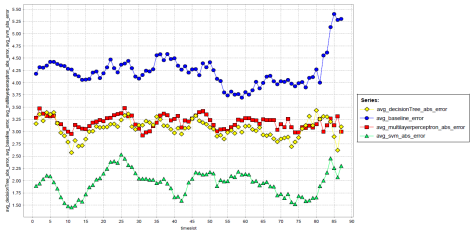
(a) *W-DecisionStump performance* (b) *W-MultilayerPerceptron*

Figure 8.1: Other algorithms performances compared to Baseline performance - all cells

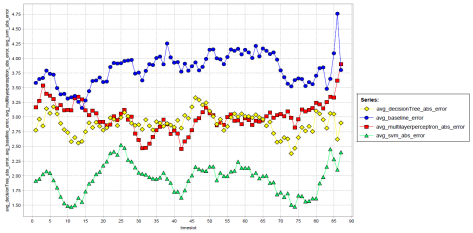
The *W-DecisionStump* algorithm has the worst performance among all the prediction algorithms considered in this paper. Indeed it is a decision tree that has only a root

node that uses the *cars-0* attribute to discriminate the cells and to make a prediction. From the computational point of view it is instead the fastest algorithm, thanks to its semplicity. The *MultilayerPerceptron* algorithm instead has the worst performance in terms of computational time (i.e, it is the slowest algorithm used so far) and makes predictions that are quite worse than the ones made by the *W-SMOreg* algorithm. Since predictions made by *W-DecisionStump* were so inaccurate, I switched this decision tree algorithm with the *W-RepTree* algortithm. The *W-Reptree* algorithm is based on a decision tree that can make a decision/regression using information about gain/variance and prunes it using reduced-error pruning (with backfitting).

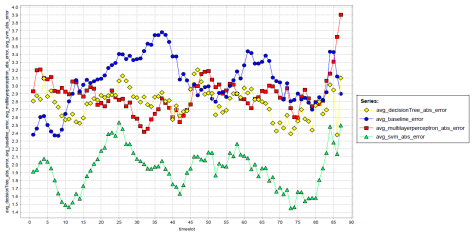
The graphs below show the comparison between four different regression algorithms: *Baseline algorithm*, *W-RepTree*, *W-MultilayerPerceptron* and *W-SMOreg* for both *provider1* and *provider2*. Each graph refers to a different test set dimension for the last three algorithms and to a different prediction horizon for the textitBaseline algorithm. This parameter is named h and it assumes the following values: 96, 48, 24 and 12. This values correspond to 1 day, 12 hours, 6 hours and 3 hours, since the default timestamp is set to 15 minutes.



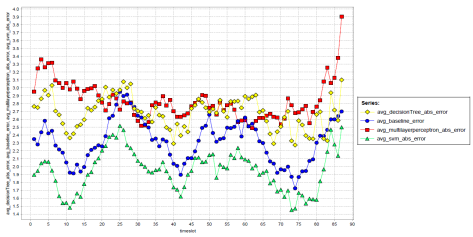
(a) *h=96 - provider1*



(b) *h=48 - provider1*

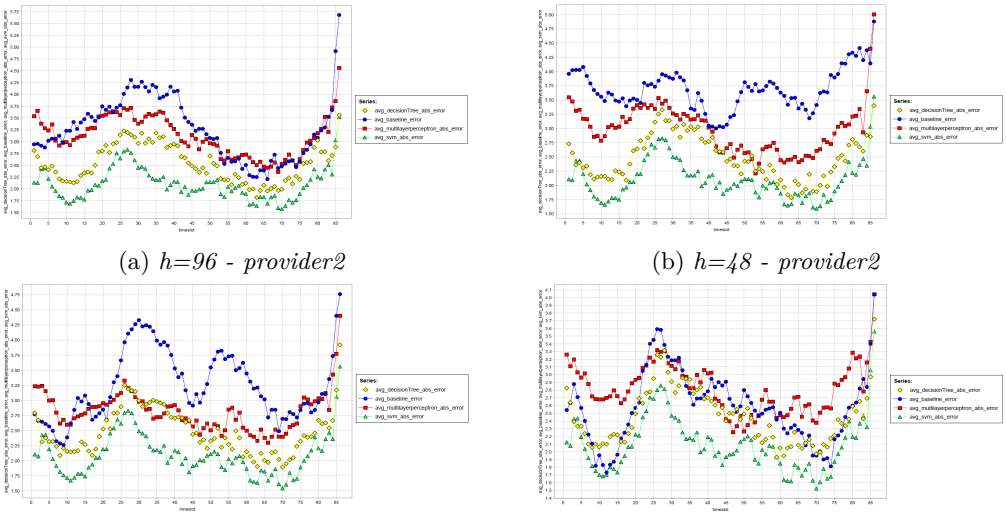


(c) *h=24 - provider1*



(d) *h=12 -provider1*

Figure 8.2: Other algorithms performances compared to Baseline performance - all cells - provider1



(a) *h=96 - provider2*

(b) *h=48 - provider2*

(c) *h=24 - provider2*

(d) *h=12 - provider2*

Figure 8.3: Other algorithms performances compared to Baseline performance - all cells - provider2

Results given by this graphs are quite clear. Predictions made by the SVM-based algorithm are the best ones and are quite close to the ones made by the *Baseline* algorithm. The *W-MultilayerPerceptron* algorithm behaves in the worst way. Its predictions are quite inaccurate and its computational time is the highest one compared to the others. The decision tree-based algorithm it's a good compromise between computational time and quality of the prediction. It is the fastest algorithm (its computational time never exceeds 3 seconds) and its prediction error is between the two algorithms previously considered. From these graphs another relevant informations can be extracted, in particular analyzing where the peaks of each curve are located. Since the horizontal axis represents time, with a granularity equal to 15 minutes, each graph gives informations about the timeslots where the algorithms behave worse. Such peaks are located in the time slots described previously in this paper (i.e., during day-time), but another slot where the prediction errors are quite high is the slot between 9 p.m. and 12 p.m. This information was not present in the analisys done before, probably because the errors where computed starting from data covering only two working days, respectively a Monday and a Tuesday in May.

Chapter 9

Temporal analysis

This chapter shows the results of an analysis of the data at different time granularities, in particular it gives information about the accuracy of the prediction of the *W-SMOreg* algorithm for each day of the week and for each slot of time in which I divided each day.

9.1 Error Temporal Analysis - provider1

The next step is to see how the prediction error is distributed through time. In order to see this I worked with two granularities: timeslots and days of the week. The division of each day in timeslots is done taking into consideration slots of three hours(i.e., each slot is made of 12 timestamps):

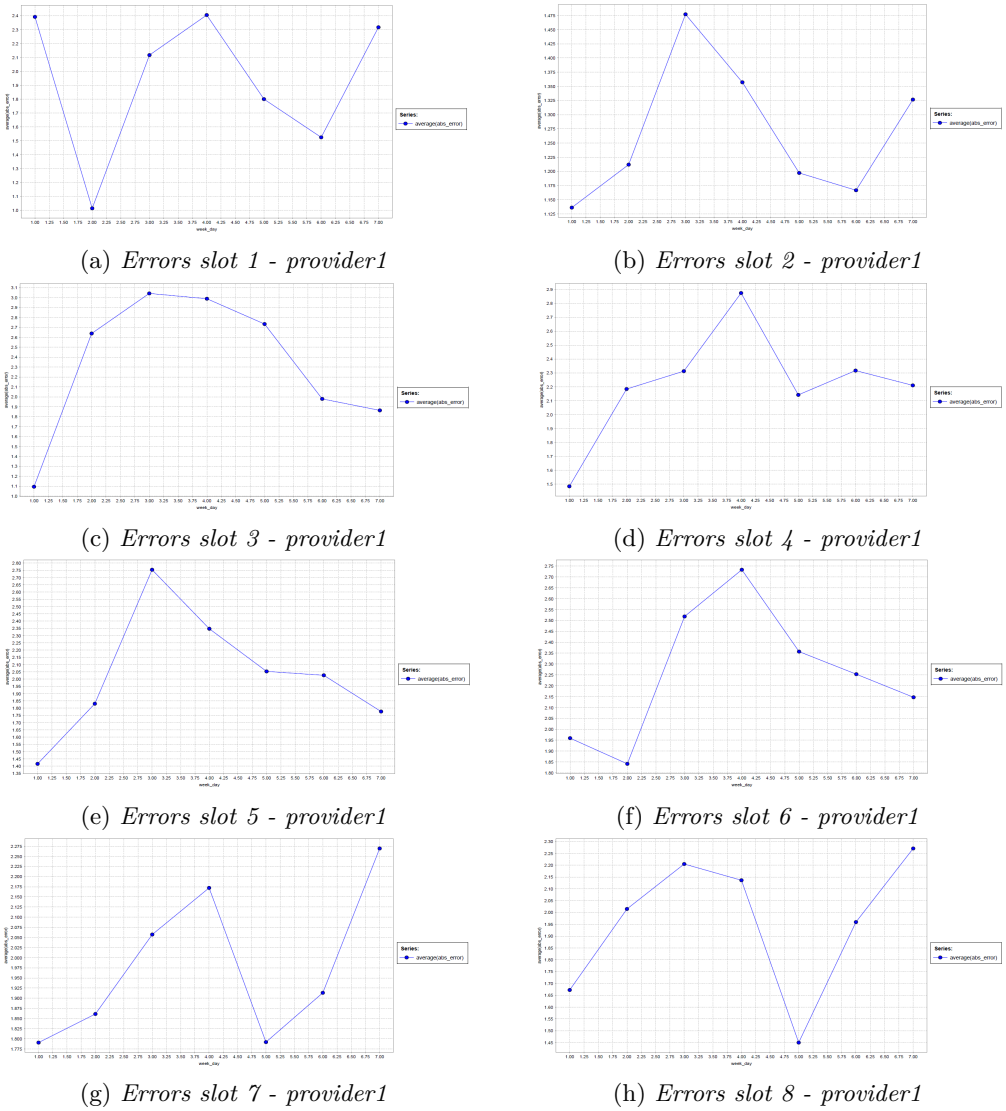
- slot **1**: from 12am to 3am;
- slot **2**: from 3am to 6am;
- slot **3**: from 6am to 9am;
- slot **4**: from 9am to 12pm;
- slot **5**: from 12pm to 3pm;
- slot **6**: from 3pm to 6pm;
- slot **7**: from 6pm to 9pm;
- slot **8**: from 9pm to 12am.

The day of the week instead is modelled as an integer number from 1 to 7, where 1 corresponds to Sunday. Since this informations were not present in the initial input dataset, each row of the input dataset was extended adding this two numerical attributes. The analysis is done taking into consideration the prediction algorithm that gave the best performances in terms of prediction error (i.e., *W-SMOreg*), considering all the results obtained so far.

The following graphs show how the error is distributed through the various slots, considering all the cells and all the days of the week. The horizontal axis represent time, and

each integer value represent a day of the week. The vertical axis represent the average absolute error, that is computed making an average on all the cells. These graphs are generated keeping constant a slot (and changing it progressively) and then aggregating by the day of the week.

All the data obtained and represented in this chapter is obtained applying the *W-SMOreg* algorithm with the parameters described before in this paper and using a test window size equal to 24, which corresponds to data covering a period of time of six hours. The horizon parameter of the *Sliding Window* operator is kept constant and equal to 8 (i.e, 2 hours).



(a) *Errors slot 1 - provider1*

(b) *Errors slot 2 - provider1*

(c) *Errors slot 3 - provider1*

(d) *Errors slot 4 - provider1*

(e) *Errors slot 5 - provider1*

(f) *Errors slot 6 - provider1*

(g) *Errors slot 7 - provider1*

(h) *Errors slot 8 - provider1*

Figure 9.1: Prediction error distribution for each slot of time

The following graph instead shows the performance of the *W-SMOreg* algorithm, making an average of the error on all the days of the week and all the cells.

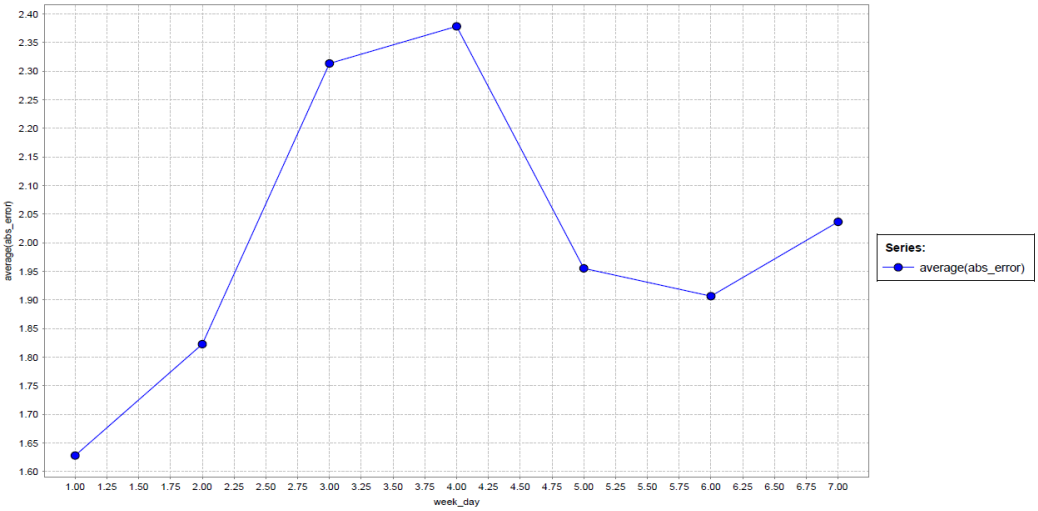


Figure 9.2: Prediction error distribution for each day of the week, for all slots

This graph shows that the days in which the algorithm behaves worst are Wednesday and Tuesday, with an average error respectively equal to 2,30 and 2,355. The days in which the prediction is more accurate are instead Sunday and Monday, with an error respectively equal to 1,655 and 1,855.

9.1.1 Analysis by slot - provider1

Here are the considerations that can be taken from the graphs in the previous paragraph, for each slot:

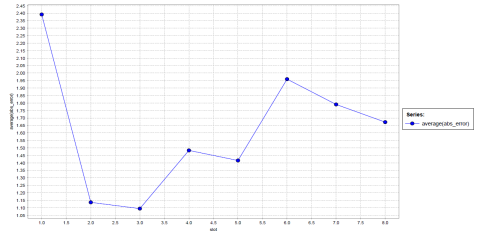
- slot 1: the average error has a range that goes (approximatively) from 1 to 2,4. The highest values are reached for Sunday, Wednesday and Saturday. Probably the peaks during weekend are related to the frequent use of the service during night-time, to move from peripheral areas to the central area, in which all the considered cells are located;

- slot 2: the average error has a range that goes(approximatively) from 1,125 to 1,475. During the first three days of the week the error assumes its highest values, reaching a peak during Tuesday and reaching a minimum during Friday. The lowest values are reached on Sunday. This is the slot for which the prediction error is lowest and for which the error range is less wide;

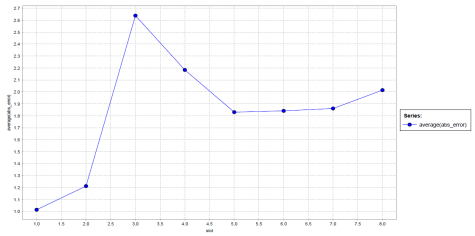
- slot 3: the average error has a range that goes (approximatively) from 1,1 to 3. The behaviour is quite similar to the previous slot. Infact during working days, the error assumes values always above 2. The lowest peaks are instead located, as the previous slot, during weekends, in particular on Sunday;

- slot 4: the average error has a range that goes (approximatively) from 1,5 to 2,9. The highest peak is reached on Wednesday and the lowest peaks are reached during weekends and on Thursday;
- slot 5: the range is from 1,4 to 2,75. The highest peak is reached on Tuesday and the lowest peak is on Sunday;
- slot 6: the range is from 1,85 to 2,75. Its highest values are located between Tuesday and Thursday, with a peak on Wednesday.
- slot 7: the range is from 1,83 to 2,275. During working days the error increases until it reaches a peak on Wednesday, then decreases significatively on Thursday and then increases again until it reaches its maximum value on Saturday;
- slot 8: the range is from 1,45 to 2,25. During working days the error is quite high on Monday, Tuesday and Wednesday. It decreases for the data relative to Thursdays and then rises up again until it reaches a maximum on Saturday.

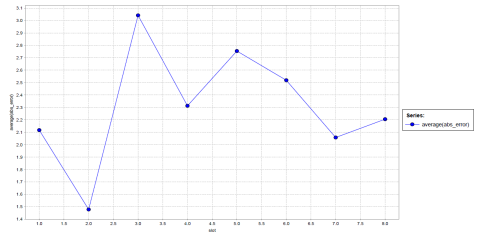
The following graphs instead show how the error is distributed through the various days of the week, considering all the cells and all the slots in which I chose to divide the 24 hours of each day. The horizontal axis represents time, and each integer value represents a slot. The vertical axis represent the average absolute error, that is computed making an average on all the cells. These graphs are generated keeping constant a day of the week (and changing it progressively) and then aggregating by the slot of time:



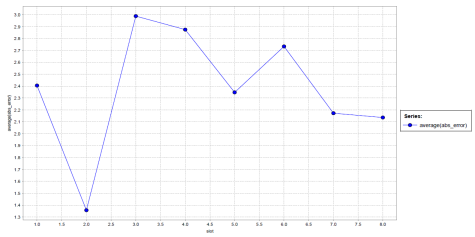
(a) *Sunday errors - provider1*



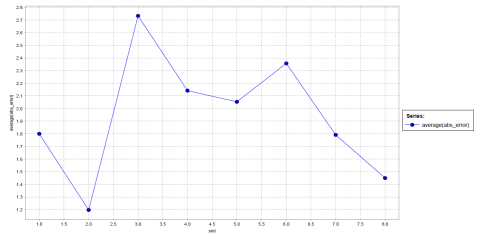
(b) *Monday errors - provider1*



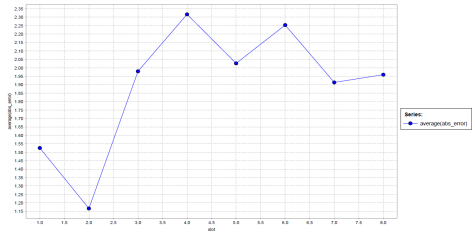
(c) *Tuesday errors - provider1*



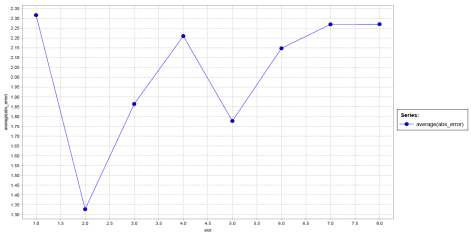
(d) *Wednesday errors - provider1*



(e) *Thursday errors - provider1*



(f) *Friday errors- provider1*



(g) *Saturday errors - provider1*

Figure 9.3: Prediction error distribution for each day of the week - provider1

The graph below instead summarizes the informations about the algorithm performance for each slot, considering making an average on all days of the week and all the cells.

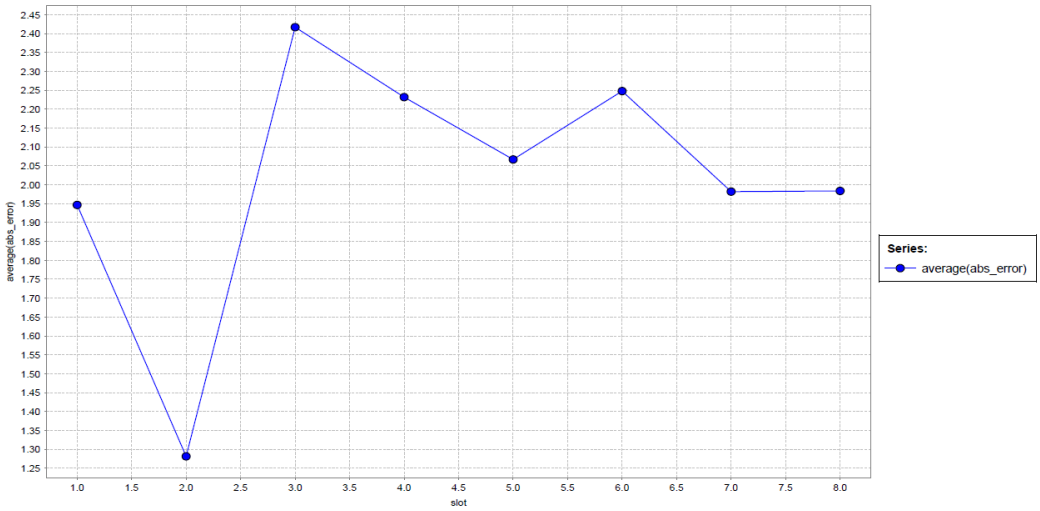


Figure 9.4: Prediction error distribution for each slot, for all days

As the graph shows the average error values are quite similar, except for slot 2, for which the error is lowest and it is equal to 1,30 approximatively. For all the other slots the error assumes values in the range between 1,95 and 2,40, so it can be approximatively be considered equal to 2.

9.1.2 Analysis by day of the week - provider1

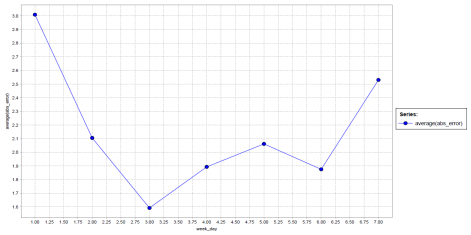
Here are the considerations that can be taken from the graphs in the previous paragraph, for each day of the week:

- Sunday: range between 1,10 and 2,40. The highest peak is located on slot 1 (between 12am and 3am) and then the error decreases for data related to slot 2 and goes up for all the afternoon slots up to slot 6;
- Monday: range between 1 and 1,65. The error during slot 1 is very low and its were the curve has its minimum. It then raises up to 2 and stays in a range between 1,80 and 2,10 until slot 7;
- Tuesday: range between 1,5 and 3,1. The highest peaks are located on slot 3 (between 9am and 12pm) and slot 5 (between 3pm and 6pm). The minimum is instead reached for slot 2 (between 6am and 9am);
- Wednesday: range between 1,35 and 3. The curve has a behaviour similar to the previous one, with a significant difference on slot 4;
- Thursday : range between 1,2 and 2,7. The behaviour is similar to the previous curve. There is just a small decrease of the prediction quality during slot 1 and a better precision on slot 8;

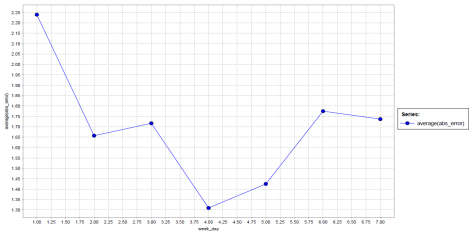
- Friday: range between 1,15 and 2,3. The shape of the curve is very similar to the one representative of Wednesday. Between slot 3 and slot the error keeps itself in a range between 2 and 2,30;
- Saturday: range between 1,35 and 2,30. In this case we have an evolution of the error quite similar to the one observed for Sunday. The highest peak is on slot 1 (between 12am and 3am). It differes significatively from Sunday curve between slot 6 and 8, where there is a less prediction accuracy that decreases progressively.

9.2 Error Temporal Analysis - provider2

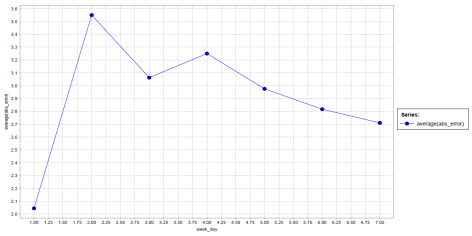
The analysis described before is done for the data collected for *provider2*. The granularity of time, the methods and the conditions are exactly the same.



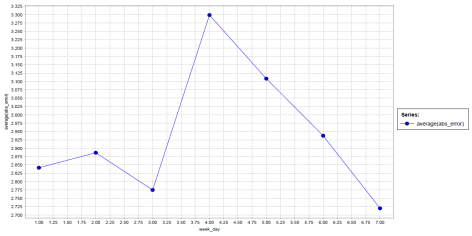
(a) *Errors slot 1 - provider2*



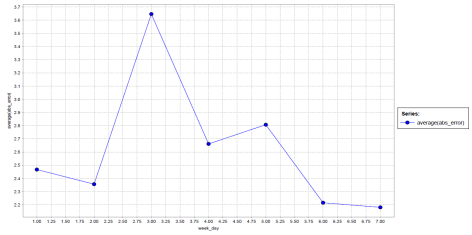
(b) *Errors slot 2 - provider2*



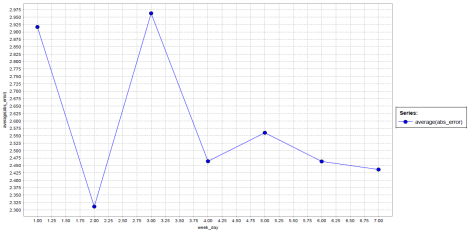
(c) *Errors slot 3 - provider2*



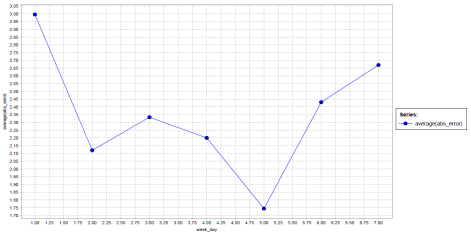
(d) *Errors slot 4 - provider2*



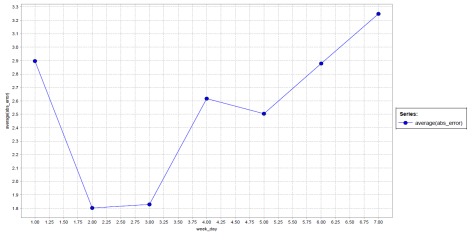
(e) *Errors slot 5 - provider2*



(f) *Errors slot 6 - provider2*

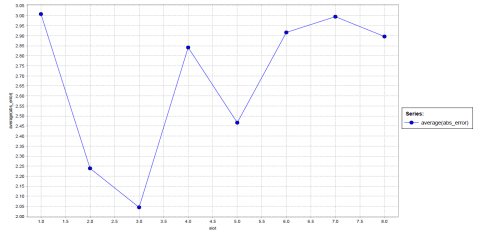


(g) *Errors slot 7 - provider2*

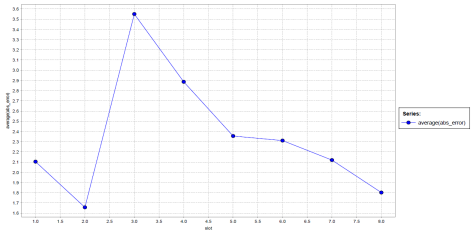


(h) *Errors slot 8 - provider2*

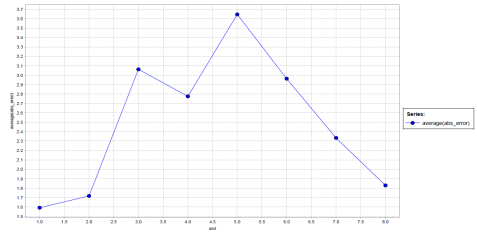
Figure 9.5: Prediction error distribution for each slot of time - provider2



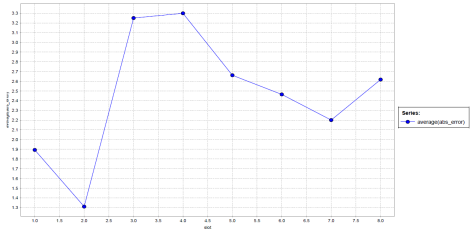
(a) *Sunday errors - provider2*



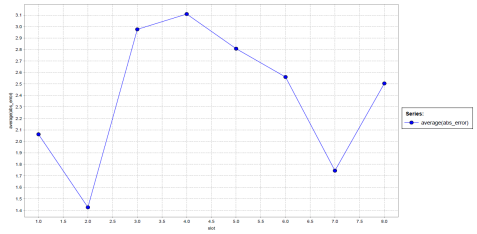
(b) *Monday errors - provider2*



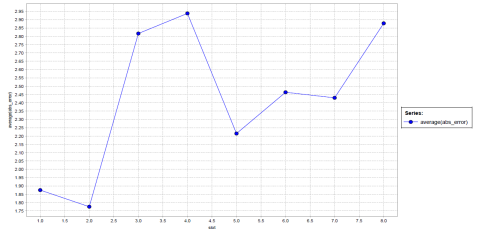
(c) *Tuesday errors - provider2*



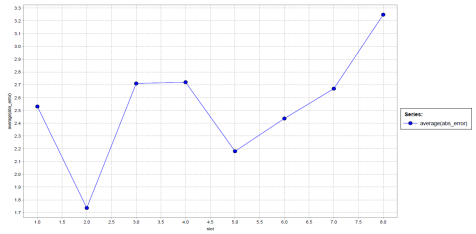
(d) *Wednesday errors - provider2*



(e) *Thursday errors - provider2*



(f) *Friday errors- provider2*



(g) *Saturday errors - provider2*

Figure 9.6: Prediction error distribution for each day of the week - provider2

The two graphs below summarize the previous graphs, showing the evolution of the error through the slots and the days of week. In the first case I just computed the error grouping by the *slot* attribute, in the second case instead I grouped data by the *week_day* attribute.

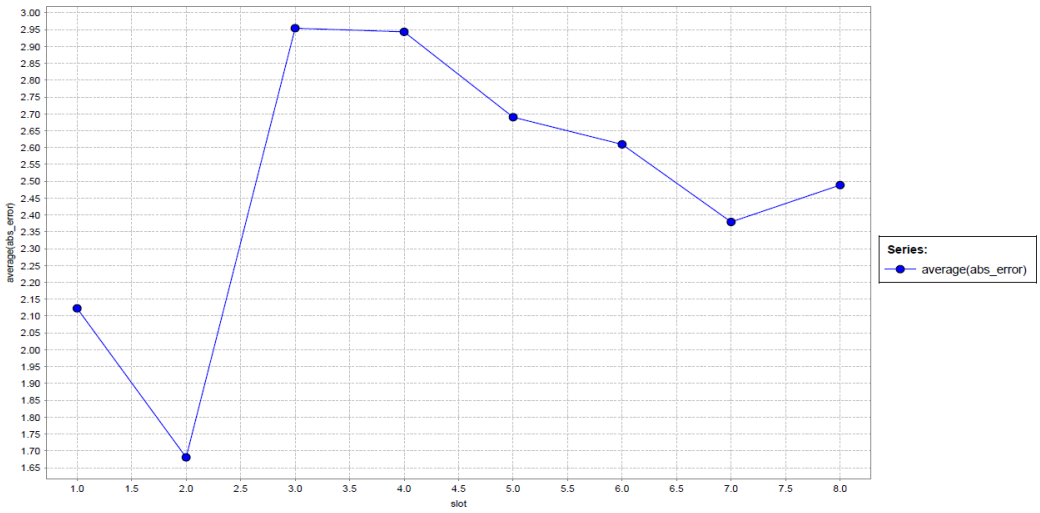


Figure 9.7: Prediction error distribution for each slot, all days of the week - provider2

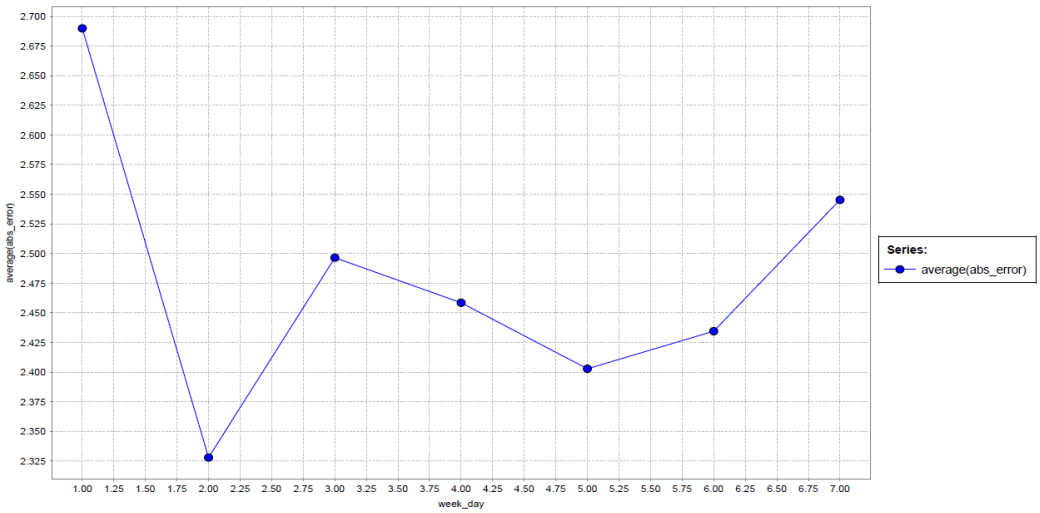


Figure 9.8: Prediction error distribution for each day, all slots - provider2

9.2.1 Analysis by slot - provider2

Here are the considerations that can be taken from the graphs in the previous paragraph, for each slot:

- slot 1: range between 2,05 and 3. The error is lowest during working days, in particular during Tuesday, and it assumes its hgihest values during weekends;
- slot 2: range between 1,65 and 3,55. The curve decreases from Sunday to Wednesday and then increases progressively from Wednesday to Saturday;
- slot 3: range between 2,15 and 3,55. The shape of the curve is very different from the previous ones. The error is lowest during Sunday and has a peak on Monday. Starting from Monday the error decreases progressivley until Saturday;
- slot 4: range between 2,725 and 3,30. The behaviour is quite different from the previous slots. The highest peak is on Wednesday. Starting from this day the curve decreases until Saturday, where we have the lowest peak of error.
- slot 5: range between 2,2 and 3,65. The highest error value is obtained for Tuesday and then the error goes down until Saturday, passing through a significant peak on Thursday;
- slot 6: range between 2,3 and 2,975. Highest peaks located on Sunday and Tuesday and lowest peak on Monday;
- slot 7: range between 1,75 and 3. The shape of the curve is quite similar to the one representative of slot 1. The error is highest on Sunday and Saturday and it is lowest on Thursday. For the other days we have very close error values;
- slot 8: range between 1,8 and 3,25. Sunday, Thursday, Friday and Saturday are the days that correspond to the lowest accuracy of prediction. From Thursday to Friday the error increases progressively until it reaches its maximum on Saturday.

Considering the error per slot of time, the shape of the curve is quite similar to the one representing the same informations for provider1. The highest peaks are located on slot 3 and 4. The lowest peak instead is located on slot 2.

9.2.2 Analysis by day of the week - provider2

Here are the considerations that can be taken from the graphs in the previous paragraph, for each day of the week:

- Sunday: range between 2,05 and 3. During daytime the prediction error is quite low for slot 2 and 3, then increases for slot 4 and keeps itself in a range between 2,9 and 3 for the slots from 6 to 8. The highest peak is on slot 1, during night-time;
- Monday: range between 1,65 and 3,55. The shape is quite opposite from the previous one. The prediction error during daytime is higher than the one measured during Sunday. Starting from the highest peak, located on slot 3, the error decreases progressively until slot 8. The minimum error value is reached for slot 2;
- Tuesday: range between 1,6 and 3,65. The shape is similar to the one of the previously described curve. The main difference is an higher value of error for slot 5, where the maximum peak is located;

- Wednesday: range between 1,3 and 3,3. This case is still similar to the ones previously considered, except for an higher value during slot 4, where we have a maximum. The error is also higher during night-time;
- Thursday: range between 1,4 and 3,1. We have the typical trend of the other working days, but an higher value for slot 1;
- Friday: range between 1,8 and 2,95. The shape of the curve is not significantly different from the other concerning working days. However the gap between slot 4 and slot 5 is quite high, and there is also an high peak of error for slot 8;
- Saturday: range between 1,7 and 3,25. The evolution of the error through time is similar to the one obtained for Sunday. The main difference is an higher distribution in the morning, in particular during slot 3. Starting from slot 5, the error increases progressively until it reaches its maximum value for slot 8.

Things change significatively between the two service providers when the error per day is analyzed. The highest prediction errors are made on weekends, in particular on Sunday. The lowest peak instead is on the same day, which is Monday. For the other working days the error keeps itself in the range between 2,4 and 2,5.

Chapter 10

Seattle data analysis

This chapter describes the analyses made using as in input the data of another city, Seattle. Sections 1 and 2 to describe the input dataset and its attributes and the pre-processing of the data. Instead sections from 4 to 7 describe the operators used in the process of analysis and show graphically the accuracy of the regression algorithm, comparing it with another regression technique. Finally the last section illustrates an analysis made at different granularities of time, in particular day of the week and slot of time.

10.1 Dataset

The dataset used in this analysis is made up of data collected through car2go APIs from car2go servers. It contains data concerning the positions of the vehicles monitored every 10 minutes for a period that goes from 07/08/2016 to 12/08/2016 [7]. The area related to the observation is Seattle and its metropolitan area.

10.1.1 Size and attributes

The dataset contains more than 450,000 rows. The most relevant attributes of each record are:

- *vin*: vehicle identification number
- *datetime*: timestamp of the observation
- *longitude*: longitude of the car's position
- *latitude* latitude of the car's position

The tuples of the dataset contain two more attributes that are not relevant for the scope of the analysis: a progressive number associated to each tuple and an attribute whose scope is not clear.

10.2 Pre-processing and vehicles per cell computation

Since the initial dataset contained the positions of the vehicles in terms of latitude and longitude and I was interested in locating them in one of the cells in which the city was divided I had to preprocess the dataset before applying the regression algorithm. The first step was to generate a CSV file in which the positions of each vehicle are located in a specific cell and then compute from this file the number of vehicles present in each cell in a given day and timestamp. Of course if I had generated the final output file at his step it would not be complete, since are generated no tuples if a cell has no available vehicles inside it in a given couple (day,timestamp). So I had for each missing couple (day,timestamp) a record that indicates the absence of vehicles in the cell in that specific period of time.

10.3 Spatial granularity

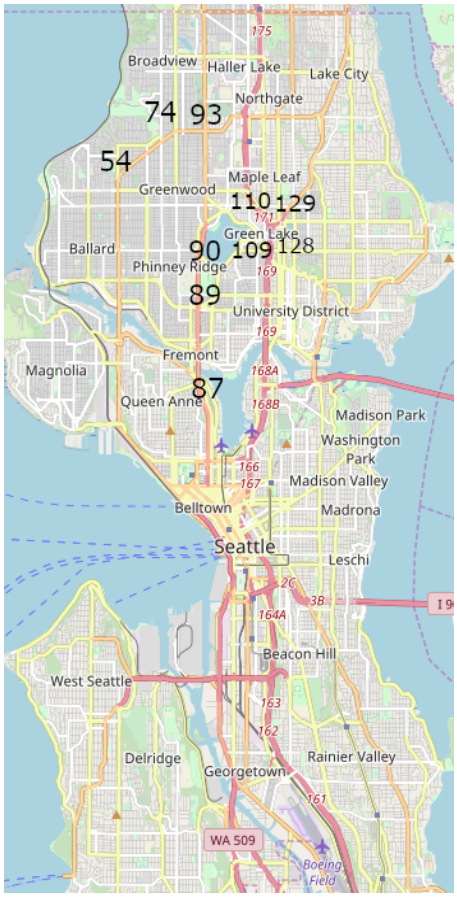


Figure 10.1: Seattle map with chosen cells

The spatial division of the area object of this analysis is quite the same of the one applied to the *use case 1* city. Seattle was divided in cells, each one covering an area of $1km^2$. Since Seattle has an extensions of $217km^2$ and the *use case 1* city instead covers $130km^2$, the number of cells in which the city is divided is larger. However, not all these cells were considered during this work. I chose to analyze only 11 cells out of 200, selecting the ones with the highest variance in terms of number of available vehicles in the cell during the day. Such cells are all located in the central area of Seattle.

10.4 SVM regression

Since in the previous chapter I found out that the SVM-based regression algorithm behaves best when trying to predict the number of cars available for rental in a cell in the future, the *W-SMOreg* algorithm was the algorithm I chose to perform regression in this analysis from now on. All the algorithm parameters are kept unchanged. As I did for the analysis described in the previous chapters, the label (i.e, the value to predict) is the number of cars that can be rented inside a cell in the future. However some parameters of the operators were changed, since the sampling time at which the positions of the vehicles are monitored is no more 15 minutes but 10 minutes.

10.4.1 Windowing operator

The *Windowing* operator acts as described before, but the window size is set to 4 instead of 3. This means that, for each attribute, I kept the four previous values that the attributes assume in the four records before the current one. This also means that each record has a vision in the past wide 40 minutes (for the data relative to the *use case 1* city the window was 45 minutes). Of course the attribute that stores the number of available vehicles makes an exception, since the horizon parameter is set to predict the *label* attribute using not the labels immediately before the one considered, but locating them in a period of time in the past. In particular three different horizon values were used, in order to see how the prediction is efficient changing the distance between the values used to make a prediction and the moment of the prediction itself:

- $horizon = 6$: setting the horizon to 6 means that to make a prediction of the number of cars I use the number of cars available 60, 70, 80 and 90 minutes ago;
- $horizon = 9$: setting the horizon to 9 means that to make a prediction of the number of cars I use the number of cars available 90, 100, 110 and 120 minutes ago;
- $horizon = 12$: setting the horizon to 12 means that to make a prediction of the number of cars I use the number of cars available 120, 130, 140 and 150 minutes ago.

10.4.2 Sliding Window operator

Analyzing the *use case 1* dataset the validation using a sliding window seemed to be the best way to train and test the regression algorithm. I chose to set the training window width parameter to 288, in order to train the regression algorithm with the data covering a period of time equal to two days. The test set instead is set to 144, meaning that, using

the data inside the training set, I made predictions for the data covering a period equal to one day. The step size parameter is kept unchanged, so the training window and the test window step forward of an amount of records equal to the test window size, until all the dataset is covered. The horizon value parameter is the same set for the *Windowing* operator, so it assumes three different values according to the one set for the *Windowing* operator at that run.

10.5 Prediction error computation

The *predictionEvaluation* part of the framework, once the *W-SMOreg* algorithm generated all the CSV files containing all the predictions for each cell and for all the horizon values, computes the prediction errors using the following formula:

$$abs_error = |rounded\_predicted\_value - actual\_value|$$

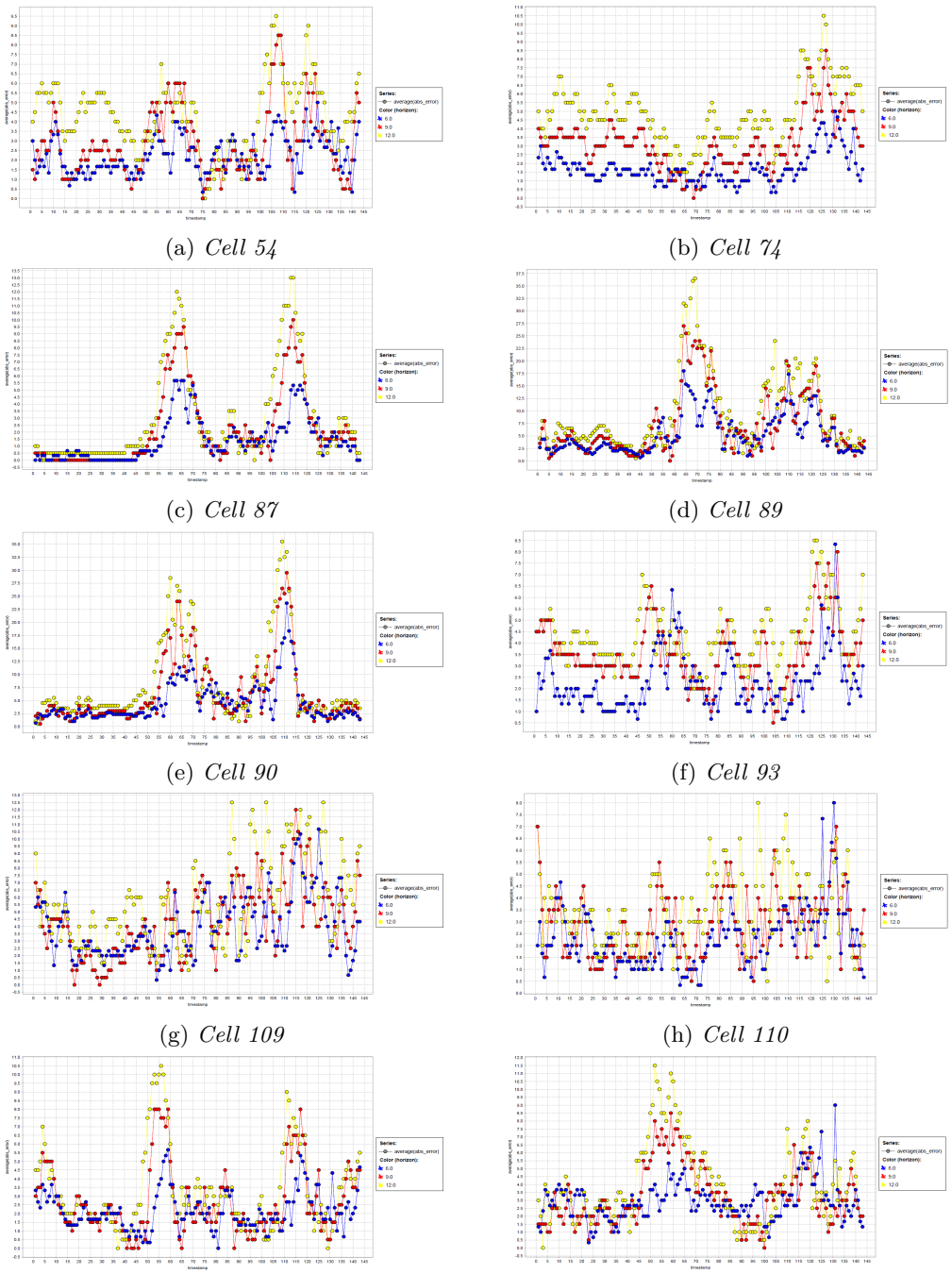
The output file generated at this step has the following attributes:

- *cell*: the ID of the cell (from 0 to 199);
- *horizon*: the integer value of the horizon parameter that was set for both the *Windowing* and *Sliding Window* operator (possible values: 6, 9 and 12);
- *day*: the integer value of the day within the month;
- *timestamp*: the integer value of the timestamp, that assumes values from 1 to 143;
- *abs_error*: the error computed applying the formula described above.

So far no attributes concerning month, day of the week or timeslot are present. The first one is absent because it is constant, since data is relative to five days of August 2016. The last two attributes will be added later, in order to make a temporal analysis by day of the week and slot of time.

10.6 Prediction error with different horizons

The graphs below show how the error made by the applied regression algorithm evolves through time, where time has the same granularity of the sampling time (i.e, 10 minutes). Each graph is relative to a cell and shows three different curves, each one representing a different value of the horizon of the prediction. The error represented in these graphs is averaged on the timestamp attribute.



(a) *Cell 54* (b) *Cell 74*

(c) *Cell 87* (d) *Cell 89*

(e) *Cell 90* (f) *Cell 93*

(g) *Cell 109* (h) *Cell 110*

(i) *Cell 128* (j) *Cell 129*

Figure 10.2: Prediction error distribution for each cell

From the graphs above two different behaviours among the different areas of the city can be deducted. Such behaviours can help dividing the considered cells in two groups:

- a group made up of cell 87,89 and 90. These cells have a similar distribution of the error and peaks located approximatively at the same timestamps. At the beginning of the day, from 12am to 8:30am, the error is lower. For cell 87 it is close to 0, meaning probably that the number of available vehicles is constant during this period of time. The same situation is present at the end of the day, from 8:30pm to 12am. The error reaches his maximum values in two different slots of time: 8:30am-12:30pm and 5:30pm-8:30pm. However the values assumed are quite different: for cell 89 and cell 88 the values during the second slot are very high and go above 30 for a prediction horizon equal to 12. This means that during this moment of the day we have a large amount of vehicles left in the cells that can not be predicted for high values of the prediction horizon. Anyway also for the smallest values of the prediction horizon the error can be considered high in the same way;
- a group made up of all the other cells but the ones belonging to the group described above. For all these cells the error is distributed more uniformly among the day and has a smaller range of values than the other group. Anyway the slots of time where the error is higher seem to be the same we have for the other group.

Performances are quite different for different values of the prediction horizon. As it was expected as the horizon value increases performance gets worse:

- horizon equal to 12 : it is the value of the horizon for which the error is highest. The gap between the curves representing the other horizon values becomes significant in particular during the slots mentioned above: 8:30am-12:30pm and 5:30pm-8:30pm;
- horizon equal to 9 : the curve representing this value in most of the cases is between the other two curves. However in many cases its error value have a gap between the blue curve that is smaller than the gap between the yellow curve. In some cases it is also the horizon value for which the prediction is most accurate.
- horizon equal to 6 : for this horizon value the regression algorithm behaves best in most of the cases, but it is also the value for which the regression algorithm needs more time to be trained and tested.

For all the analysis described after this one, the horizon value of the *Windowing* and *Sliding Window* operators is set to 9. For the reasons described above it seems to be a good compromise between computational time and efficiency of the prediction.

10.7 SVM and Baselines comparison

The next step is to see how the performance of the SVM-based algorithm differ from the one of the Baseline algorithm, which is a prediction algorithm that takes as a prediction the label at a specified time in the past, without considering all the other attributes. It can be seen as a sort of naive prediction algorithm, that says that at a specific timestamp the number of predicted available cars is equal to the number of cars available x timestamps ago. During these analysis three values of x were considered:

- 6: the prediction is equal to the number of cars available an hour ago;
- 9: the prediction is equal to the number of cars available an hour and a half ago;
- 12: the prediction is equal to the number of cars available 2 hours ago.

The graph below shows the distribution of the error through time of 4 different algorithms:

- *W-SMOreg*
- *baseline_6*: baseline algorithm with horizon parameter equal to 6;
- *baseline_9*: baseline algorithm with horizon parameter equal to 9;
- *baseline_12*: baseline algorithm with horizon parameter equal to 12.

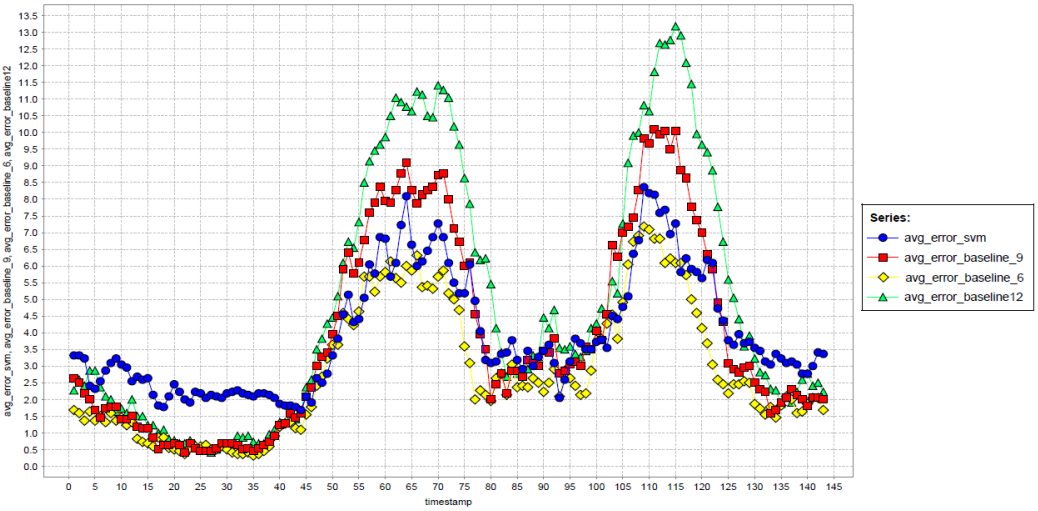


Figure 10.3: Comparison between W-SMOreg and 3 different baselines

The behaviour of the four error curves can be considered approximatively the same, with highest values located in the same range of timestamp values. For most of the time the SVM error stays between *baseline_6* and *baseline_9* error, except for two timestamp ranges where it assumes the highest error values among all the algorithms:

- 1-45, that correspond to 12am-7:30am hour slot;
- 130-143, that correspond to 9:30pm-12am hour slot.

The error range of the regression algorithm is approximatively from 2 to 8. The highest values are reached as usual in the same slots mentioned in the previous section.

10.8 Temporal analysis

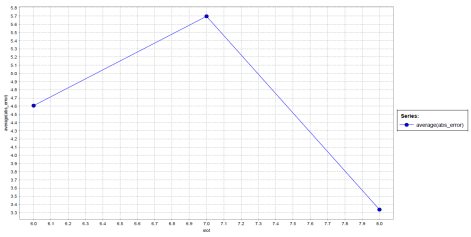
The next step is to see how the *W-SMOreg* algorithm behaves for each day of the week and for each slot of time. Since data cover only five days, there is no data relative to Sunday, Monday, Friday and Saturday. That is because of two reasons:

- the regression algorithm uses Sunday and Monday data to train, since the training window covers two days data;
- the initial input dataset goes from approximatively 2:30pm of a Sunday to 6pm of a Friday.

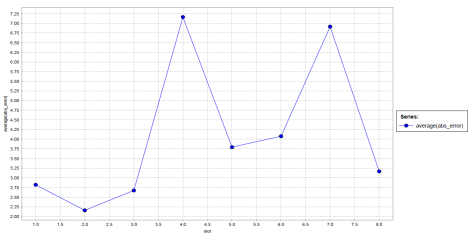
For the same reasons the error distribution of each day through the various slots of time is not complete, since some values are missing. The division of the day into slots and the enumeration of the days of the week are the same as used in the analysis of the car2go dataset concerning *use case 1*.

10.8.1 Analysis by day of the week

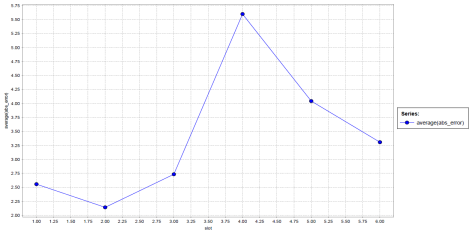
Each graph below puts in evidence how the error is distributed through the 8 slots of time in which the day was divided, for each day of the week. A single graph is relative to a single day. The error in this case is computed by keeping constant the *week_day* attribute, grouping data by the *timeslot* attribute and computing the average absolute error.



(a) *Tuesday*



(b) *Wednesday*



(c) *Thursday*

Figure 10.4: Error distribution through the slots for each day of the week

The following considerations can be made by observing these graphs, for each day:

- **Tuesday**: the error assumes values that go from approximatively 3.3 to 5.7 . The highest peak is reached for slot 7 (period of time between 5pm and 9pm). The lowest peak is instead reached at the end of the day;

- **Wednesday**: the error assumes values that go from 2.25 to 7.1 . The highest peak is reached for slot 4 (between 12pm and 3pm), but also the value assumed for slot 7 is quite high. The lowest peak is reached for slot 2 (between 3am and 6am), but also for the previous and the following slot is quite low;

- **Thursday**: the error assumes values that go from 2.10 to 5.55 . The highest peak is reached for slot 4 (between 12pm and 3pm). The lowest error values are located in the first three slots, showing a behaviour similar to the Wednesday data.

The graph below instead shows the distribution of the error through the various slots, considering all data.

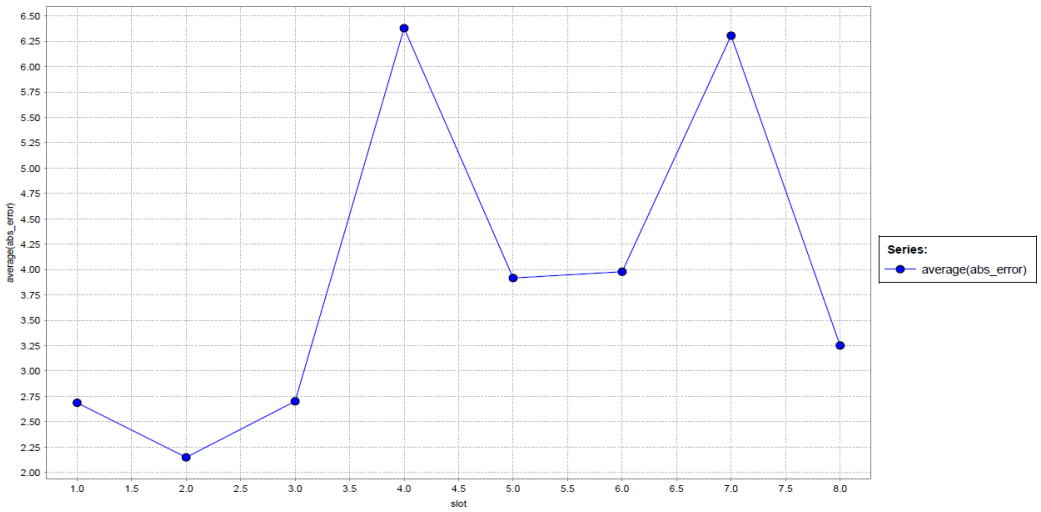


Figure 10.5: Comparison between W-SMOreg and 3 different baselines

This graph confirms the observations made before for each day:

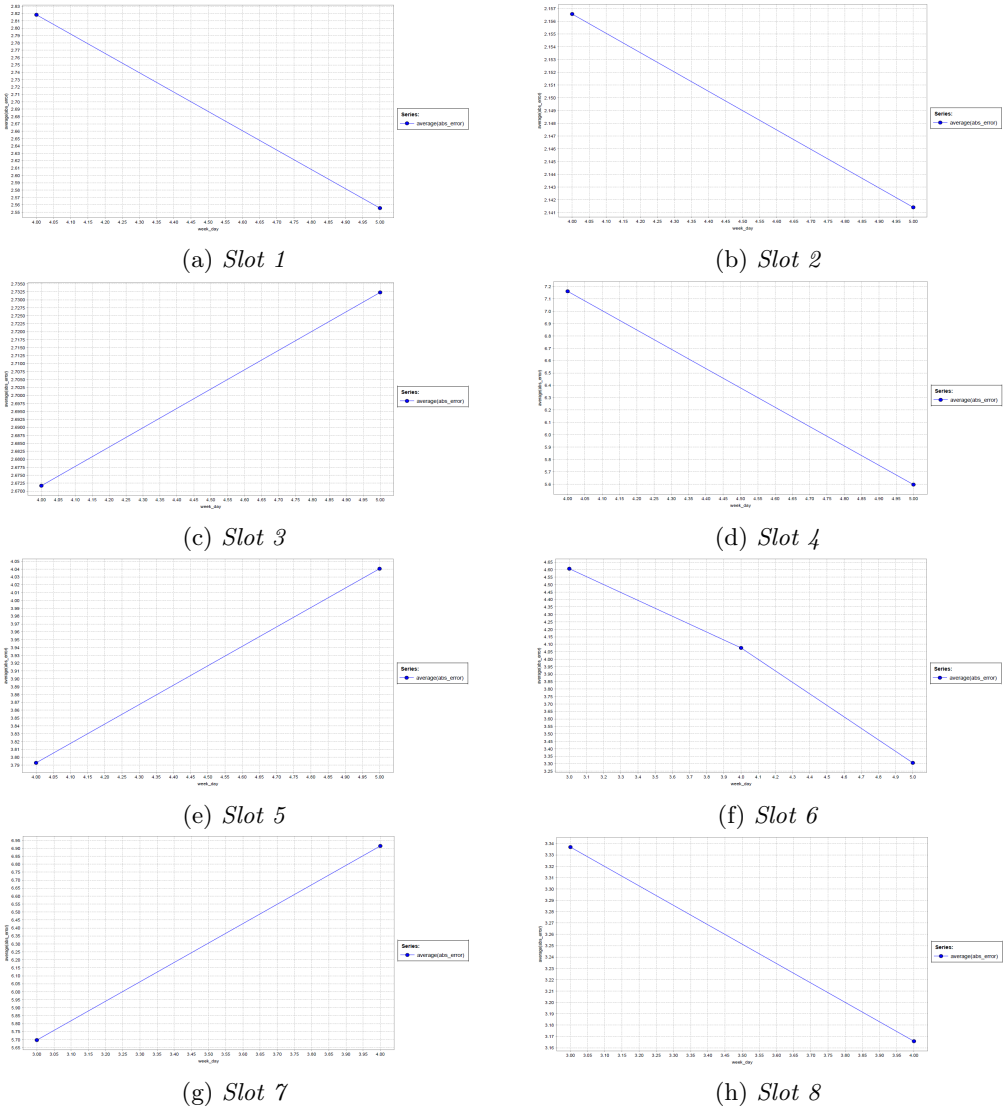
- prediction is more accurate for the periods of time between 12am and 6am and between 9pm and 12am;

- prediction is instead less accurate for slot 4 and 7.

- prediction between slot 5 and 6 (from 12pm to 6pm) has an average absolute error equal to 4.

10.8.2 Analysis by timeslot

The analysis by timeslot is definitely more exaustive than the one made by day of the week. Basically the reason is that the amount of time covered by the dataset was not wide enough to make an accurate analysis of how the error in each slot is distributed through the days of the week. That is why most of graphs have only two or three points and I obtained only data concerning Tuesday, Wednesday or Thursday.



(a) *Slot 1* (b) *Slot 2*

(c) *Slot 3* (d) *Slot 4*

(e) *Slot 5* (f) *Slot 6*

(g) *Slot 7* (h) *Slot 8*

Figure 10.6: Prediction error distribution for each timeslot, for each day

Even if the analysis is not enough exaustive, some considerations can be made for each slot:

- **slot 1**: it is one of the slots of time for which the error range is less wide. For the analyzed two days, Wednesday and Thursday, the average error is equal to 2.82 and 2.55 respectively;
- **slot2**: it is the slot for which the error range is least wide. It is equal to 2.16 for Wednesday and to 2.141 for Thursday;
- **slot 3**: the behaviour is similar to the previous two slots, except that the error is slightly higher on Thursday than on Wednesday. Anyway the difference between the two error values is very small;
- **slot 4**: together with slot 7 is the slot for which the difference between the two values is higher. The difference is 1.55 and the error is higher for Wednesday, where the error assumes its maximum value among all the curves represented in this set of graphs;
- **slot 5**: the behaviour is very similar to the one described for slot 3, except that the range is slightly higher;
- **slot 6**: it is the only slot for which we have data relative to three days and not only two. The error descreases starting from Tuesday to Thursday, with a gap between the two values equal to 1.3 . The gap between Wednesday and Tuesday is instead equal to 0.55 and the gap with Thursday is equal to 0.75;
- **slot 7**: the gap between the two values, relative to Tuesday and Wednesday is equal to 1.2. The error in this case is higher on Wednesday than on Tuesday;
- **slot 8**: in this case the behaviour is very similar to the one observed for slot 1, except that the two considered days are Tuesday and Wednesday and not Wednesday and Thursday.

Chapter 11

Portland data analysis

This chapter illustrates the analyses made using as in input the car sharing data of the city of Portland. Sections 1 and 2 to describe the input dataset,its attributes and how the city was divided in different areas. Instead sections from 3 to 6 describe a comparison between the *W-SMOreg* algorithm and the *RepTree* algorithm, the process of optimization of the parameters and the comparison between these regression algorithms and the *Baseline* approach. Last sections describe the temporal analysis of the error distribution and some experiments made considering different SVM based algorithms and different kind of training sets.

11.1 Dataset

This dataset contains data collected in the same way as the previous one. It is relative to the positions of 316 vehicles monitored almost every 15 minutes. The period of the observation is very wide as it goes from June 2012 to December 2013 [8]. This data has been presented in the PDX Tech Meetup [9] in 2015 and in the MapCamp Portland Hackathon [10] in 2014.

11.1.1 Size and attributes of the original dataset

The dataset contains more than 10 million rows. The most relevant attributes of each tuple are:

- *licence*: it is the plate of the vehicle. It was used as an ID for the vehicle;
- *date*: timestamp of the observation;
- *latitude*: latitude of the car's position;
- *longitude*: longitude of the car's position;
- *fuel*: fuel level of the car at the moment of observation.

11.1.2 Base dataset

Starting from the original input dataset data was elaborated in order to obtain the number of vehicles available in a cell at a given timestamp. Such elaboration led to the generation of a dataset with the following attributes:

- *cell*: the integer value that uniquely identifies the considered cell;
- *timestamp*: the progressive integer number that identifies the timestamp;
- *day*: the integer value that identifies the day within the month in which the number of cars is sampled;
- *month*: the string that identifies the month in which the number of cars is sampled;
- *year*: the integer value that identifies the year in which the number of cars is sampled;
- *numcars*: the number of cars that is sampled at the moment identified by the previous four attributes. It is an integer value.

11.1.3 Enriched dataset

To perform the analysis by day of the week and slot of time, two attribute were added as it was done for the datasets concerning the *use case 1* city and Seattle, using the same dimension of the slots and the same enumeration of the days of the week.

11.2 Spatial granularity

Also in this case the area of interest (Portland) was divided in cells, each one covering an area of $1km^2$. In this case a different approach was adopted to choose the cells with the most interisting informations to analyze. Once the dataset with the informations of availability of vehicles was generated, I chose to order the cells according to the variance of the *numcars* attribute. Among these cells I chose the ones with the highest variance. In particular the chosen cells were:

- cell 20;
- cell 40;
- cell 41;
- cell 60;
- cell 61;
- cell 80;
- cell 100;

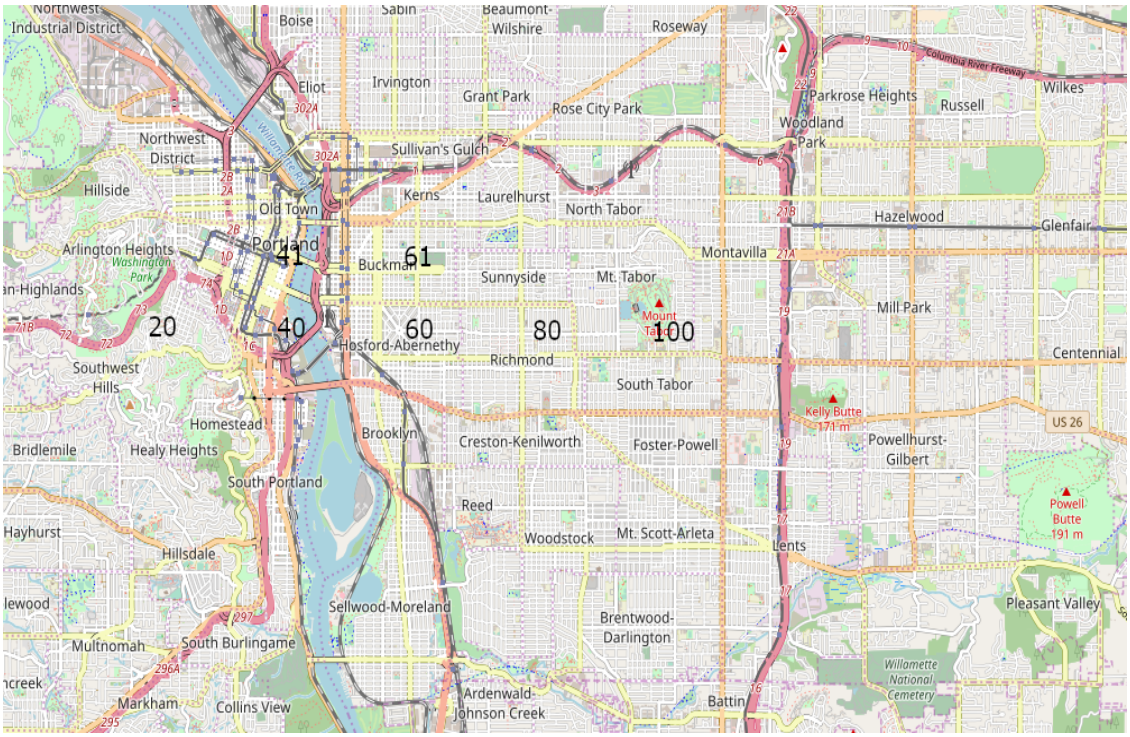


Figure 11.1: Portland map with chosen cells

11.3 SVM and RepTree regression

In the analysis of this dataset two regression algorithms were used to perform regression, and their results were compared. The first one, used for all the datasets considered in this paper is the *W-SMOreg* algorithm; the second one is instead the *RepTree* algorithm, a decision tree based algorithm that performs regression. For both the algorithms the parameters the same used for the dataset relative to the *use case 1*. In particular the horizon parameter used for the validation of the two algorithms is kept equal to 8, so the prediction is set 2 hours far from the values used to build the model. However in this case the test set width is kept constant and equal to 96, the amount of records covering a period equal to one day. The same thing was done for the training set, that is kept equal to 1344, the amount of records that cover a period of time equal to 2 weeks.

11.3.1 Windowing Operator

Also in this analysis the data is prepared by the *Windowing Operator* before running the regression algorithm. The window size is kept equal to 3 as in the previous analysis and the horizon is kept equal to 8.

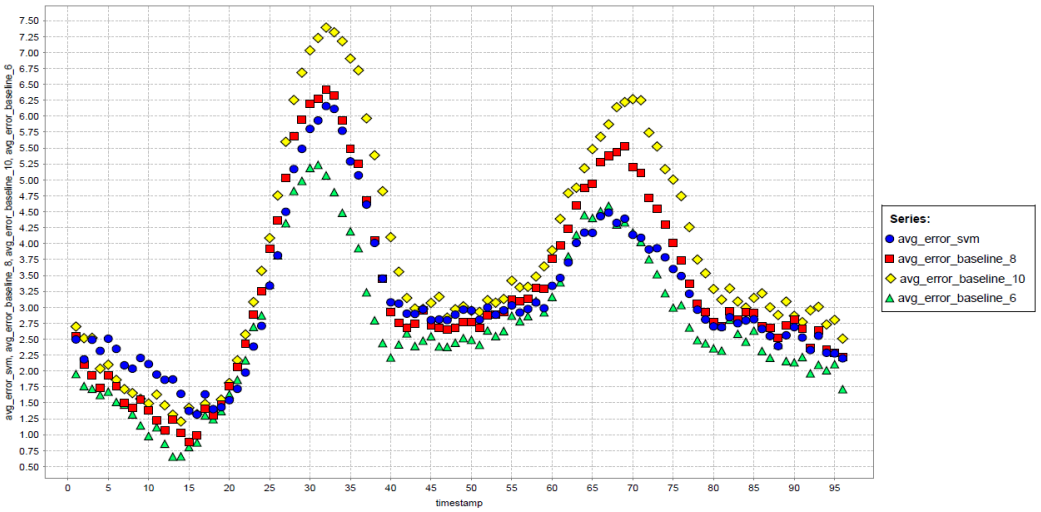


Figure 11.2: Comparison between W-SMOreg and 3 different baselines

The graphs above show the evolution fo the error through, comparing *W-SMOreg* and *RepTree* algorithms with three baselines, each one with a different prediction horizon:

- **baseline_6**: it is a baseline that makes its prediction taking the value of the *numcars* attribute an hour and a half ago;

- **baseline_8**: it is a baseline that makes its prediction taking the value of the *numcars* attribute 2 hours ago;

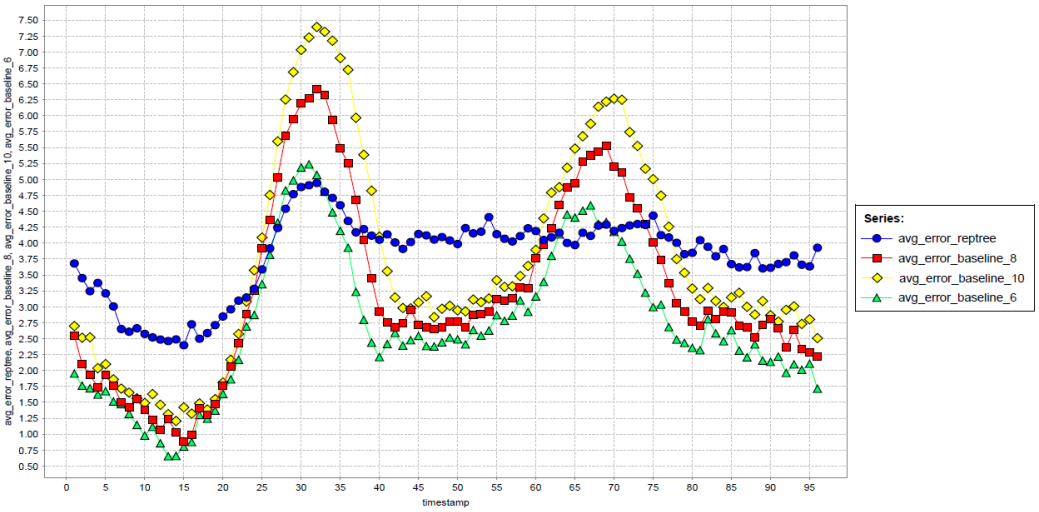


Figure 11.3: Comparison between RepTree and 3 different baselines

- **baseline_10**: it is a baseline that makes its prediction taking the value of the *num-cars* attribute 2 hours and a half ago.

The algorithms seems to behave in two different ways, in particular:

- the *W-SMOreg* algorithm seems to follow the shape of the curves representing the three different baselines, keeping itself between *baseline_6* and *baseline_8*;
- the *RepTree* algorithm instead follows the shape of the other curves only in part and the width of the values that its prediction error assumes is smaller than the one assumed by the other regression algorithm. In particular the error curve follows the shape of the other curves up to the first peak and then seems to keep itself quite stable for the other timestamp values.

11.4 Optimization of the SVM algorithm parameters

During this analysis a process of optimization of the parameter of each algorithm was run before running each SVM based regression algorithm. The *Grid Optimization (Evolutionary)* operator was used to reach this scope, trying to optimize the following parameters searching their optimal values in the following ranges:

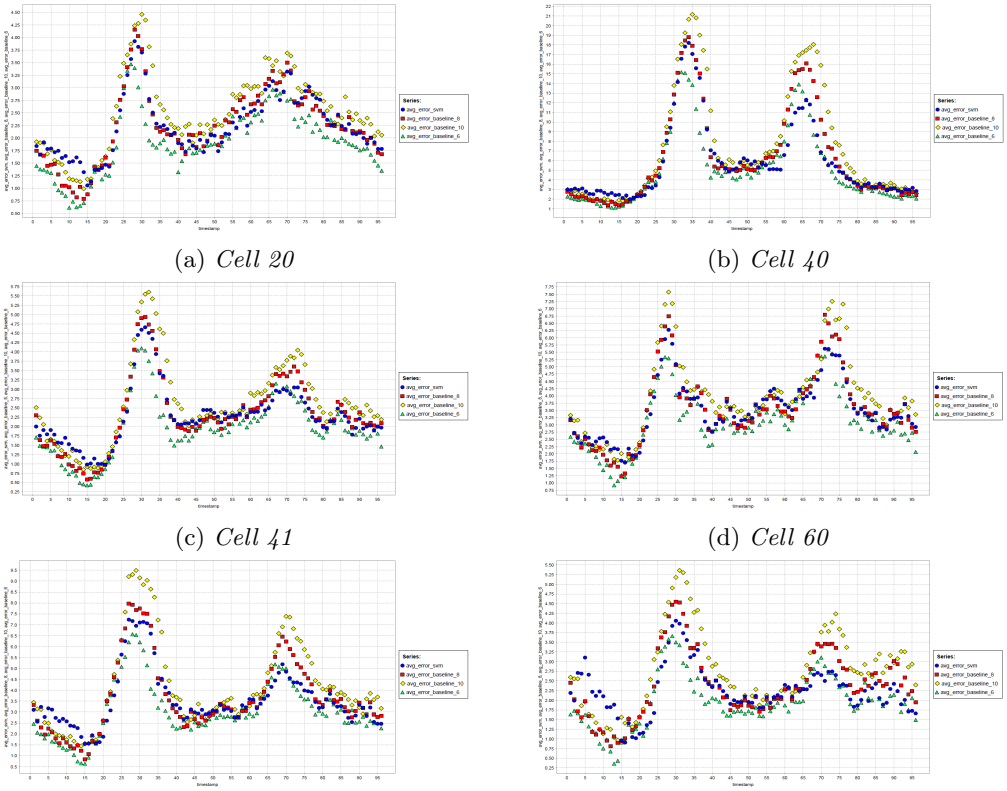
- *degree*: range from 0 to 100;
- *gamma*: range from 0 to 50;
- *C*: range from 0 to 1024;
- *nu*: range from 0 to 0,5.

The parameters of this block were kept unchanged, in particular the *max generations* attribute is kept equal to 50. This means that the optimization algorithm will stop after 50 evaluations. This parameters are set for the optimization algorithms applied on the dataset, except the *LibSVM* with *linear* kernel. In this case I had to set a smaller range for the *C* cost parameter and to reduce the *numer of max generations* to 25. I had to make this change because the computational time of the process was too long, since sometimes it exceeded the 12 hours of running time without giving any parameters result file.

Since using a test window with a width equal to 96 (i.e, 1 day) was computationally prohibitive, the test window size was changed to 192 (i.e, 2 days). The input data used to optimize the algorithm is relative to cell 40 and covers the data of July. I chose this cell because its the cell with the highest variancy in temrs of number of vehicles present inside it.

11.5 SVM and Baselines comparison for each considered cell

The graphs below shows the same comparison described in the previous section, but considering each cell separetely and considering only the *W-SMOreg* algorithm. These graphs confirm the results obtained in the previous section. The *W-SMOreg* algorithm error is for most of the timestamps between the *baseline_6* and *baseline_8* curves. Its peaks vary significatively among the cells and its highest value is reached for cell 40, which is the cells whose behaviour seems to be the most difficult to predict. However the curves represented in the graphs have a similar shape. The lowest accuracy of the prediction models is reached for almost the same timestamp ranges, which are 20-35 and 60-75.



(a) *Cell 20*

(b) *Cell 40*

(c) *Cell 41*

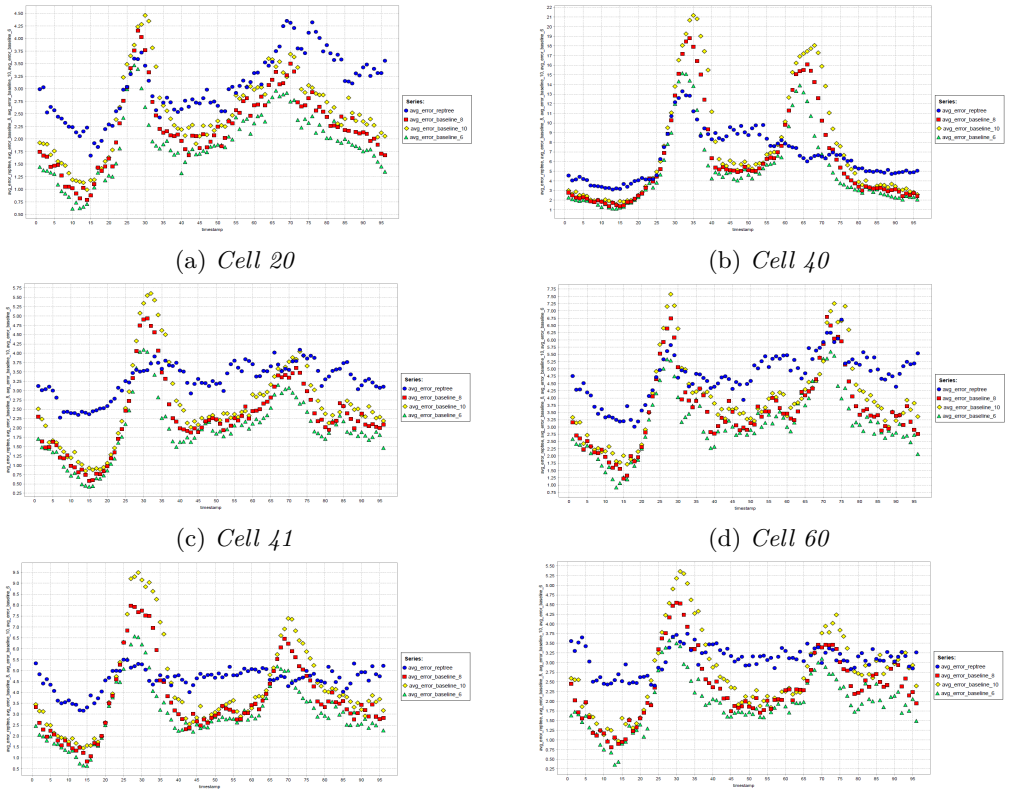
(d) *Cell 60*

(e) *Cell 80*

(f) *Cell 100*

Figure 11.4: Prediction error distribution for each cell

11.6 RepTree and Baselines comparison



(a) *Cell 20*

(b) *Cell 40*

(c) *Cell 41*

(d) *Cell 60*

(e) *Cell 80*

(f) *Cell 100*

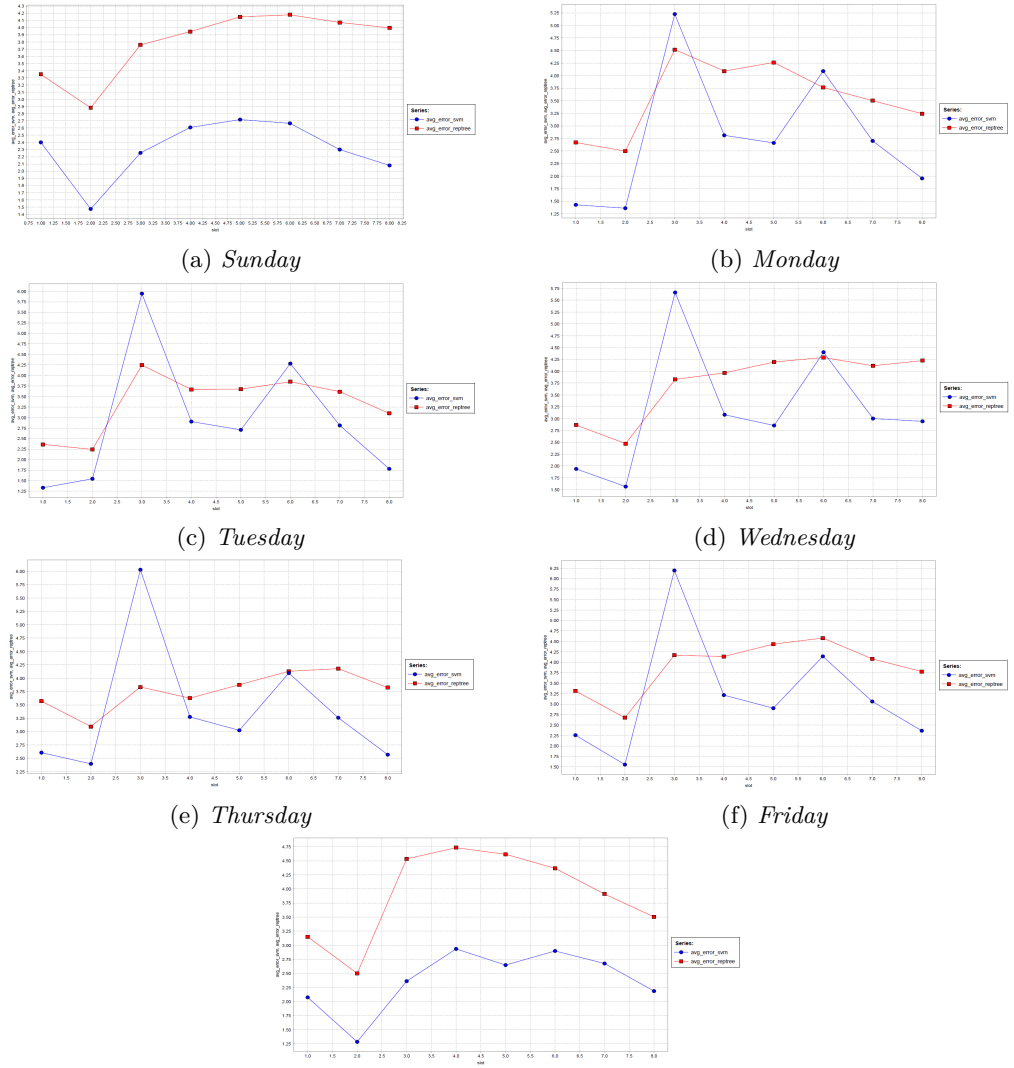
Figure 11.5: Prediction error distribution for each cell

Things change significatively between the two regression algorithm I exploited to make predicictions starting from the dataset. In general the *RepTree* algorithm has less accuracy than the *W-SMOreg* algorithm, but it behaves better in the slots in which the *W-SMOreg* algorithm and the three baselines behave worst in temrs of accuracy of the prediction. On the other slots instead it is the algorithm that behaves worst among all the considered ones. Anyway it is a very fast algorithm, so it can predict a value for the *numcars* attribute in an amount of time that is far smaller than the one required to train and test the other regression algorithm.

11.7 Temporal analysis

The temporal analysis of the data is done adding two more attributes to the input dataset, in order to assign to each timestamp a slot of time (3 hours wide) and to each day a numerical attribute identifying the day inside the week. In this way it was possibile to see in which parts of the day the regression algorithms have less accuracy and for which days the prediction of the number of cars differs most from the actual number of cars value. So, as it was done for the *use case 1* dataset, the day is divided into 8 slots, each one 3 hours wide. A numerical attribute to identify the day of the week is also added, using the same enumeration of the previous analysis (1 corresponds to Sunday, 2 corresponds to Monday and so on). Each one of the graphs compares the *W-SMOreg* algorithm with the *RepTree* algorithm.

11.7.1 Analysis by day of the week



(a) *Sunday*

(b) *Monday*

(c) *Tuesday*

(d) *Wednesday*

(e) *Thursday*

(f) *Friday*

(g) *Saturday*

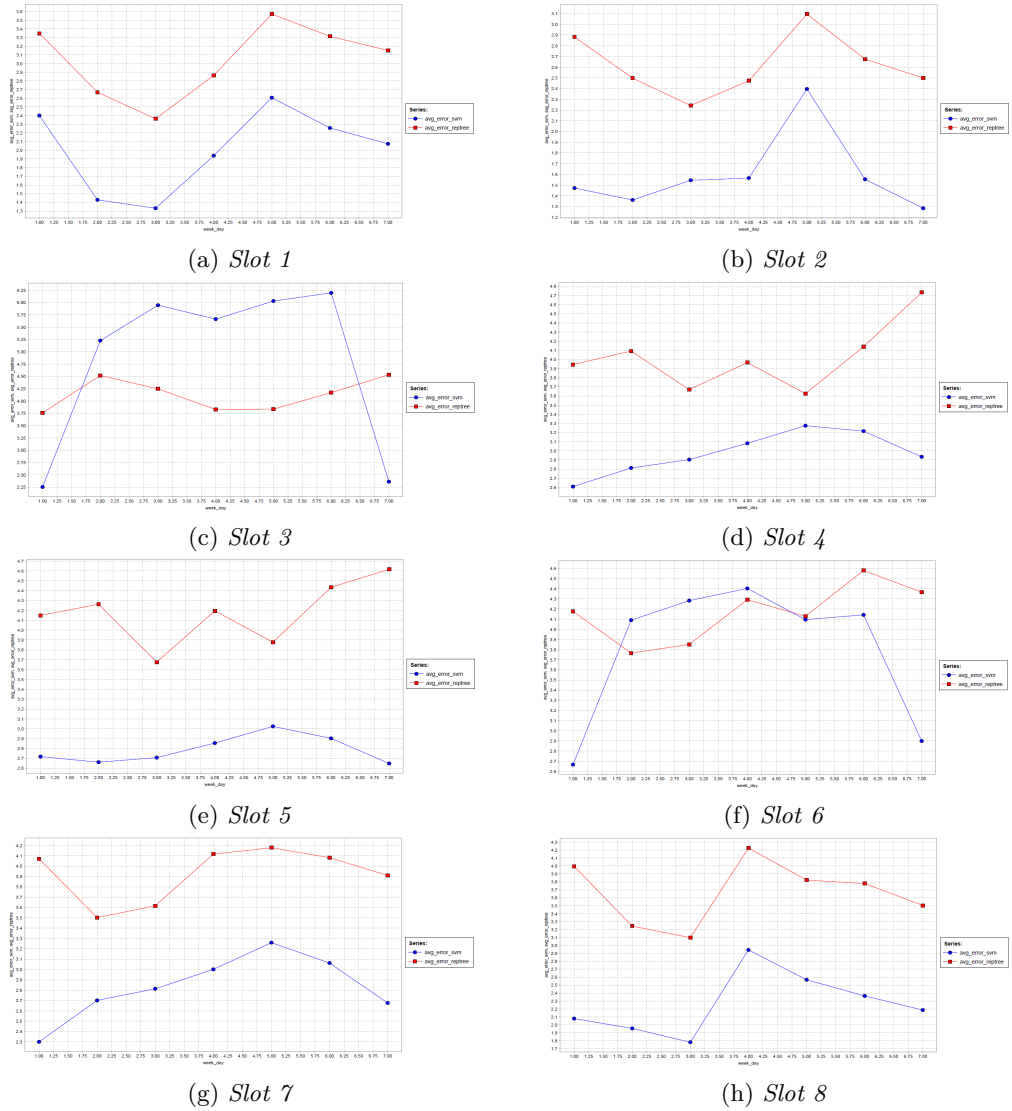
Figure 11.6: Prediction error distribution for each slot, through each day of the week

This considerations can be taken from the graphs above, for each day:

- **Sunday**: the *W-SMOreg* algorithm behaves best for all the slots in this case. The lowest accuracy for both the algorithms is reached for the slots between 3 and 6, so in a period of time that goes from 9am to 6pm;
- **Monday**: in this case the *RepTree* algorithm gives better performances than the *W-SMOreg* algorithm for the slots values in which the error is highest for the SVM based algorithm. For both the algorithms the highest peaks of error are located on slot 3 and slot 6. Lowest peaks instead are located on slot 1 and 2;

- **Tuesday**: the shapes of the curves are very similar to the one described for the previous day of the week. The peaks of error are located on the same slot values and the error ranges of values have approximatively the same width;
- **Wednesday**: the shape of the curve representing the *W-SMOreg* algorithm is quite the same described before. A small change in the *RepTree* curve can be noticed. Infact the error does not decrease starting from slot 3 as in the previous cases, but it increases slightly, reaching its maximum for slot 8;
- **Thursday**: for this day the difference between the behaviours of the two algorithms is more noticeable. For the timestamps located in the third slot, the *Reptree* algorithm is far more accurate than the *W-SMOreg* algorithm. In the other critical slot, slot 6, the behaviour of the two algorithms can be considered the same;
- **Friday**: the error evolution is very similar to the one described for the previous day. Critical slots are alway slot 3 and 6, where the two algorithms show more inaccuracy;
- **Saturday**: the graph is very similar to the one representing the error evolution on Sunday. There is only one main difference between the two graphs, that is the decrease of the error values for the *RepTree* curve starting from slot 5 to slot 8.

11.7.2 Analysis by slot



(a) *Slot 1* (b) *Slot 2*

(c) *Slot 3* (d) *Slot 4*

(e) *Slot 5* (f) *Slot 6*

(g) *Slot 7* (h) *Slot 8*

Figure 11.7: Prediction error distribution for each day of the week, through each slot

This considerations can be taken from the graphs above, for each slot of time:

- **slot 1**: the shape of the two curves is very similar, except that the *RepTree* algorithms behaves worse than the *W-SMOreg* algorithm for all the values representing the days of the week. The highest errors are made on Sunday and Thursday. On Monday and Tuesday we have instead the highest accuracy of the prediction.
- **slot 2**: also in this case the *W-SMOreg* algorithm gives better performances that the decision tree based algorithm. Also in this graph there is a peak fo error eached on Thursday;
- **slot 3**: this is one of the slots for which the two algorithms behave very differently. It is the slot in which the accuracy of the *RepTree* algorithm is better than the one given by the *W-SMOreg* algorithm for almost all the days. The SVM based algorithm behaves better only on Sunday and Saturday, but during working days the decision based algorithm gives far more accurate predictions. The error seems also to be higher during working days and lower during weekends;
- **slot 4**: the *RepTree* curve in this case stays always above the *W-SMOreg* curve. The two curves assume close values only for Thursday, where the decision tree based algorithm has its lowest peak of error and the SVM based algorithm has its highest peak of error. The *RepTree* algorithm so is in general more inaccurate than the other algorithm, in particular on Saturday, where the gap between the two curves is approximatively equal to 2;
- **slot 5**: the graph is quite similar to one drawn for slot 4, except that the gap between the two curves is a little bit higher;
- **slot 6**: since this slot and slot 3 are the slots that are most critical in order to make predictions, the evolution of the error through the days of the week is very similar.
- **slot 7**: nothing particular can be said about the performance of the two algorithms for this slot of time, except that the curves have two opposite behaviours during Sunday and Monday. The *RepTree* curve decreases between Sunday and Monday, the *W-SMOreg* algorithm instead increases from Sunday to Thursday and starts decreasing until Saturday;
- **slot 8**: the curves are comparable to the ones drawn for slot 1, except that the highest peaks of error of the two algorithms are not located on Thursday but on Wednesday.

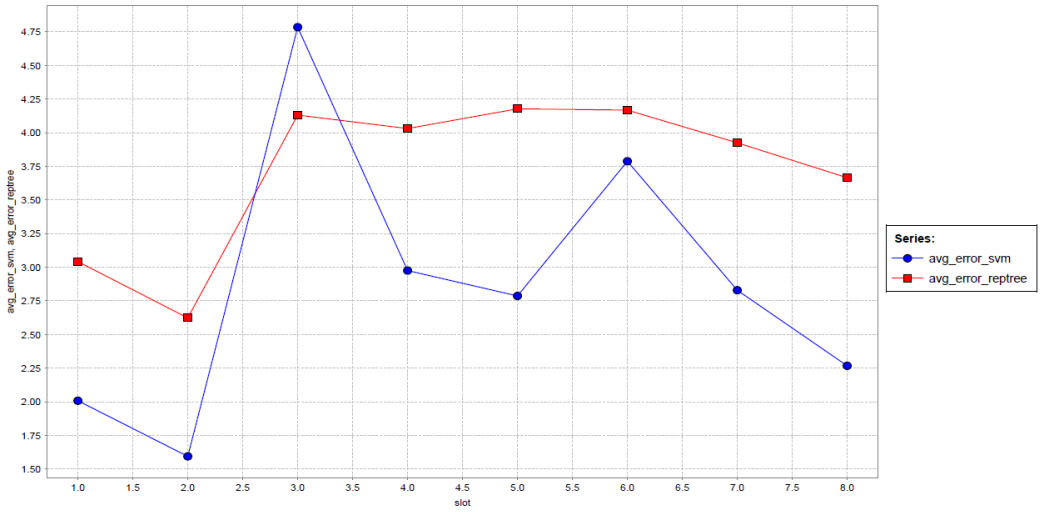


Figure 11.8: Prediction error distribution per slot

The graph above summarizes the results described in the section relative to the analysis by slot of time. As it was said in that section of this paper, the most critical slots for the two algorithms are slot 3 and slot 6, that correspond respectively to the periods of time between 9am-12pm and 6pm-9pm. Predictions are instead more accurate at the beginning of the day and at its end. The two applied algorithms have a gap of error approximatively equal to 1, making the *W-SMOreg* the most realiable algorithm considering its average performance. However the *RepTree* algorithm is more accurate or has comparable performance during critical slots.

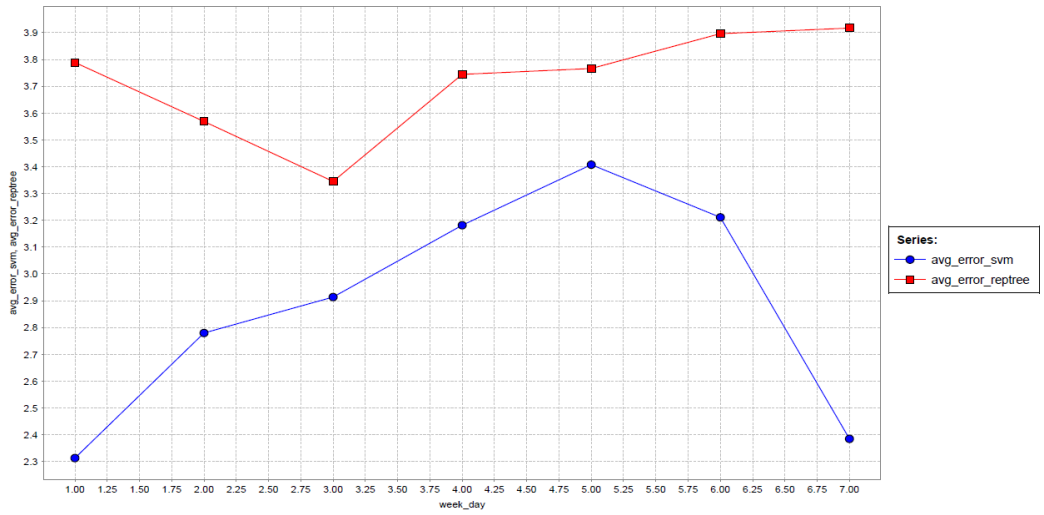


Figure 11.9: Prediction error distribution per day of the week

The graph above summarizes instead the results obtained by the analysis by day of the week. Considering this granularity the behaviours of the two algorithms are quite different:

- **W-SMOreg**: the most critical days are Thursday and Friday and during working days the error increases progressively between Sunday and Thursday. The best performance is given when predicting the number of vehicles per cell during weekends;
- **RepTree**: the shape of the curve in this case is quite different. Between Sunday and Thursday the error curve decreases progressively and then starts increasing progressively up to Saturday. In this case the best accuracy is reached on Monday and Tuesday and the worst performance is reached during weekends.

11.8 SVMs with different kernels comparison

Since the *W-SMOreg* algorithm was the algorithm that gave the best performance so far and it is SVM based, the next step is to compare other SVM algorithms with different kernels in order to see which one performs best. The comparison described in this chapter is between the *W-SMOreg* algorithm (polynomial kernel) and the *LibSVM* algorithm with *rbf* and *sigmoid* kernel. Both the *LibSVM* algorithms parameters were first optimized through the operator described before, choosing the same parameters and the same ranges. The input data of the optimization process is the same used before. The graph below shows the comparison between these algorithms, considering all the cells and making an average on the *timestamp* attribute.

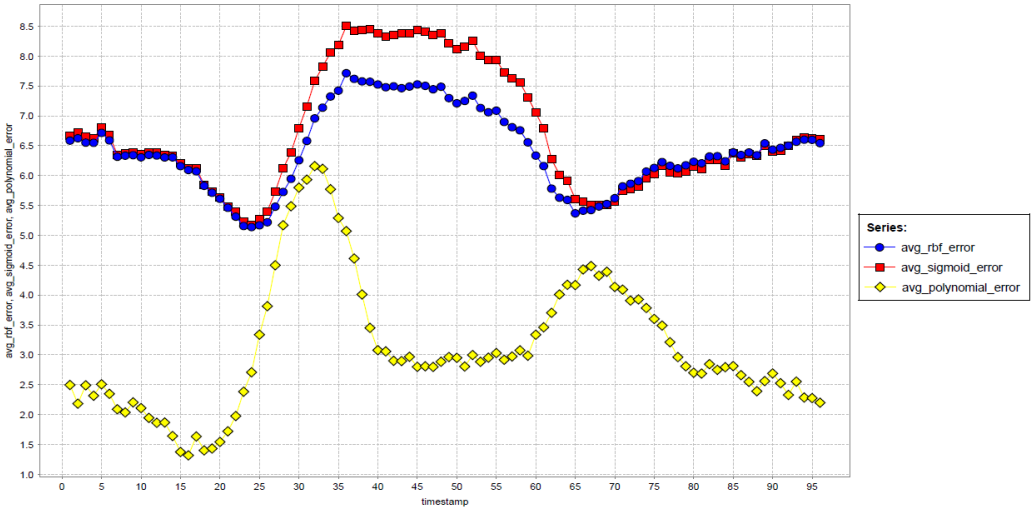


Figure 11.10: Comparison between SVMs with different kernels - all cells

The graph below instead shows the same comparison, considering only cell 40 (i.e, the cell that was considered during the optimization process).

11.9 Temporal Analysis with different training sets

This section describes the analysis by slot and by day of the week done using different input dataset, each one related to a specific day of the week or to a specific slot of time. In the first case this means that only data related to a day of the week is used to train and test the *W-SMOreg* algorithm. In other words to predict data for a specific day of the week I analyze only data relative to that day of the week in the past. The analysis was done training the *W-SMOreg* algorithm because from the analysis described in the previous section it seems to be the algorithm that gives the best performance when predicting the number of cars available in each cell. The same kind of input dataset was built for each slot of time, as it was done for each day of the week.

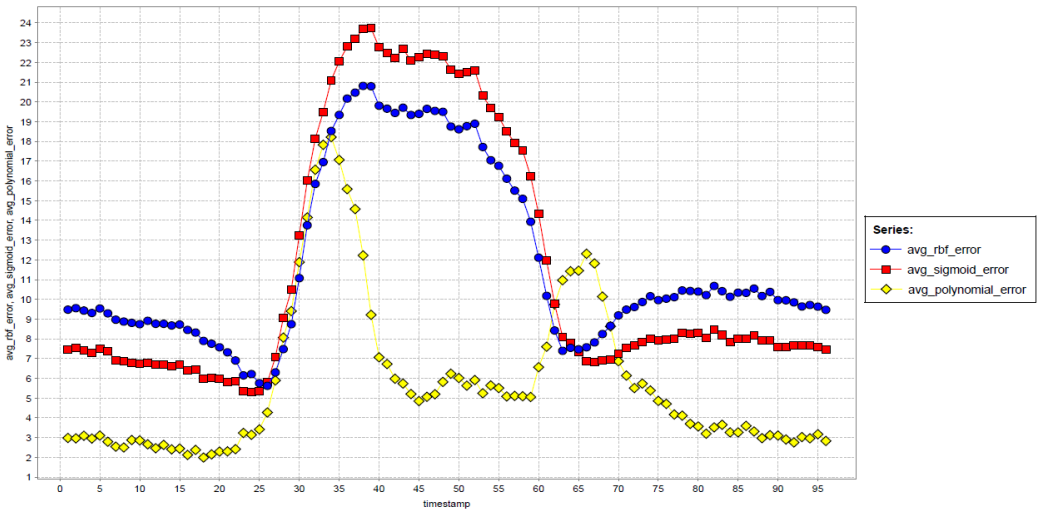
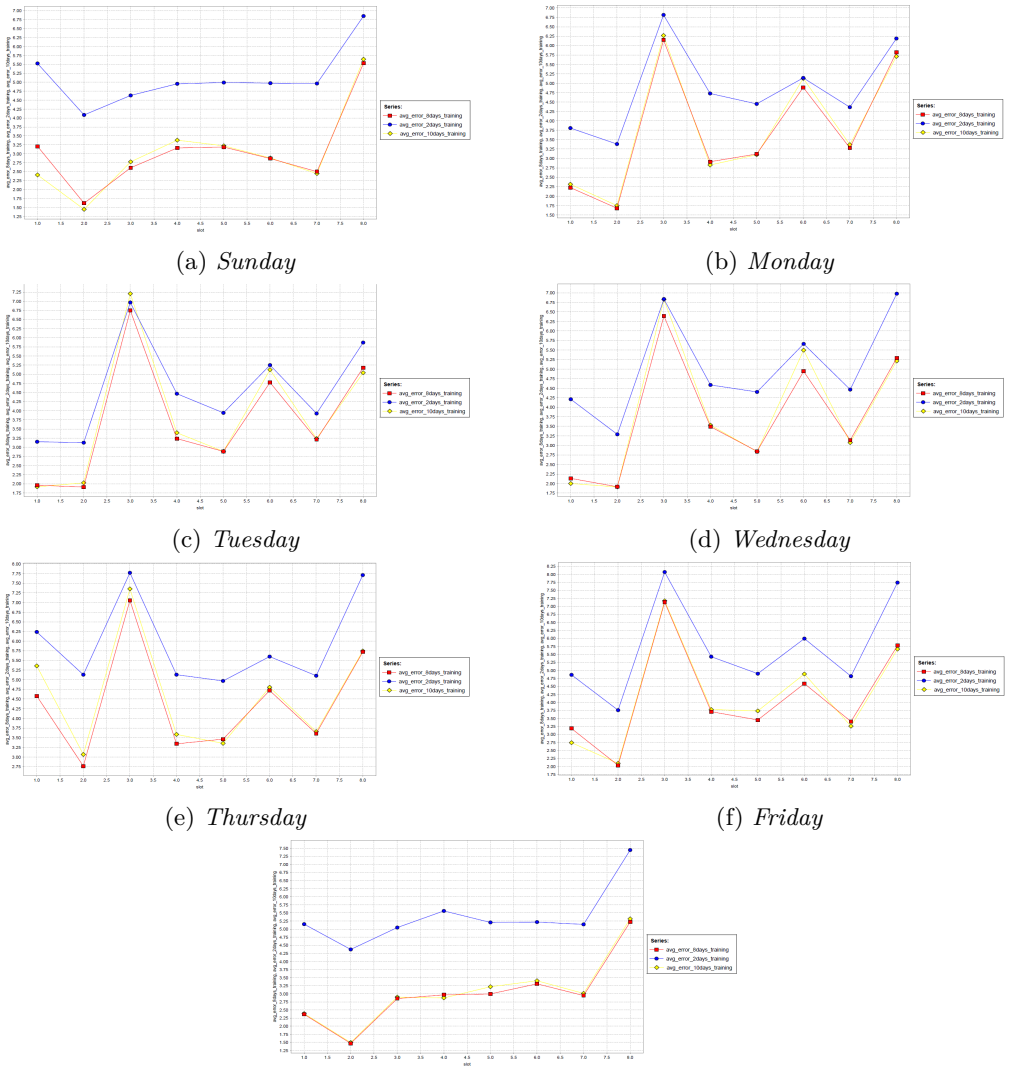


Figure 11.11: Comparison between SVMs with different kernels - cell 40

11.9.1 Analysis by day of the week

The graphs below show the error distribution through the various slots of time, giving as an input to the regression algorithm at each run the data relative to a specific day of the week. As in the previous cases the valildation was done using the *Sliding Window* operator, setting a training window with a width equal to 192(blue curve), 768 (red curve) and 960 (yellowe curve), and a test window equal to 96. This means that to predict the data relative to a certain day of the week I used data relative to that day in the previous two weeks, 8 weeks and 10 weeks.



(a) *Sunday*

(b) *Monday*

(c) *Tuesday*

(d) *Wednesday*

(e) *Thursday*

(f) *Friday*

(g) *Saturday*

Figure 11.12: Prediction error distribution for each day of the week, through each slot

As it was expected for each slot value each curve assumes a lower value if the training set dimension is higher, except some cases in which the red curve stays above the blue curve

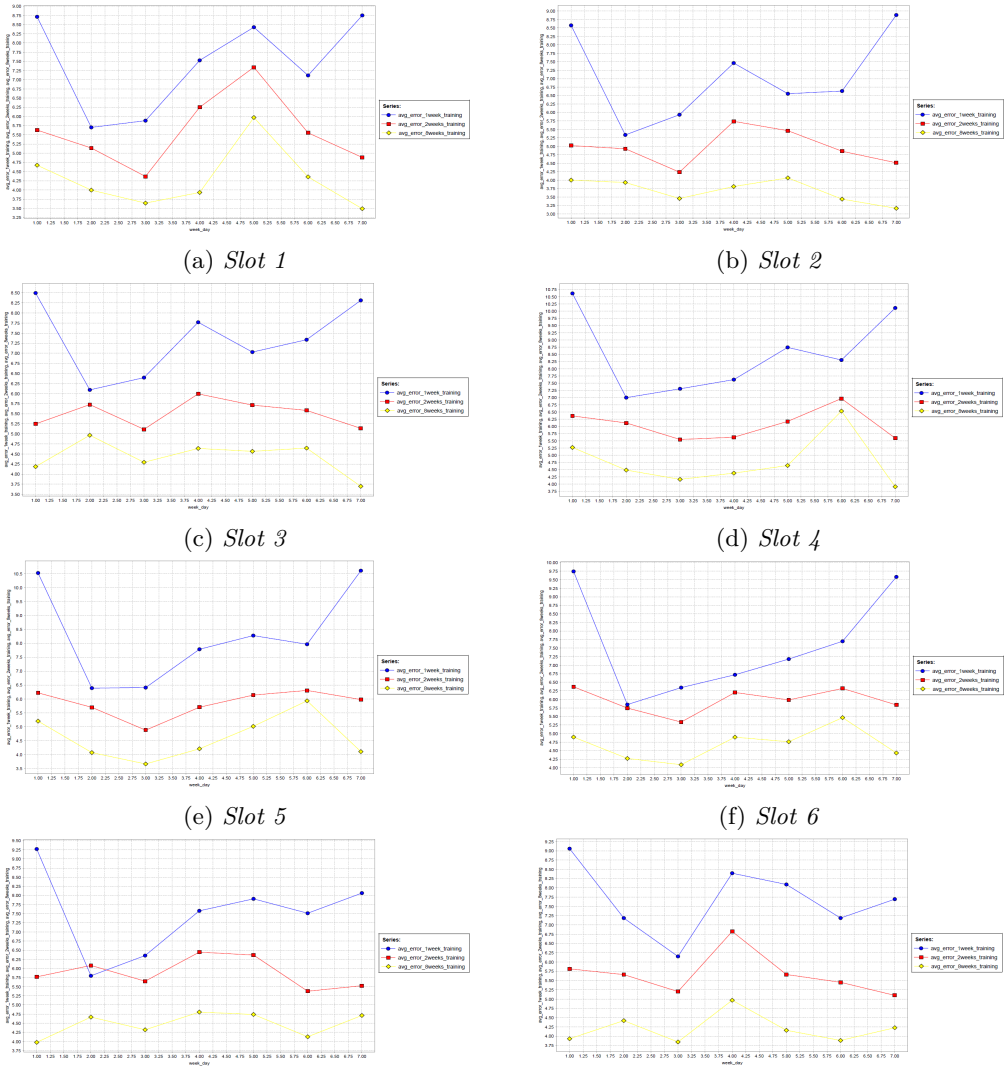
or very close to it. The highest error values are assumed for data relative to Thursday and Friday, and the less accurate prediction are made for slot 3, except during weekends, where the most critical slot is slot 8.

11.9.2 Analysis by slot of time

The graphs below show the error distribution through the various days of the week, giving as an input to the regression algorithm at each run the data relative to a specific slot of time. As in the previous analysis the validation was done using the *Sliding Window* operator, setting a training window width equal to 84 (blue curve), 168(red curve) and 672 (yellow curve) and a test window equal to 12. This means that to predict data for a certain slot I used data relative to that slot in the previous week, 2 weeks and 8 weeks.

Also in this case as the training set increases the error assumes lower values for each day of the week. The highest error values are reached for slot 4 and 5, and the most critical days in which making a prediction of the available vehicles are Wednesday and Thursday in most of the considered slots of time.

An important consideration could be taken by analyzing all these graphs. In all the test cases described in this section the algorithm has in general a worse performance when training is done considering each day or each slot of time data than when training cosndiering all days of the week and slots together. This is probably related to the dimension of the training set and to the fact that increasing the dimension of the training set in an analysis such this means probably going too far in the past and refer to data that is too much different to the data to be predicted.



(a) *Slot 1*

(b) *Slot 2*

(c) *Slot 3*

(d) *Slot 4*

(e) *Slot 5*

(f) *Slot 6*

(g) *Slot 7*

(h) *Slot 8*

Figure 11.13: Prediction error distribution for each slot, through each day of the week

Chapter 12

Conclusions and possibile improvements

Among all the considered algorithms, the *W-SMOreg* seems to behave best considering all the datasets which I worked on. However also the *RepTree* algorithm has acceptable performances, considering that its computational time is very low and its prediction results are not so inaccurate. From a computational point of view the *W-SMOreg* algorithm takes about 10 minutes to train and test, the *RepTree* algorithm instead takes no more than 5 seconds to generate a tree that gives a good performance, even if it is an algorithm based on simpler concepts. So in general the SVM based algorithms gave a better performance in terms of accuracy but a worse performance in terms of computational time. The analysis described in this paper also puts in evidence that to have a good performance in predicting a period of data that varies from 3 months to 4 months, using data covering a period of 2 weeks is enough to have a good accuracy in predicting the number of available cars in each cell in which the 3 cities subject of this analysis were divided. Even if in terms of numbers the average error is higher for the two USA cities, it has to be considered that in this cities the car sharing service is a highly consolidated reality and the number of cars made at the customer disposal is higher than in *use case 1* city. This means that the usage of the service is probably more frequent than here in Italy, so the prediction of the availability of cars could be more difficult than here. Another difference between the 3 considered cities is the usage pattern, in particular in terms of the periods of time during the day in which the service is exploited but also the days of the week in which the service is most used. Some possibile improvements and extension of this work could be consider other regression algorithms and study their performances, for example other kinds and implementations fo neural networks. Another possible extension and improvement could be dividing the cities in a grid with smaller granularity, considering cells smaller or bigger in order to see how their dimension could affect the regression algorithms performance. From the data point of view the input datasets could be enriched with more informations than the timing informations I added to the inital datasets. For example it could be extended adding informations about the services or commercial activities present in each area, or informations about weather or other external factors that could push a service customer to exploit the service to travel among the different city areas.

Bibliography

[1] H. Becker,F. Ciari,K.W. Axhausen,*Modelling free-floating car-sharing use in Switzerland: A spatial regression and conditional logit approach*, 17th Swiss Transport Research Conference, Monte Verit / Ascona, May 17, 2017

[2] S. Schmller, S. Weikl, J. Mller, K. Bogenberger, *Empirical analysis of free-floating car- sharing usage: The Munich and Berlin case*,Transportation Research Part C: Emerging Technologies, V. 56, July 2015, pp. 34-51

[3] C.Qian, W. Li, M. Ding, Y. Hui, Q. Xu , D. Yang,*Mining Carsharing Use Patterns from Rental Data: A Case Study of Chefenxiang in Hangzhou, China*, World Conference on Transport Research 2016 , Shanghai, 10-15 July, 2016, pp 1-8

[4] C. Boldrini, R. Bruno, M.H. Laarabi,*Can data mining help car sharing?*, European Transport Conference (ETC 2016), Barcelona, Spain, 2016

[5] X. Zhu, J. Li, Z. Liu, F. Yang,*Location deployment of depots and resource relocation for connected car-sharing systems through mobile edge computing*, Internation Journal of Distributed Sensor Network, V. 13, n.6, 2017

[6] https://machinelearningmastery.com/support-vector-machines-for-machine-learning/ (Last Accessed in August 2018)

[7] https://nygeog.carto.com/tables/car2go/public (Last accessed in August 2018)

[8] https://aaronparecki.com/car2go/ (Last Accessed in August 2018)

[9] https://www.meetup.com/it-IT/Innovation-in-Motion/events/223841329/?eventId=223841329 (Last accessed in June 2018)

[10] http://caseorganic.com/2014/01/mapcamp-portland-hackathon-6pm-friday-jan-10-2014-through-6-30pm-sun-jan-12-2013-at-isite-design/ (Last accessed in June 2018)

[11] http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/REPTree.html (Last accessed in August 2018)