Cover Page

Student Information

Name:	Oscar Armando Leal Marcos
Email:	Leal-Marcos_Oscar@student.ceu.edu
Student ID:	1903161
Supervisor:	Prof. Gyorgy Bogel

Abstract

This document summarizes the author's Capstone Project Work of the MSc Business Analytics at Central European University, Budapest, 2020.

The client that supported the project is an operator in the car rental industry, specifically their customer services department. They happen to have several dashboards developed in proprietary software tools to monitor the operations of the department and performance of their agents. They find these tools not flexible enough, neither interactive. As a hypothesis, the client opted to take Shiny as an opportunity to migrate their current dashboards to this other tool.

For this, the author's task was to get involved in the data sets that are used to map their current dashboards, by using an anonymization script, doing a data exploratory analysis, and posteriorly developing a potential substitute for their dashboards, in *flexdashboard* with Shiny, written in the programming language R. After two major iterations, the final output came to be a potential demo that is worth to be considered in the future by the company.

Contents

Capstone Project Discussion

In order to satisfy the goal of keeping the ability of measuring agents at a given season or time frame, and to monitor the causes of the reports that are handled by the customer services of the company, in a new technology, the previous dashboards were shown to the author. These dashboards were noted to be not too intuitive; not flexible enough and they also happen to be lacking interactivity. It was mentioned that there are performance issues as well.

For data handling, the preferred language was R for its ability to create visualizations in a straightforward way, and with these plots, generate the desired dashboard. Throughout the program, all kinds of R skills were learnt, and as a final project, a dashboard in Shiny wraps up most of the concepts learned.

When the data sets were introduced in wide table formats in *.xlsx*, they needed a moderate clean and feature engineering based on merges. One data set contained report of complaints with their causes and details, and the other one was about the daily schedule and records of activities from the agents (these records are coming from either e-mails or phone calls), both data sets had a high number of variables. Monitoring the activity of agents is a common strategy that is known to be effective in order to quantify the performance of workers, in order to make important decisions of how much work force is the best to proceed in different times of a day, and on the days of a week.

As of the data pipeline, the source comes to be these two data sets extracted from their internal tools, posteriorly, an anonymization script that hides sensitive data from the sets is ran, then we use the sets to create a cleaning and analysis script that provides us an EDA with minimal effort. With this, the author managed to understand the structure of the data sets in a more thorough way.

Then, following the analysis, a dashboard was designed and developed in *flexdashboard*, where, with a Shiny engine is possible to code the dashboard in R, this app manages to fit two of the former dashboards provided by the client into one, by separating them in tabs. The reason to opt in for *flexdashboard* is because of its power of constructing a layout modularly in a much more straightforward way than it would be with Shiny alone. The only file needed is the *.rmd* which itself is containing the layout, and the operations necessary for the data to be displayed in desired dimensions. This file is easy to follow, allowing future revisions to modify or create existing/new dashboards.

The developed set of dashboards contain multiple filters and plots (from library *ggplot* and *plotly,* filters are built in a smart way where you can select all / deselect all and have a live search feature built within.

In the report of complaints dashboard, they vary from being able to filter multiple dates, countries, station types, categories of reasons, and the type of bill payer, the plots contain data organized month by month, showcasing the amount of reports separated by categories, sub categories, and countries.

For the daily activity of agents, the filters range from month, manager name which lets you find the subset of agent ID's belonging to him, the plot contains a bar chart where it is possible to count the amount of work given by the subset selected of one or more agents.

Data tables are available next to the plots described. These happen to have a functionality of being able to download that particular subset of data, or the raw data of the dashboard (in the case of the complaints report dashboard) with the filters selected.

Although the data set provided represents records from only the first months of the year 2020, it is easy to come to a scalable solution once it is incrementally adding more data perhaps with automatized ETL scripts that can gather the data and refresh the dashboards.

Deployment and Conclusions

Given the results, the dashboards are now considered to be in a demo phase. Future discussion regarding the adaptation of new technologies is expected, and the results are a fair enough base to support the change. The features that Shiny and *flexdashboard* provide are more than good enough to at least consider including these tools at a certain incremental degree within a company.

With the demo, more ideas come into mind about what to add or modify, these ideas could be going around the path of prediction of the timeframes provided (more data would be needed), more dashboards can be incorporated by just adding another tab into the existing demo, more filters, perhaps it would be safer to require a username and password to log in to the app, if the server will be hosted as web.

For the server the best choice most surely is to add a *ShinyProxy* in order to let Java to run the Shiny apps in a closed environment utilizing Docker. It is also needed to be considered, if this is heading towards the full automatization, how will an ETL process be running in the background to fetch the freshest data, so that the dashboards are always counting with the most updated set.

Once the possibility of using these technologies is proved, it most likely will make them to be adopted incrementally, especially since these tools are mostly open source, they will most likely, with a small amount of work added, provide better dashboards that can help the company drive decisions in a more effective way, where the visualizations will cause immediate insight of them.

The code provided by the author is a great base for adding other dashboards, data sets, filters, plots, etc. in a modular way. Depending on the decisions that are made by the company, this code can be modified to make a better approach once the use cases are redefined after an interest is noted on these tools.

Lessons Learnt

Interactivity in a dashboard can be challenging to achieve especially if the files sizes that are being handled are high in volume. *Flexdashboard* makes easier to establish layouts in order to focus in the pure development, further customization can be made by including *.css* as visual guidelines for the HTML rendered, for this, more experience in design might be useful if further changes are noted.

Nonetheless, the modularity that is offered with these tools can make attractive to switch from previous tools to these that come from an open source ground. Thanks to the support and feedback from the client, it has been possible to develop such dashboards in a new technology that could be adapted in the future by the company.

As stated by the client, this project has started internal discussions about the technology, and how to integrate it in the company.

As an idea for stepping the demo towards production would be to rent a Linux server, where the Shiny app can be served by certain tool/proxy of preference, available for the users which happen to be the customer service personnel.

With the current global situation, priorities might differ from an immediate adaptation, nonetheless, the demo showcases important aspects that indicate the power of customization that *flexdashboard* with a Shiny engine written in R can provide.