

DAT410 - Assignment 8

Eric Johansson & Max Sonnelid

March 15, 2021

Group number: 70	
Module number: 8	
Eric Johansson	Max Sonnelid
970429-2854	960528-5379
MPDSC	MPDSC
johaeric@student.chalmers.se	sonnelid@student.chalmers.se
25 hours spent	25 hours spent

We hereby declare that we have both actively participated in solving every exercise. All solutions are entirely our own work, without having taken part of other solutions.

Introduction

The aim of this module is use the gained knowledge from this course and extend it by creating and completing an own, interesting mini-project. This group's choice was to complete the "Walmart Recruiting - Store Sales Forecasting" competition on Kaggle and our solution will thoroughly be described below. Thereafter, the lectures during this week will be summarized. Finally, reflections on both the previous module and the course as whole will be given.

1 Description of mini-project - "Walmart Recruiting - Store Sales Forecasting"

The group had for a long time talked about doing a Kaggle competition, but because of busy schedules during this reading period, it was hard to find any time to do that. However, when it for the last module was possible to freely choose a mini-project related to the course, this seemed like a perfect opportunity for a Kaggle competition.

Kaggle is one of the biggest online communities for data scientists and machine learning practitioners, where the users are able to publish and access both data sets and models. An important feature of Kaggle is its competitions, where companies and organizations are able to post a data-related problem. Any user can access the data and problem statement and then publish their solutions visible for anyone. Some challenges hosted on Kaggle have lead to big impact by e.g. enabling better HIV research and improving traffic forecasting. As Kaggle is a great place for finding clever data scientists, some companies also post competition with the aim of recruiting successful participants.

This group's choice of competition fell on such a recruiting competition, namely the "Walmart Recruiting - Store Sales Forecasting". It was chosen because of three main reasons. Firstly, all members of the group has a big interest for business from our background in Industrial Engineering. It is therefore an interesting task to see what data and challenges that one of the world's biggest retail businesses has, especially what they focus on when aiming to recruit data scientists. Secondly, we had not worked anything with regressors (which is the general method for sales forecasting) in the course so far and thought that working with regressors would be a good extension to the previous work with classifiers. Thirdly, the level of this competition seemed appropriate for one week of work as the properties of the data and the problem provided possibilities to easily adapt the complexity of the solution to the time available.

1.1 Problem statement

The starting point for the "Walmart Recruiting - Store Sales Forecasting" competition is historical data for 45 Walmart stores located in different regions and where each store has several departments. Especially important for Walmart are sales around selected holiday markdown events, which are known to affect sales. However, Walmart finds it challenging to accurately forecast sales around these events. The objective of this competition is to forecast the sales for each department in each store, where special emphasis is given to accurately forecast sales around the holiday markdown events.

Evaluation is made on the metrics "weighted mean absolute error (WMAE)", which is created to reflect the stated objective:

$$WMAE = \frac{1}{\sum w_i} \sum_{i=1}^n w_i |y_i - \hat{y}_i|$$

where

- n is the number of rows.
- \hat{y}_i is the predicted sales.
- y_i is the actual sales.
- w_i is are weights, where $w = 5$ if the week is a holiday and otherwise $w = 1$.

1.2 Available data

Fours different csv files were given for solving the above stated problem:

stores.csv provides general facts about all stores and contains 45 rows (one for each store) and 3 columns: Store, Type and Size.

features.csv provides facts about certain conditions connected to a store and date. It contains 8,190 rows (one for each combination of store and date) and 12 columns: Store, Date, Temperature (average temperature in region), Fuel Price (fuel price in region), MarkDown1, MarkDown2, MarkDown3, MarkDown4, MarkDown5, CPI (customer price index), Unemployment (unemployment rate) and IsHoliday (whether week is holiday or not). The MarkDown1-5 columns contain anonymized data related to the promotional markdowns and is only available after November 2011.

train.csv contains the training data for each store and department. It contains 421,570 columns (one for each combination of store, department and date) and 5 columns: Store, Department, Weekly Sales and IsHoliday.

test.csv contains the test data for each store and department. It contains 115,064 columns (one for each combination of store, department and date) and 5 columns: Store, Department, Date and IsHoliday.

1.3 Observations about the data

In order to create an optimal model for this task, it is necessary to understand the properties of the available data and therefore the first step was to create histograms for all features. Because of limited space, the histograms are not included in this document, but are visible in the attached code. No changes on the data were made directly based on the histograms, but there were three features we wanted to take a closer look at, namely Weekly Sales, CPI and Unemployment.

As Weekly Sales is the target feature that is being predicted in this task, a graph showing the weekly sales per week was created in Figure 1. As this task puts special emphasis on the holiday weeks, it made sense to highlight these weeks in the graph, which was done by drawing red lines for them. Two major conclusion can be drawn from the graph. Firstly, the weekly sales does not differ any particularly between the observed years and therefore no account must be taken of sales growth. Secondly, during the year the sales are considerably higher at the end of the year and around the two last holidays. The outcome of this observation was that we realized the the model can benefit from differing between the two first holiday weeks and the two last holiday weeks. Therefore, a new column BigHoliday was created for this purpose and decreased the WMAE slighrly.

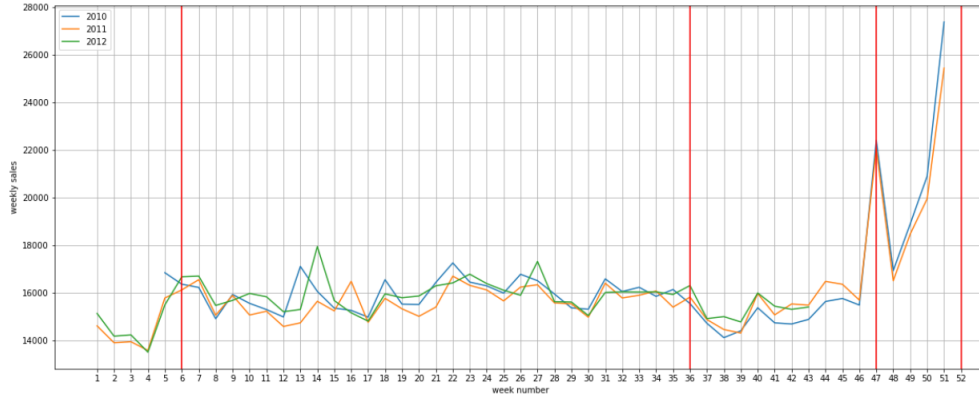


Figure 1: The weekly sales for each year, with the four holiday weeks marked by the red lines and with the week number on the x-axis and the weekly sales on the y-axis.

From May 2013 and on, data about unemployment and CPI are missing (probably because these dates are in the future relative to the release date of the competition) and these NaN values need to be handled in some way. Therefore, the existing values for these features were plotted in Figure 2 and Figure 3 in order to be able to study the behavior of the features. Two suggested ways for handling the NaN values were then thought of. Firstly, by taking the last values for both unemployment and CPI and assigning them to all NaN values. This is probably a reasonable idea as the values are only missing for three months (last date in the data is end of July 2013). Secondly, as it seems like the unemployment has a decreasing tendency and the CPI has an increasing tendency, it could be interesting to take this change into account when assigning values to the NaN values. However, because of the short time period, the benefit from taking the change into account would probably not be rather big. In the end, it was noticed that the selected regression model, XGBoost, had a built-in function for handling missing values and because of time deficiency the suggested ways of handling these NaN values was not implemented.

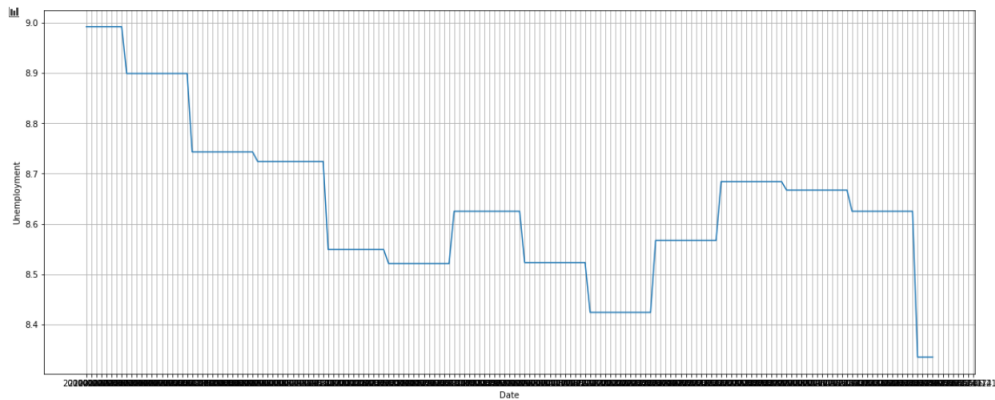


Figure 2: The development of unemployment levels per date, with the date on the x-axis and the unemployment level on the y-axis.

1.4 Data pre-processing

As the data used for training and testing was spread out on four different files, the first step was to merge all the data into one single DataFrame. For the training data, this was done by firstly merging `train.csv` with `features.csv` on the features Date, Store and IsHoliday (in order to avoid getting two holiday columns) and secondly merging this resulting DataFrame with `stores.csv` on the feature Store.

Even though the column Date is already present, we thought that the ML algorithm would benefit from having more information extracted from this column and therefore new column with week number, year, day and weekday were created.

A column called BigHoliday was also created, where week 47 and 52 were assigned the value 1 and

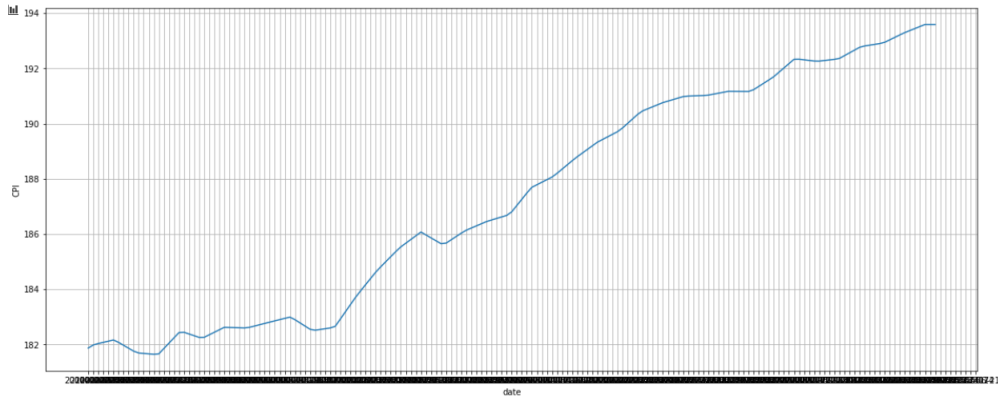


Figure 3: The development of CPI levels per date, with the date on the x-axis and the CPI on the y-axis.

all other weeks the value 0.

In order to enable an ML algorithm to process the data, the features need to be converted to numerical format. For `IsHoliday`, this was simply done by converting the Boolean values to 1s and 0s. The `Date` column contained hyphens to separate the date, year and month, which were removed and then the data was converted from string format to numerical format.

All the five `Markdown` contained many `NaN`-values, which is a rather challenging task to handle. Until no better solution for these columns have been found, they are dropped from the dataset.

The resulting `DataFrame` was split into target values (`Y`) consisting of the `Weekly Sales` column and data values (`X`) consisting of the other columns. Finally, this data was split into training and test data by Sklearn's `train_test_split` method with a test size of 20 %.

1.5 Machine Learning algorithm and results

An interesting ML algorithm that we had heard about and wanted to try was XGBoost, which is an abbreviation for eXtreme Gradient Boosting. It is the currently dominating algorithm in Kaggle competitions with tabular data. As this task is based on tabular data, this algorithm therefore seems like an appropriate choice for this task. (Brownlee, 2016)

XGBoost is based on a gradient boosting decision tree algorithm. Such a model produces a prediction model as an ensemble of many decision tree models. New models are created with the purpose of correcting the errors made by the existing models and each new model follows the gradient of the error in order to minimize the final loss function, hence the name gradient boosting. Models are added until the loss does not decrease anymore. Gradient boosting is suitable both for classification and regression tasks. (Brownlee, 2016)

Compared to other gradient boosting algorithms, XGBoost is advantageous for two main reasons. Firstly, because of its fast execution speed which is enabled by e.g. parallel processing and cache optimization. Secondly, because of its high performance which is e.g. enabled by automatic handling of missing data values and efficient regularization. (Brownlee, 2016)

In order to find suitable hyperparameters for the XGBoost model, the values were firstly chosen by selecting reasonable values based on our previous experience. In order to optimize the hyperparameter values, grid search was used, where the hyperparameters `max_depth`, `n_estimators`, `reg_lambda`, `gamma`, `min_child_weight` were tuned in the grid search model one-by-one by first looking at a wide range of values with big steps between the values and then sequentially narrow down the search range around the optimal values. These were the optimal hyperparameter values found `n_estimators=60`, `reg_lambda=1.0`, `gamma=0`, `max_depth=15`, `min_child_weight=5`. Interestingly, this hyperparameter tuning only improved the score to a very small degree, which either shows that the starting values were reasonable (less probable) or that the grid search can be performed in an even more structured way (more probable)

With the above stated hyperparameters, it was eventually possible to achieve a weighted mean absolute error of 1,343 on the test data created from the `train_test_split` method.

Furthermore, we also plotted the feature importance scores, which can be seen in Figure X. The feature importance score for XGBoost is simply calculated as the number of times each feature is split on in all the created tree models. As can be seen, the features `IsHoliday`, `Type_A`, `Type_B`, `BigHoliday` and `Type_C` have a clearly lower score than all the other features. It was therefore attempted to remove all these features from the data set. However, this increased the WMAE considerably. The low feature importance scores could therefore probably be explained by the fact that these features are not split on particularly often, but the few times they are split on, it considerably contributes to lowering the WMAE.

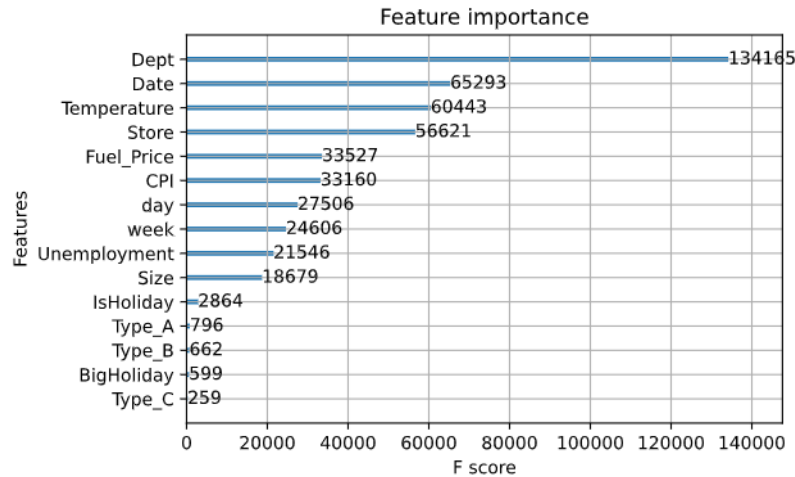


Figure 4: Feature importance scores plotted for all features for the XGBoost model.

For the sake of comparison, we wanted to see if another regressor could match or beat this result. As recommended on the lecture, the Random Forest algorithm is a good first choice for any classification/regression task and therefore we choose to try it. However, after optimizing the hyperparameters, the minimal weighted mean absolute error achieved was 1,527, which does not beat XGBoost. It could of course have been interesting to also compare XGBoost with other regressors, but because of time deficiency and that many sources state that XGBoost is in general superior for tabular data, we did not try any other algorithms.

1.6 Kaggle submission

After having improved the WMAE on the training data to a desirable level, it was then time to train it on the entire training data and let the model make predictions on the real test data. The submission file was then created by the method `produce_output`, which for each weekly sales prediction added a column with the store number, department number and the date (as stated in the competition rules) and eventually created a CSV file with 115,064 rows.

After submitting this file on Kaggle, we eventually succeeded at achieving a WMAE of 4,390, which would give us a ranking of 379 among all 688 submissions (see Figure 5). This competition was however officially closed and we were therefore unfortunately not able to get an official ranking for our submission, even though late and unofficial submissions were still accepted. Considering that the aim with this competition was to recruit data scientists to Walmart, one of the biggest retailers and therefore one of the biggest data owners in the world, it is very probable that many people with more advanced knowledge than our and people who have spent more than our 20 hours on the submission have participated in this competition. Therefore, we regard this middle position in the ranking as a success with respect to our skills and time available.

1.7 Potential for improvement

As described in the section regarding the Machine Learning algorithm, many sources claim that the XGBoost model is superior for tasks based on tabular data. As the Random Forest model had a considerably worse performance than XGBoost, any major performance improvements can probably not be expected from trying different kinds of regressor. Something that could potentially contribute to small improvements is performing grid search in an even more structured way, with more research






Name		Submitted	Wait time	Execution time	Score
submission_file.csv		a few seconds ago	1 seconds	10 seconds	4390.64398
Complete					
Jump to your position on the leaderboard ▾					
376	▲ 5	David		4368.54853	1
377	▼ 2	18554		4371.15642	4
378	▲ 5	Alex Prodan		4384.85116	61
379	▼ 2	Jose		4401.82480	48
380	▼ 2	Chase		4409.82451	10

Figure 5: WMAE calculated on our submission together with the leaderboard rankings around the same WMAE.

about the hyperparameters and recommended starting values for them. However, as a basic grid search has already been done, the improvements will probably only be small by extending the grid search.

A major deficiency in the final model is that it does not include the five different Markdown columns in any way, as they include many NaN values that need to be handled, and thus much data that could contribute to a lower WMAE is completely ignored. One attempt was made to replace all the NaN values with 0s, but this led to a slight increase in WMAE and it was decided to drop all the Markdown columns until a better solution was found. Two potential solutions could be suggested. Firstly, the NaN values in each column could be replaced by the average value for each column. Secondly, the NaN values in each column could be replaced by the average value for each store in each column. However, reflecting about the pros and cons of these two alternatives ends up in the conclusion that more knowledge about the data in the Markdown column is needed. If more time was available, it would probably have been a good idea to visualize the Markdown data and draw conclusions from how it is distributed, e.g. in relation to the store number, department number et cetera.

2 Summary of lectures

2.1 Max Sonnelid

2.1.1 Follow-up lecture - Dialogue systems

- The lecture commenced by discussing about the *challenges of implementing a dialogue system*, where topics as the difference between intent detection and domain detection, that keyword matching is too simple in many cases and the challenge of correctly storing history were discussed.
- There are two main ways for asking for information: in a *query-driven approach* keywords are used for finding relevant documents, while in a *question-driven approach* questions are used for getting specific answers.
- A question-driven approach can either be *information retrieval-based* (sources information from documents), *knowledge-based* (sources information from encoded database of information) or a *hybrid* of the two latter types (which IBM Watson is an example of).
- In order to select both the *relevant document* that could answer the question and select the *relevant passages* from that document, ML algorithms could help ranking documents and passages according to relevancy.
- When the relevant passage in a document has been identified, the next task is to identify the *relevant answer span* according to the answer type. This answer extraction can be done by looking for *answer-extraction patterns* in documents by taking advantages of answer phrases and question phrases.

- *Knowledge-based question answering* is based on logical representation of knowledge, which is could be done by a semantic parser which turns a text-based question into a mathematical form, which in turn then can be translated to a database query.
- As knowledge can have very many different shapes, e.g. structural knowledge, meta-knowledge and declarative knowledge, it is not possible to represent all kind of knowledge in one type of structure. However, one structure used for storing declarative knowledge are *RDF-triple databases* (abbreviation for resource description framework), which store knowledge in triplets consisting of a subject, predicate and object, which simplifies retrieving answers from queries.

2.2 Eric Johansson

2.2.1 Follow-up lecture - Dialogue systems

One of the main ideas of this module was that by building a model entirely from scratch, a lot of issues arises, which is nice to be aware of when implementing already existing models.

There are two ways of asking for information. The first way is query driven, where input is scaled down to queries and then those queries are feeded in a database - this is known as information retrieval. The other way is more specific. A certain question is asked, and we want the computer to understand the question and then directly give an appropriate answer. Question answering can be based on multiple models. The first one is called information retrieval, the second is called knowledge-based and the last is hybrids such as IBM Watson.

The first part of an information retrieval model is to tweak the input string into useful queries. It can also be beneficial to synonyms to important words in order to increase the chance of success. Moreover, queries can be reformulated in a way that they are most likely to appear in the data base.

Another way of retrieveing answers is to use a semantic parser, which translates sentences into some mathematical form.

A practical example of a system that answers question is IBM Watson, which is a mixture of a knowledge-based and information retrieval model. First it processes the data and then looks for answers in text resources as well as in structured data. Multiple answers are then collected and compared against each other. Finally one answer is chosen together with its probability.

3 Reflection on previous module

3.1 Max Sonnelid

The interesting thing about last week's module, which revolved around dialogue systems, was the clear separation between the abstracted general functionality and the situation-specific implementation, where the general functionality mainly consisted of the frame structure and the structure for asking questions until the whole frame is filled and the situation-specific implementation mainly consisted of the situation-specific frames and related data. Creating a dialogue system for a very specific purpose could probably have been done without paying any attention to general-purpose functionality and our first early attempts at creating the dialogue systems consisted of text being processed by an immense number of if statements. However, we soon realized that if we were going to create a dialogue systems that could handle at least three topics, it was not feasible to continue with the only if statement based approach.

It was a rather tricky task though to define the general structure of an arbitrary dialogue as one of the properties of natural dialogue is that details are mentioned in an unstructured way and that the original purpose of a dialogue, e.g. book a flight ticket, might be extended by e.g. discussing hotel options also. We soon realized that it would not be possible to handle every single property of a natural dialogue. Instead we had to find a good balance between, on the one side, restricting unstructured aspects of a dialogue in order be able to provide insightful answers and, on the other side, allow unstructured aspects of a dialogue in order to allow a natural type of dialogue. This challenge is somewhat analogue to the challenges of making neural networks interpretable. In order to get as high accuracy as possible from neural networks, one often has to forsake the interpretability of the model or the other way around. The message from both these cases is that there exists an inherent conflict between making an efficient AI system and making an AI system adapted to human

behavior. This is probably a conflict I will face and handle many times in my future Data Science career.

3.2 Eric Johansson

This weeks assignment consisted of building a simple dialogue system which were able to help out with a few predetermined tasks. We chose to go with the examples that were given in the assignment PM, namely, weather predictions, restaurant recommendation and transport support.

I really enjoyed working with this assignment, even though it (according to me) did not involved a lot of AI. It was fun to think a lot about how build a model that could solve some tasks and at the same time would be easy to further develop. Before we started building our model, we read the article that was given in the PM, which introduced us to the concept of frames. We interpreted that as building a class called frame which worked as a framework of how to implement new tasks. By trying to stick to that framework, some problem arise. For example, if two task differed a lot from each other, how do you follow a predetermined framework to solve these tasks? After some thought, we managed to solve this problem, and by doing that, learnt a lot. Another important lesson was about how to handle strings from user input to classify a problem and to retrieve data that is needed to yield an answer. I really liked the concept of using regular expression and would really like to learn more about it. To further improve our model, we asked some friends to use it. In this way, we could get new perspectives on how question regularly are formed which enabled us to make our model able to handle these inputs. Overall, it was a fun project. Once again, it was not very clear how to solve the problem, something I think is good to get use to.

4 Reflection on the course as whole

The main thing we both really appreciated in the course was the great freedom of choosing your own focus areas in the implementation parts of the assignments. With such a great freedom and few instructions, you have to use a lot of creativity and all of your so far accumulated knowledge for solving the assignment. Furthermore, without specifying each individual step of the implementation, you are also given project management responsibilities in the course, including setting up an appropriate time plan and identify which parts of a project that are "must-haves" and which parts that are only "good-to-have". This probably resembles a real AI project in the industry a lot, and developing both creative and project management skills is useful for our future careers. When it comes to developing the creative skills, the freely chosen mini-project was also really appreciated as it was very motivating to be able to expand knowledge in an area of our own choice.

Furthermore, the course benefited from having a good mix between both reading articles about a certain topic, implement an AI system for the topic and discuss around challenges with the topic. If you have some knowledge about the subject area before-hand, it is much easier to reflect about whether a certain evaluation metrics is appropriate, what features to select and/or transform et cetera. We also enjoyed the concept of reflection. By talking about a previous project each Friday, more lessons were learned about each subject. Thoughts that arose during the week were often answered during these lectures. It was also to talk theoretically about how to further improve each system.

A small detail that we think the course would benefit greatly from is to have a slight emphasis on developing an attractive graphical interface. In order for an AI systems to be attractive and understandable for the user, it is crucial that the system has an (somewhat) attractive graphical interface. We as developers would also probably feel that we have created something more useful if it is more nice-looking. Implementing a graphical interface should of course only be a small detail of the implementation and an example of how the course team could help in this would be to give examples of easily implementable libraries for this purpose.

We don't really know how much time other groups have spend on this course, but by speaking some other participants, it seems that everyone put a lot of time into their project. By having a large set of mandatory tasks that should be solved each week, we had to work a lot. With that said, I think the biggest reason of why we put so much time into this course was because it was very interesting and this is the type of course that becomes better the more effort you put into it.

References

Brownlee, J. (2016). A gentle introduction to xgboost for applied machine learning. <https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/>.