OTTO VON GUERICKE UNIVERSITÄT MAGDEBURG

# DECISION SUPPORT APPLICATION

# SUPERMARKET

**Under the guidance of: Stefan Willi Hart**

**Submitted by**

Jagriti Labh,

Pooja Mishra,

Tejaswini Mahendra,

Tina Ancelin Joseph Martin

## Table of Contents

## Table of Tables

## Table of Figures

## 0. Document Information

### 0.1 Versioning

This document is in

**Versioning:** v0.8 – ready for review; v0.9 – ready for sign-off; 1.0 sign-off

**Table 1: Version History**

| Version | Date | Who | Comment | Release |
|---|---|---|---|---|
| **0.1** | 27/03/2019 | | | |
| | | | | |
| | | | | |

## 1. Introduction

This report is a comprehensive technical design document which gives an overview of the project implementation.

It serves as a manual for the project and provides deep insight into various aspects of it which would help a person in understanding an overview neatly and precisely. The document also focuses on explaining about dataset and the result obtained from it in the form of data and it also explains clearly the analysis adapted for the successful working of the project.

As we dive deep into the document will discuss about various stages of project. Here we describe our dataset (Rossman dataset) which was selected keeping in mind the keen requirements of our project which is supply chain management. In accordance with the dataset data we created the data model along with functional and non-functional requirements and also physical and logical data models with their uses. In short, we sorted the data which was analysed using machine learning algorithms and stored the data in table.

### 1.1 Data Description

### 1.2 Scope and Objectives

**Table 2: Services in Scope**

| Process | Explanation |
|---|---|
| **Load and Transform data** | We have three train, test and store CSV files. Which is transformed into one CSV file. Excel and Python, Jupyter IDE is used for correction and removing duplicates from the table. Later, the CSV file is imported to SAP Hana in GBI-011 schema. The table is added to the catalog using .hdbtable and .hdbti files. |

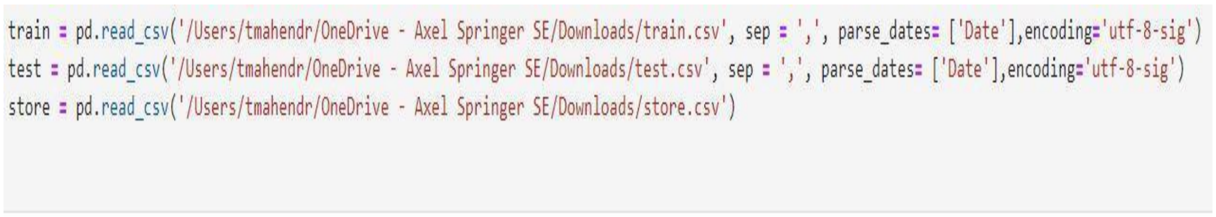
| | |
|---|---|
| **Analyze Data** | PAL-Predictive analysis tool and Python, Jupyter IDE is used for analyzing the data. We have divided the analysis in to Descriptive and Predictive. Whereas in Descriptive analysis gives past and current state of the store. (eg: how many products have been sold or bought by customers). The predictive analysis seeks to give you a glimpse into the future. It takes existing data and applies statistical techniques. Furthermore, information is explained briefly below. |
| **Store and Visualize data** | The Results of analyses is stored and Visualize data in SAPUI5. MVC architecture is used for developing user interfaces. MVC design pattern disconnects these major components allowing for efficient code reuse and parallel development. It is explained in more details below part of the document. |

## 2. Technical Interface

This section is the main part of the project, where the data is modelled, cleaned and is then used for the analysis. This will be implemented using Python, Jupyter IDE. In the below sub sections the process will be explained in a detailed way.

### 2.1 Procedure

Code snippet for loading the Rossmann Dataset csv. Since, we obtained a dataset which is pre-cleaned. We started of with the analysis.



```
train = pd.read_csv('/Users/tmahendr/OneDrive - Axel Springer SE/Downloads/train.csv', sep = ',', parse_dates= ['Date'],encoding='utf-8-sig')
test = pd.read_csv('/Users/tmahendr/OneDrive - Axel Springer SE/Downloads/test.csv', sep = ',', parse_dates= ['Date'],encoding='utf-8-sig')
store = pd.read_csv('/Users/tmahendr/OneDrive - Axel Springer SE/Downloads/store.csv')
```

Figure 1 Loading the Rossmann Dataset csv

### 2.1.1 Descriptive Analysis

- This code snippet shows the histogram of Sales when the store is open.

```
train.query('Open == 1')[['Sales']].hist(bins=100, figsize=(8,4), xrot=45, sharey=True, color = 'orange');
```

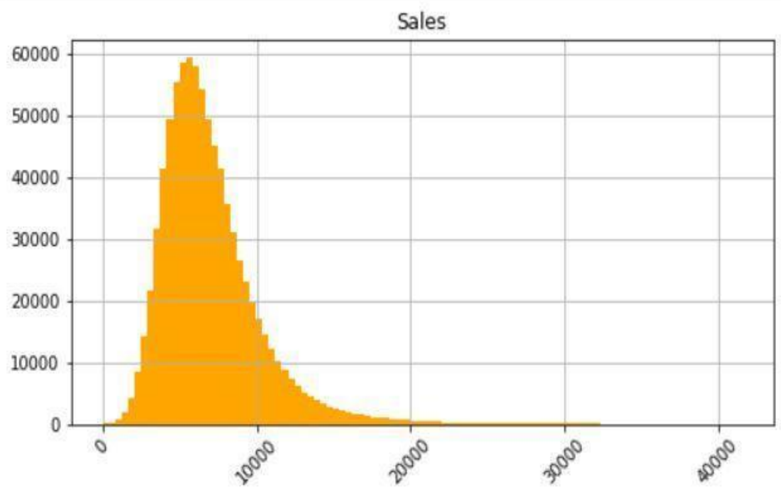


Figure 2 Sales when the store is open

- This code snippet shows the histogram of Customers when the store is open.

```
train.query('Open == 1')[[ 'Customers']].hist(bins=100, figsize=(8,4), xrot=45, sharey=True);
```

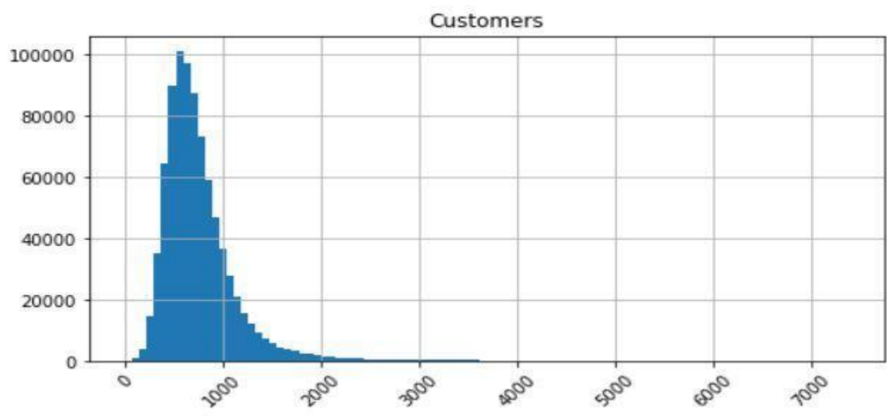


Figure 3 Customers when the store is open

- This code snippet shows the analysis of random store sales with dates.

```
fig = plt.figure(figsize=(8,6))

rand_stores = np.random.randint(0,1115, 20)
for i in rand_stores:
    data = train[(train["Store"] == i)][['Date','Sales']]
    plt.plot_date(data.Date, data.Sales,'-', alpha=0.5)

fig.autofmt_xdate()
```

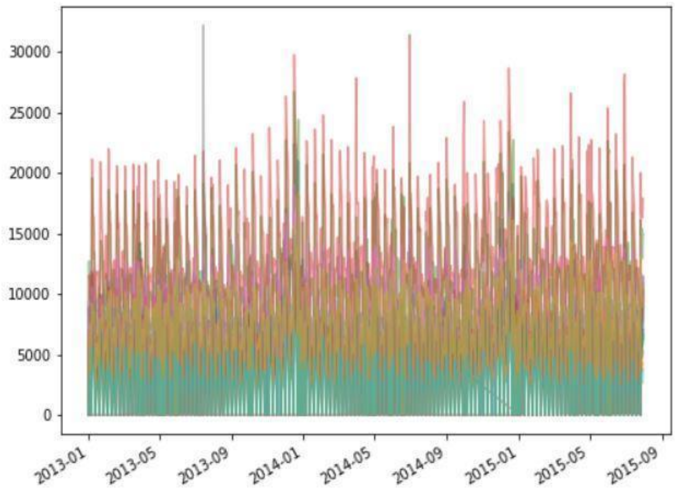


Figure 4 Store sales vs dates

- This code snippet shows the analysis of random store customers with dates.

```
fig = plt.figure(figsize=(8,6))

rand_stores = np.random.randint(0,1115, 20)
for i in rand_stores:
    data = train[(train["Store"] == i)][['Date','Customers']]
    plt.plot_date(data.Date, data.Customers,'-', alpha=0.5)

fig.autofmt_xdate()
```

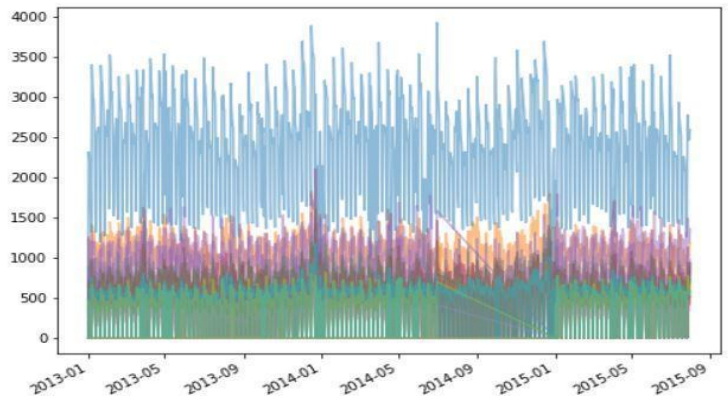


Figure 5 Store customers vs dates

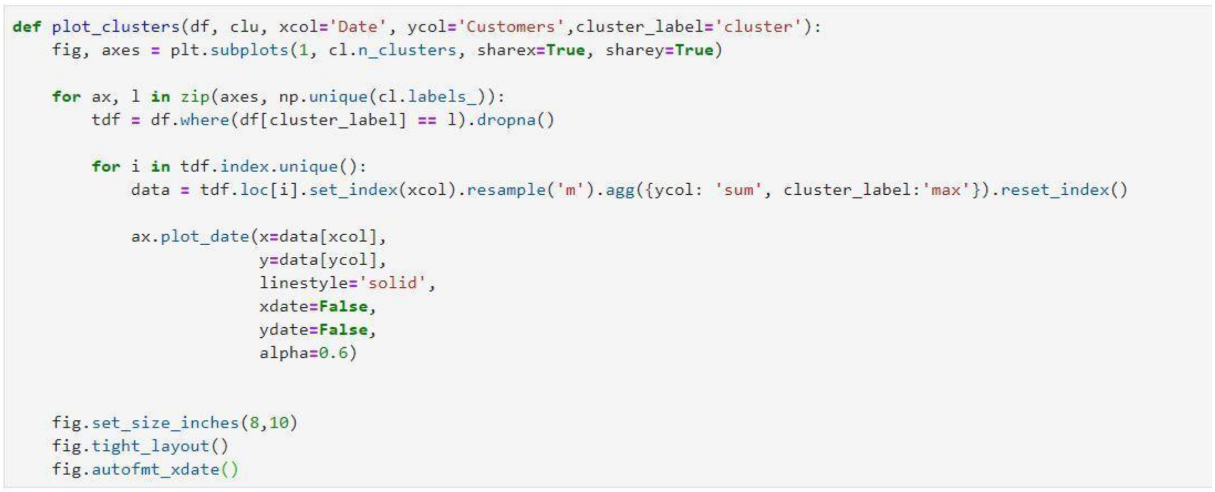
### 2.1.2 Predictive Analysis

1. Agglomerative Hierarchical Clustering**:** In this technique, initially each data point is considered as an individual cluster. At each iteration, the similar clusters merge with other clusters until one cluster or K clusters are formed.

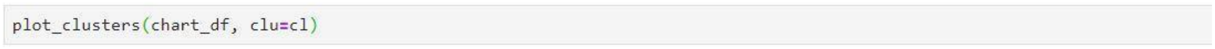
   The basic algorithm of Agglomerative is straight forward.

   a. Compute the proximity matrix.
   b. Let each data point be a cluster.
   c. Repeat - Merge two closest clusters, update the proximity matrix.
   d. Until only one single cluster remains ------------ [3]

- This code snippet shows the clusters of Customers for different stores with Date.



```python
def plot_clusters(df, clu, xcol='Date', ycol='Customers',cluster_label='cluster'):
    fig, axes = plt.subplots(1, cl.n_clusters, sharex=True, sharey=True)

    for ax, l in zip(axes, np.unique(cl.labels_)):
        tdf = df.where(df[cluster_label] == l).dropna()

        for i in tdf.index.unique():
            data = tdf.loc[i].set_index(xcol).resample('m').agg({ycol: 'sum', cluster_label:'max'}).reset_index()

            ax.plot_date(x=data[xcol],
                         y=data[ycol],
                         linestyle='solid',
                         xdate=False,
                         ydate=False,
                         alpha=0.6)


    fig.set_size_inches(8,10)
    fig.tight_layout()
    fig.autofmt_xdate()
```



```python
plot_clusters(chart_df, clu=cl)
```

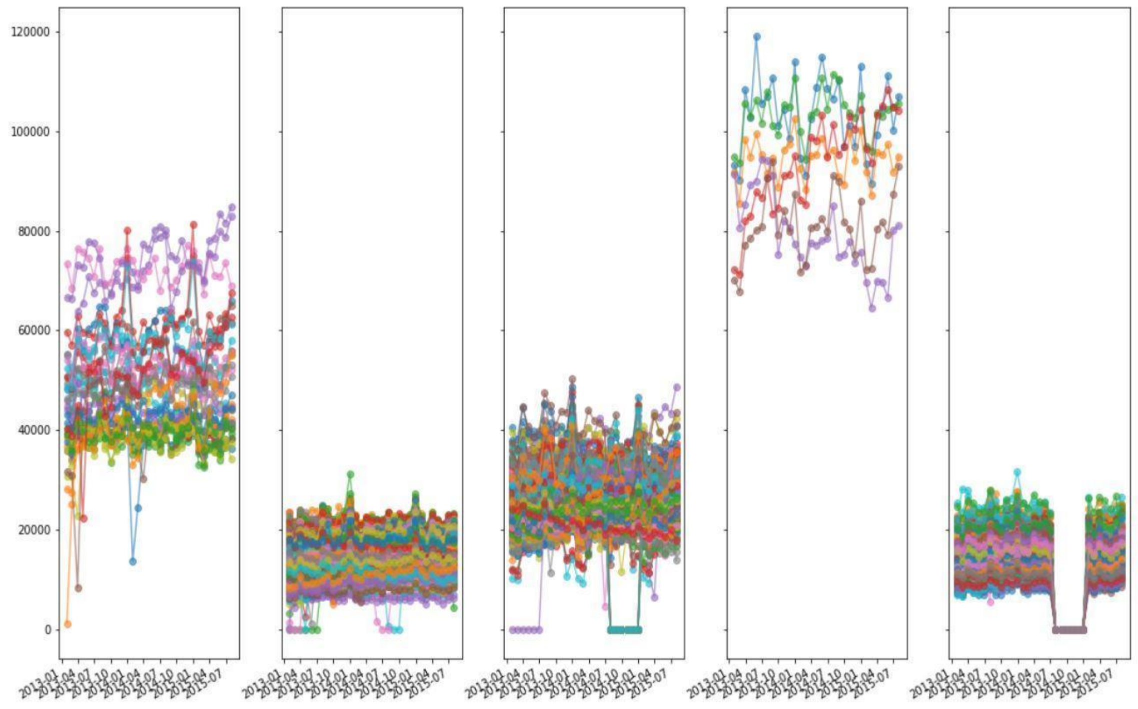


**Figure 6 Customers for different stores with date**

2. Time Series Forecasting: Time series is a collection of data points collected at constant time intervals. These are analyzed to determine the long term trend so as to forecast the future or perform some other form of analysis. But what makes a Time series different from say a regular regression problem? There are 2 things:

   a. It is time dependent. So the basic assumption of a linear regression model that the observations are independent doesn't hold in this case.
   b. Along with an increasing or decreasing trend, most TS have some form of seasonality trends, i.e. variations specific to a particular time frame. For example, if you see the sales of a woollen jacket over time, you will invariably find higher sales in winter seasons. ------------ [2]

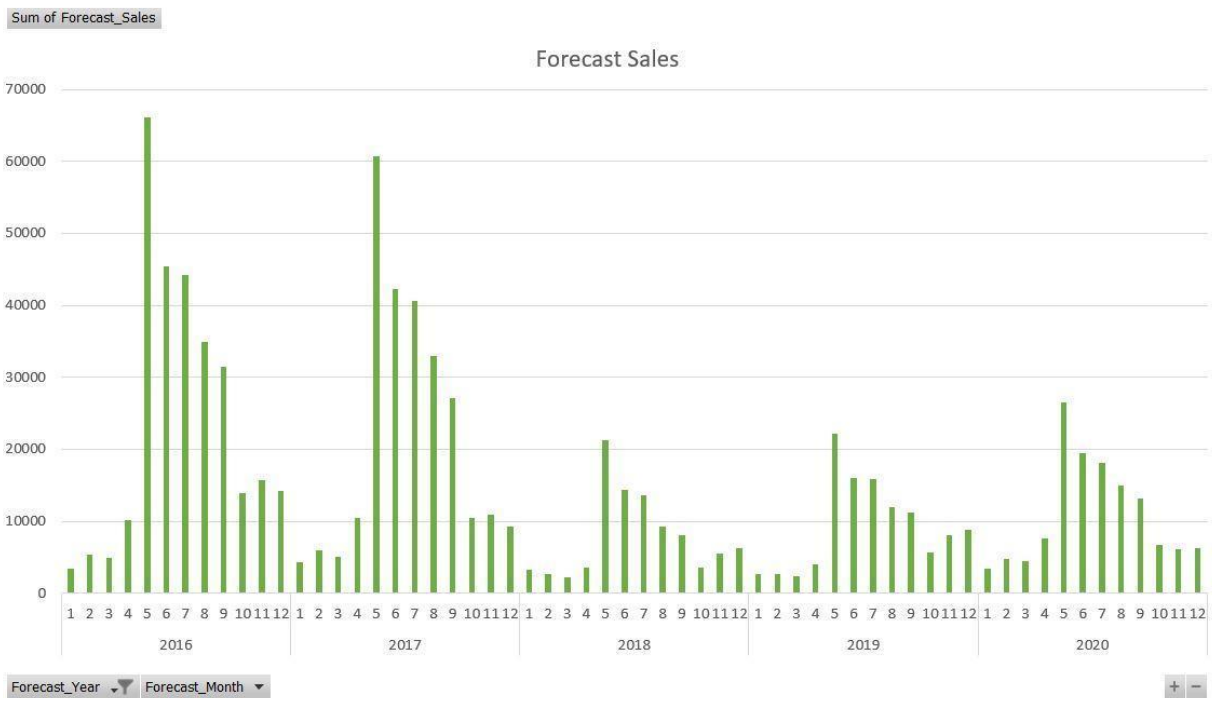


Figure 7 Sum of Forecast sales

## 3. Data Management and Data Model

This section shows how we integrated our dataset and tools to model the data. We have three train, test and store CSV files for different Rossmann Stores. The data is imported from the .csv file to the editor. The data is then added to the catalog using .hdbtable and .hdbti files. The data remains in the repository until it is activated. Once activated, the data is transferred to the catalog.

.hdtable defines the structure and schema of the table to be created.

For Example:

```
table.schemaName = "GBI_011";
table.tableType = COLUMNSTORE;
table.columns = [
{name = "Id"; sqlType = NVARCHAR; length = 10; },
{name = "Store"; sqlType = NVARCHAR; length = 20; },
{name = "DayOfWeek"; sqlType = INTEGER; },
{name = "Date"; sqlType = DATE; },
{name = "Sales"; sqlType = INTEGER; },
{name = "Customers"; sqlType = INTEGER; },
{name = "Open"; sqlType = NVARCHAR; length = 1; },
{name = "Promo"; sqlType = NVARCHAR; length = 1; },
{name = "StateHoliday"; sqlType = NVARCHAR; length = 2; },
{name = "Year"; sqlType = INTEGER; },
{name = "Month"; sqlType = INTEGER; },
{name = "Day"; sqlType = INTEGER; },
{name = "WeekOfYear"; sqlType = NVARCHAR; length = 35; },
{name = "Salespercustomers"; sqlType = DECIMAL; precision = 17; scale = 2; },
{name = "StoreType"; sqlType = NVARCHAR; length = 1; },
{name = "CompetitionDistance"; sqlType = INTEGER; },
{name = "CompetitionOpenSinceMonth"; sqlType = INTEGER; },
{name = "CompetitionOpenSinceYear"; sqlType = INTEGER; }
];
table.primaryKey.pkcolumns = ["Id"];
```

**Figure 8 Schema .hdbtable**

## Calculation view

When data is added to the catalog, we create a calculation view of type Dimension. In projection the gbi-student-011. Rproject table is added all the columns from input data is mapped to the output data. Then join the projection to the semantics label the “Id” as primary key for calculation view.

Save the calculation view. Run the view to check the contents.

# 4. SAPUI5 Implementation

SAP UI5 is a JavaScript application framework which follows MVC architecture. As the name indicates, it comprises of Model, View and Controller layers.

## Model:

1. The model is the central component of the pattern. It expresses the application's behavior in terms of the problem domain, independent of the user interface. It directly manages the data, logic, and rules of the application.

2. A model stores data that is retrieved according to commands from the controller and displayed in the view.

## View:

1. A view can be any output representation of information, such as a chart or a diagram. Multiple views of the same information are possible, such as a bar chart for management and a tabular view for accountants.
2. A view generates new output to the user based on changes in the model.

## Controller:

1. The third part, the controller, accepts input and converts it to commands for the model or view.
2. A controller can send commands to the model to update the model's state (e.g., editing a document). It can also send commands to its associated view to change the view's presentation of the model (e.g., scrolling through a document).

The data binding is done with the help of OData Services which is presented to the user using SAP UI5. An example of data binding with respect to the OData Services are described in. xsodata file in the package.

```
"GBI_011"."Forecast_011 " as
"ForecastSales"
key("Id")
aggregates always;
```

The above example shows the connective from SAPUI5 to OData services where the data is being fetched and displayed. The XS Engine manages this by binding the data and the user interfaces together. The XS Engine is a small application server which helps in building the GUI in SAP HANA.

The index.html helps in building the application and structuring them. The MainView.xml in the control package holds the tabs used for displaying on the first page. Each tab has different XML files to display the fetched data and the analytic result images which are displayed using the library of carousel. The Descriptive analysis and the predictive Analysis tab are mainly used for this purpose to display the carousel image. The data which has been cleaned and exported into SAP HANA contains the forecasted data which is fetched to the frontend and displayed in the tab Forecasted Data. Each JavaScript file contains the action of each XML file.

Below are few steps on how the GUI was created.

- Create a package under gbi-student-011 in the Editor as shown in the image below and name the project, in our case “Rproject”.

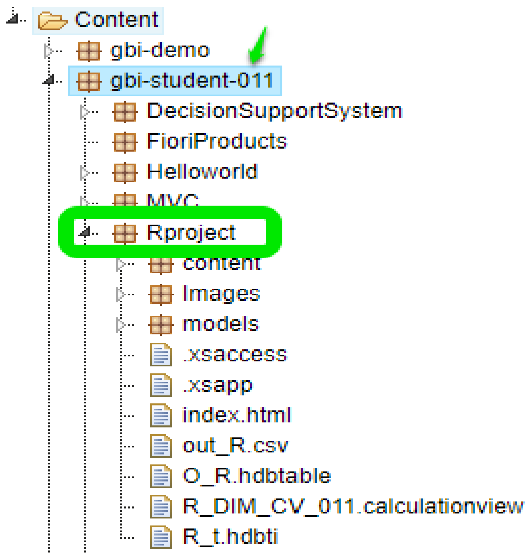
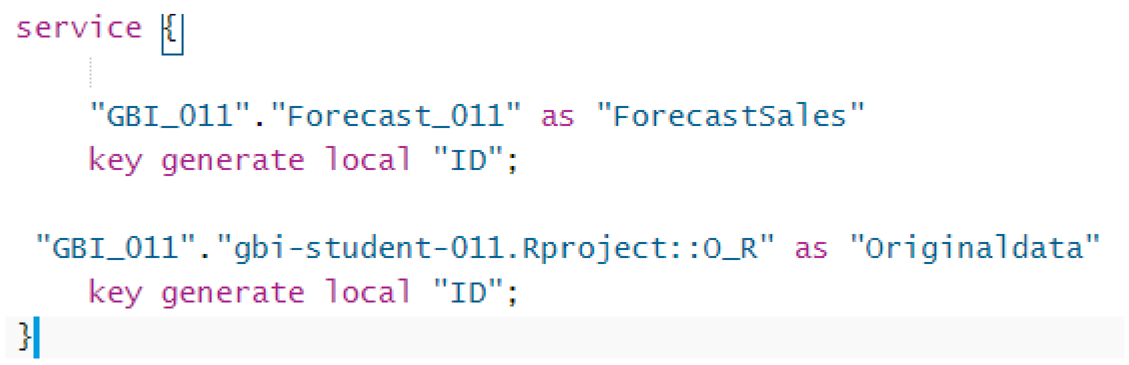


**Figure 9 Editor Image for Rproject**

- Once the package is created add the model and the content package, where the .xsodata file, .js, and .xml files are created.
- The index.html starts the application in a webpage and all the libraries are loaded in this file.

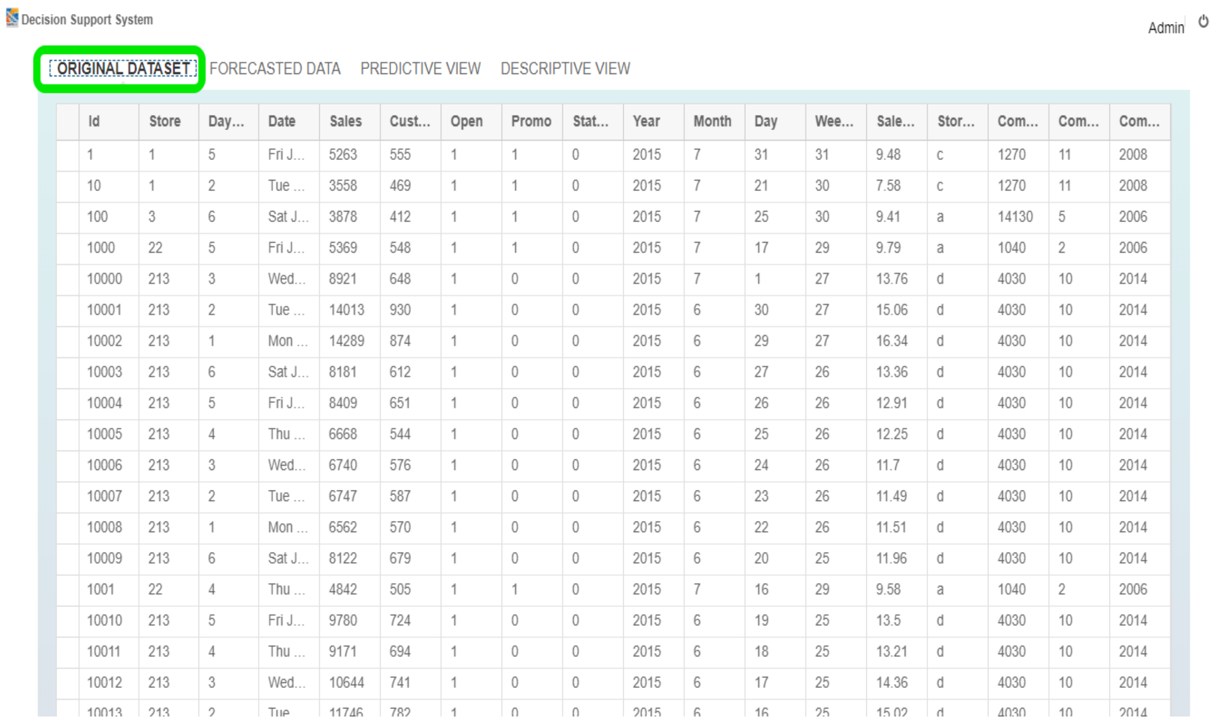
- The GBI.xsodata contains the table that is being accessed from the catalog and displayed in the GUI.

According to our use case, we have used the below schema to obtain the raw cleaned data which is then displayed in the front end.



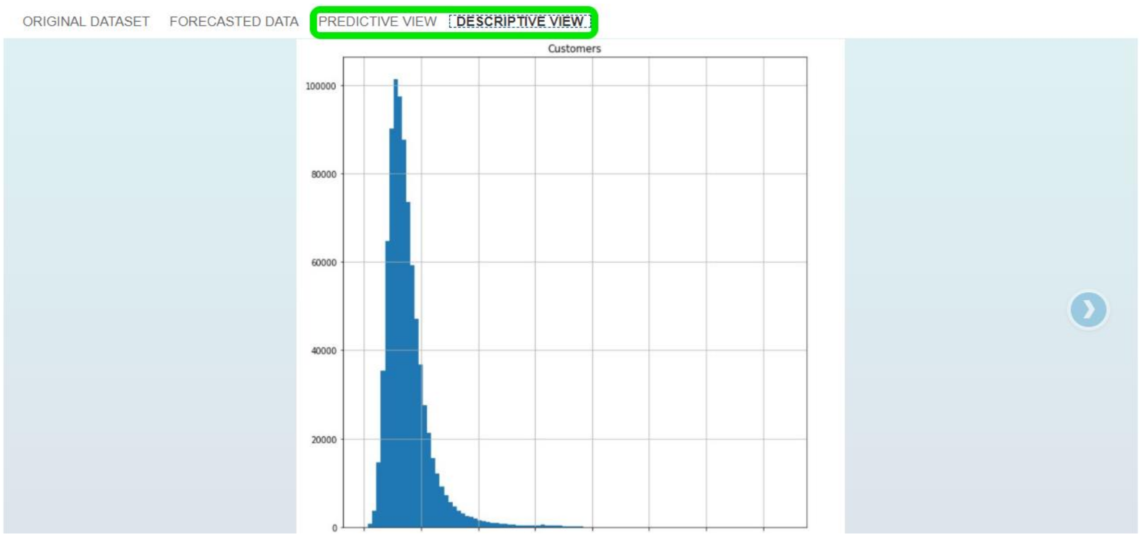
```
service {

    "GBI_011"."Forecast_011" as "ForecastSales"
    key generate local "ID";

 "GBI_011"."gbi-student-011.Rproject::O_R" as "Originaldata"
    key generate local "ID";
}
```

- In the Content package, the MainView.xml contains the first page to be displayed in SAP HANA and the necessary tabs. The MainView.js provides action for the elements in the MainView.xml
- The MainView.xml also contains the key to each of the tabs which are represented in another, XML and their respective .js files for the event handling.
- The Original Data.view.xml and Original Data.controller.js display the raw data of Rossman. this is done by using tables to display the data in the form of rows and columns.



| Id | Store | Day... | Date | Sales | Cust... | Open | Promo | Stat... | Year | Month | Day | Wee... | Sale... | Stor... | Com... | Com... | Com... |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 5 | Fri J... | 5263 | 555 | 1 | 1 | 0 | 2015 | 7 | 31 | 31 | 9.48 | c | 1270 | 11 | 2008 |
| 10 | 1 | 2 | Tue ... | 3558 | 469 | 1 | 1 | 0 | 2015 | 7 | 21 | 30 | 7.58 | c | 1270 | 11 | 2008 |
| 100 | 3 | 6 | Sat J... | 3878 | 412 | 1 | 1 | 0 | 2015 | 7 | 25 | 30 | 9.41 | a | 14130 | 5 | 2006 |
| 1000 | 22 | 5 | Fri J... | 5369 | 548 | 1 | 1 | 0 | 2015 | 7 | 17 | 29 | 9.79 | a | 1040 | 2 | 2006 |
| 10000 | 213 | 3 | Wed... | 8921 | 648 | 1 | 0 | 0 | 2015 | 7 | 1 | 27 | 13.76 | d | 4030 | 10 | 2014 |
| 10001 | 213 | 2 | Tue ... | 14013 | 930 | 1 | 0 | 0 | 2015 | 6 | 30 | 27 | 15.06 | d | 4030 | 10 | 2014 |
| 10002 | 213 | 1 | Mon ... | 14289 | 874 | 1 | 0 | 0 | 2015 | 6 | 29 | 27 | 16.34 | d | 4030 | 10 | 2014 |
| 10003 | 213 | 6 | Sat J... | 8181 | 612 | 1 | 0 | 0 | 2015 | 6 | 27 | 26 | 13.36 | d | 4030 | 10 | 2014 |
| 10004 | 213 | 5 | Fri J... | 8409 | 651 | 1 | 0 | 0 | 2015 | 6 | 26 | 26 | 12.91 | d | 4030 | 10 | 2014 |
| 10005 | 213 | 4 | Thu ... | 6668 | 544 | 1 | 0 | 0 | 2015 | 6 | 25 | 26 | 12.25 | d | 4030 | 10 | 2014 |
| 10006 | 213 | 3 | Wed... | 6740 | 576 | 1 | 0 | 0 | 2015 | 6 | 24 | 26 | 11.7 | d | 4030 | 10 | 2014 |
| 10007 | 213 | 2 | Tue ... | 6747 | 587 | 1 | 0 | 0 | 2015 | 6 | 23 | 26 | 11.49 | d | 4030 | 10 | 2014 |
| 10008 | 213 | 1 | Mon ... | 6562 | 570 | 1 | 0 | 0 | 2015 | 6 | 22 | 26 | 11.51 | d | 4030 | 10 | 2014 |
| 10009 | 213 | 6 | Sat J... | 8122 | 679 | 1 | 0 | 0 | 2015 | 6 | 20 | 25 | 11.96 | d | 4030 | 10 | 2014 |
| 1001 | 22 | 4 | Thu ... | 4842 | 505 | 1 | 1 | 0 | 2015 | 7 | 16 | 29 | 9.58 | a | 1040 | 2 | 2006 |
| 10010 | 213 | 5 | Fri J... | 9780 | 724 | 1 | 0 | 0 | 2015 | 6 | 19 | 25 | 13.5 | d | 4030 | 10 | 2014 |
| 10011 | 213 | 4 | Thu ... | 9171 | 694 | 1 | 0 | 0 | 2015 | 6 | 18 | 25 | 13.21 | d | 4030 | 10 | 2014 |
| 10012 | 213 | 3 | Wed... | 10644 | 741 | 1 | 0 | 0 | 2015 | 6 | 17 | 25 | 14.36 | d | 4030 | 10 | 2014 |
| 10013 | 213 | 2 | Tue | 11746 | 782 | 1 | 0 | 0 | 2015 | 6 | 16 | 25 | 15.02 | d | 4030 | 10 | 2014 |

- Similar to the ORIGINAL DATASET tab the FORECASTED DATA was also displayed with the “ForecastSales” schema from GBI.xsodata which returns the cleaned data set necessary for forecasting.
- The PREDICTIVE VIEW and the DESCRIPTIVE VIEW  is controlled by the Analytical Image.view.xml, the Analytical Image.controller.js. and Descriptive View.view.xml and Descriptive View.controller.js respectively.
- Which displays the images stored in the package “Images” using a carousel as a styling element.



Hence the entire frontend is done using SAPUI5 to display the data and the images. Below is the analysis report has done automatically by SAP HANA when the input data is provided. We have used the forecasted data to display the graph within SAP HANA using flattened datasets. But in our case, we decided to use Python as we had hands-on experience with it.

## 5. Functional Requirements

## 5.1 Overview

The changes made to the SAP HANA database (Which now contains the Rossmann data) such as insert, delete, and update, etc., needs to be backed up into the backup node. These changes are logged into the staging store by using the information from the log parser.

### 5.1.1 Database Entities

train table, test table, and store table.

### 5.1.2 Pre-Execution Processes

We make sure that there are no errors in the process and write statements needed for the update, delete and join.

### 5.1.3 Post-Execution Processes

Post-execution an SQL file and table is created that contains the data necessary for our analysis.

### 5.1.4 Transformation Details

The three tables train, test, and store are cleaned in Excel. The three tables are transformed into one table using Python, Jupyter IDE, all duplicates were removed and stored the data.

## 5.2 Corporate Store Load

### 5.2.1 Process Description

Users who need access to the project (Users who wish to view our analysis or a developer who wants to improve the application) will be given unique user credentials which will be authenticated on the cloud platform.

### 5.2.2 Variables and Parameters

User credentials / parameters such as User ID and password is required to log   in to the application.

### 5.2.3 Database Entities

train table, test table, and store table.

### 5.2.4 Pre-Execution Processes

Each user who needs access to the application has to be provided with a unique user ID. The users are then allowed to set their passwords.

### 5.2.5 Post-Execution Processes

The user logs out from the application once the process is complete.

### 5.2.6 Process Description

The users are first given the credentials needed to login to the application. Once the credentials are set, if the authentication is successful, the user can log in into the application. The session can be closed by logging out of the application.

## 6. Non-Functional Requirements

### Performance

As the world is so globalized and has become fast and rapid, the performance places a crucial role in the decision making for the proper growth of the competitive market. The performance can be defined in several ways. It depends on the output of our analysis. It helps us to predict how efficient is our application in predicting the appropriate result and in turn how well it will prove to be a good decision making tool to helps client forecast their requirements. Data model has shown good performance for the data set which was used for our project.

### Availability

Availability in terms of our project was that how much of the data was used after the final implementation process? Does the data which was obtained was valid and useful or not and whether it was providing appropriate information to the client or end users in terms of being useful to them in decision making? Predictive Analysis processed along with the data model was used to define the amount of data generated and also its capability. Data generated in our case is highly proficient and relevant data.

### Maintainability, Adaptability and Portability

Whenever an application is developed it should be easy to maintain, should adapt to any changes and should be portable in order to run on any platform. Our application is developed to adapt and show good performances with respect to changes in data with no unpredicted errors.

# 7. Security

## Communication channels

Security is one of the biggest processes any organization creating any application should consider. There are many types of security concerns one should take into account. By default, the communication channels are not secure. To secure them, we use certain protocols and channels such as Secure Sockets Layer (SSL) protocol. SSL is a cryptographic protocol that provides security and data integrity for communication over TCP\IP networks. The way SSL works is by the property of handshaking among client and server by initiating the further processes.

## Application-specific security

The application access is restricted to only valid users having a “Valid User Identification” and “Password”. They can login to the application with their credentials and can perform any valid operations on the data (such as read, write and delete), more often the developers who are working on the project will be using or have been granted (all) read, write and the delete operations to cope up with the needs of the requirements. Any members other than the project members who are involved indirectly to the project would have less access and cannot perform all the operations and would be advised to use the application for analysis or evaluation purposes only.

## 8. References

[1] S. W. Hart, “Temp_FD”.

[2] “Analytics Vidhya,” 2019. [Online]. Available: https://www.analyticsvidhya.com/blog/2016/02/time-series-forecasting-codes-python/.

[3] “Towards Data Science,” 2018. [Online]. Available: https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec.