DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Documentation Advanced Practical Course: Developing innovative services at the example of SAP technologies

Author:	Paromita Banik, Cristina Bobes, Zixi Chen
Supervisor:	Clemens Drieschner, Marwin Shraideh
Chair:	Chair of Information Systems (I17)
Submission Date:	15.07.2021

Contents

List of Figures

1. Introduction

1.1. Motivation

Since the early 2000's scientists tried to combine the techniques learned from the fields of data mining, statistics, artificial intelligence, machine learning and operation research in order to improve the decision making process, uncover financially valuable patterns and optimize industrial systems. The field containing all of this knowledge together is called data analytics where hidden information in the data can be discovered and future events can be forcasted by analysing large datasets [1] [2].

Sales forecasting, as the name suggests, is the process of estimating future sales based on historical sales data and influencing factors such as holidays, promotions and competitions. Accurate sales forecasts enable companies to make proper business decisions and planning which not only helps to successfully meet customer needs but also to balance supply and demand and reduce cost and waste in the business process. By understanding how various factors impact sales amount and by foreseeing patterns and trends in sales, companies are able to evaluate and change their business strategies to achieve better outcomes. An intuitive example of this could be, knowing how holidays affect sales, companies can give discounts on their products in that certain time period to promote sales. Sales forecasting is therefore of crucial importance in business planning and a lot of major strategies and decisions heavily depend on these results. This is a reason why it often becomes a matter of concern for companies to find a tool with best predictive functionalities that would give them the most accurate sales forecasts.

Our industry partner NTT Data challenged us to develop a use case based on a scenario where a customer is operating over 3000 stores in 7 European countries and their store managers are asked to predict their daily sales up to 6 weeks in advance. With the store sales influenced by the many different factors (holidays, competition, promotions etc.) and with thousands of individual managers predicting sales based on their unique circumstances, the accuracy of results of the sales forecasts can be quite varied, and therefore, a tool with strong predictive functionalities is required. Therefore, the goal of this project is to build a predictive model in SAP Analytics Cloud (SAC) and SAP HANA XSA and compare the two SAP solutions in terms of their features, limitations and predictive functionalities to find which one suits best for data analytics projects.

Figure 1.1 presents the overview of our project.

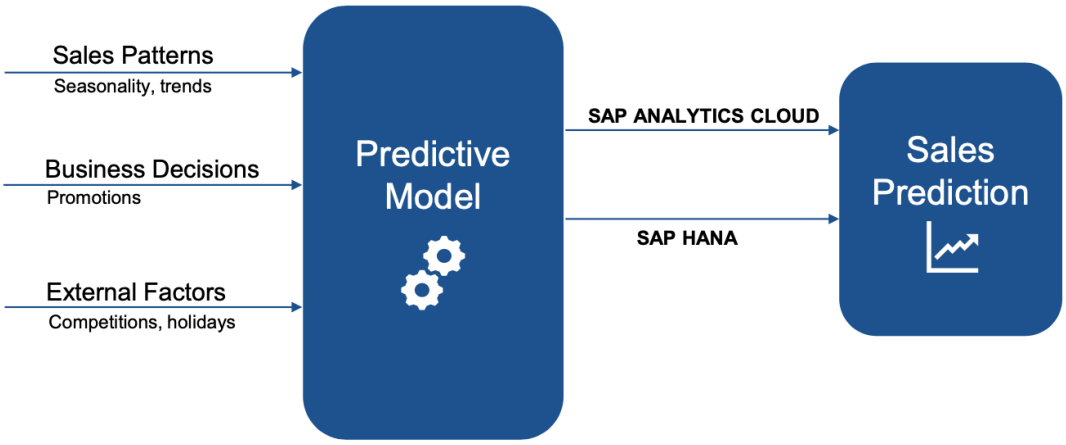


Figure 1.1.: Overview of the project

1.2. Planned Approach

In order to build predictive models in both SAC and SAP HANA and to compare the capabilities of both platforms, we tested them on the same use case and datasets provided by our industry partner NTT Data. Since we had no prior knowledge of SAC or SAP HANA, we planned to approach this project by first researching the features and functionalities of both SAP products. Therefore, we learned which features and functionalities are provided to solve data analytics tasks and also understand which machine learning models are supported. These insights helped us plan a solution architecture which is described in detail in the next section.

In order to begin with the data analysis in this project, we divided our task into four stages for each platform:

1. Data Preprocessing
2. Correlation Analysis for Feature Selection
3. Build and train predictive models
4. Visualization of the results

In the data preprocessing stage, we clean and prepare the raw data into an understandable and usable format. In the correlation analysis stage, we discover the dependencies between the variables to find which variables are related to the target variable in order to exclude them from the later modelling process. In the stage of building predictive model, we use machine learning algorithms and predictive functionalities to build the models which will predict the target variable (in our case is 'sales'). In the visualization stage, we display the predictions of the models created in each platform through charts to show the results.

Lastly, we planned to describe how both SAC and SAP HANA performed throughout this entire experiment by stating their limitations. It is important to indicate any technical limitations from the first stage of data preprocessing to the final stage of visualization. We also demonstrate how the predictive tools provided by SAC performed in comparison to our Python data analysis script incorporated in the SAP HANA solution.

1.3. Planned Solution Architecture

1.3.1. For SAC

After a detailed research and understanding of what features of SAC can be helpful for our planned approach, we split the architecture in SAC as backend and frontend as depicted in Figure 1.2. The stages 1-3 enumerated in Section 1.2 are implemented in the backend part and the visualizations in stage 4 are presented in the frontend. In order to clean and prepare the data in stage 1, we planned to use the feature Smart Transformation provided by SAC. Smart Transformation can be used to update values, sort, delete, combine and split columns to better prepare data for modelling and visualizations. For stage 2, we planned to use Smart Discovery in SAC as an alternative to the correlation analysis done in SAP HANA. Smart Discovery is a powerful feature in SAC that runs automated machine learning algorithms to discover the impact various variables have against the set target metric. To create the predictive model in stage 3, we planned to use the Time-Series Forecast Predictive Scenario from the Smart Predict feature in SAC that automatically generates a forecasting model from historical data with time dimension and therefore fits best for our use case.

For the frontend part, which takes place in stage 4, we planned to use the SAC Dashboard Story for visualizing the data and displaying the result of the forecast. These stories are smart features offered by SAC which provide powerful data visualizations with charts, graphs, tables and other visual elements to help users discover insights hidden within the data.

1.3.2. For SAP HANA

After researching the features and functionalities provided by SAP HANA, we found out that SAP HANA develops towards supporting all kinds of fullstack projects and therefore is open to the cooperation with the Python language as well. Python is a very powerful tool for data analysis, because it offers ready to use libraries with implemented machine learning methods and data transformation techniques, leading it to be one of the most popular languages for these kind of projects. Therefore, our approach is to implement the required data analysis in a Jupyter Notebook, using the Python language. We chose to write the script in Jupyter Notebook, because we could test each part of the code independently, by diving it in different cells. This way, we could save time by compiling one step at a time instead of the whole script at once. The data source for the data analytics task solved in Python was provided by SAP HANA.

This means, the stages 1-3 enumerated in the Section 1.2 will be done in the backend using Python and the frontend part of this solution will include the visualization of the results and

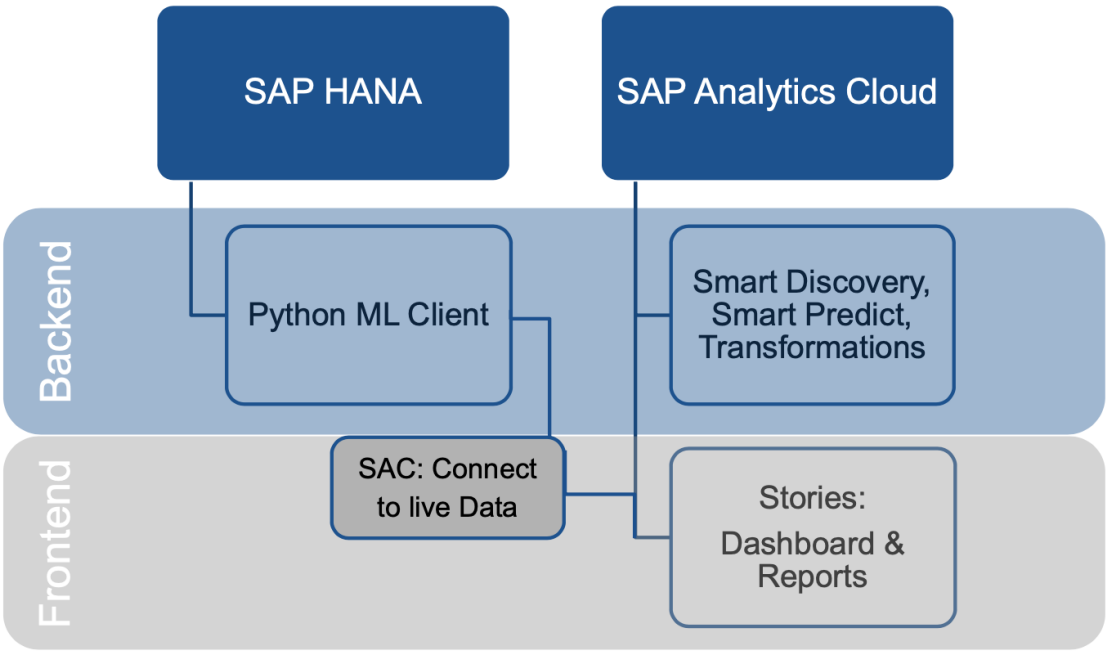


Figure 1.2.: Planned Solution Architecture

will be presented in SAC, as it can be seen in the architecture depicted in Figure 1.2.

The data exchange between SAP HANA and the Python script will be managed using REST API, which should make the tables accessible to this external script. The results of the analysis should be saved in SAP HANA after completion and afterwards imported to SAC through the Connect to live Data feature provided by SAC.

1.4. Project Plan

We set 8 milestones for our project based on our planned approach in Section 1.2 to accomplish our goal. Our first milestone was to research SAC and SAP HANA in detail to understand how they can be used and what functionalities they have to process data, build models and create visualizations. The second milestone was to finish preprocessing of the data to guarantee data quality. The third milestone was to study the correlation between the variables and determine the variables that highly influence the target variable. After all the cleaning and preparing of the data, the fourth milestone was to build the Time-Series Forecast in SAC. The fifth milestone was then to create UI as a dashboard in SAC to visualize the predictions and in parallel to build the predictive model and carry out analysis in Python script for SAP HANA. The sixth milestone was to connect the Python script to the data source in SAP HANA by using Machine Learning APIs of HANA. The seventh milestone was to create a dashboard again in SAC to present the visualizations of the results from SAP HANA. Finally,

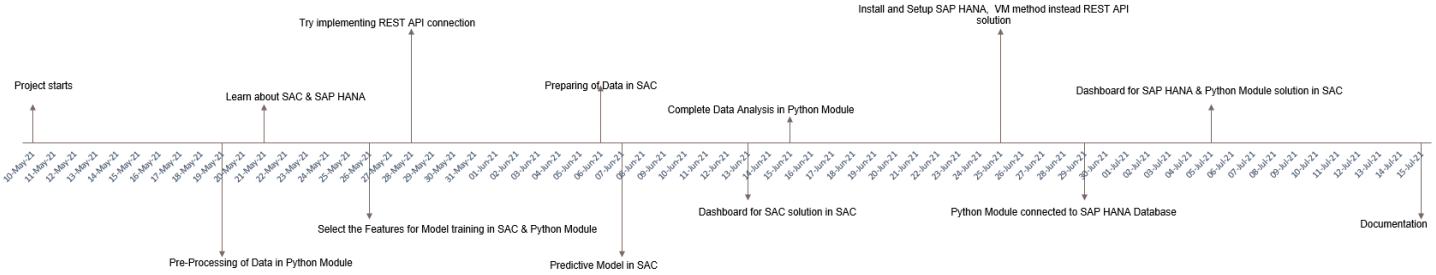


Figure 1.3.: Planned Solution Architecture

the eighth milestone was to document all our findings and the difficulties we faced and how we overcome them.

However, due to some dead-ends faced during the project, some of our milestones got adapted according to our requirements. We learned in an early stage of this project about the possibility to use REST API to connect SAP HANA to the Python script, hence this was one of our planned milestones. Later, when this did not work for us, we went for our alternative approach to install and setup SAP HANA using a virtual machine and then connect the Python module to the SAP HANA database. Figure 1.3 presents the additional milestones and our progress throughout the project.

1.5. Limitations and Setbacks

1.5.1. For SAC

During this project, we came across a lot of limitations and complications in SAC which lead us to taking longer time for implementation than expected. Although SAC has many in-built automatic features for data analysis, most of these features are limited to a specific number of rows of data or to a specific function or requirement. This resulted in a few complicated workarounds for which SAC should have simply provided support, so that the whole process can run smooth without any other external tools. These limitations and workarounds are documented in detail in Chapter 4. The most prominent setback we faced was that our data was not usable for the Time-Series Forecast in SAC which was our initial plan (refer to Section 6.1.1). Therefore, we had to change our direction to the Regression predictive model in SAC as an alternative solution.

1.5.2. For SAP HANA

When we first started learning through different tutorials how different developers recommend doing Python Data Analysis using SAP HANA, we saw that we only need some credentials in order to access the data stored in SAP HANA. Only after we did most part of the analysis in the Python module and started trying to request data from HANA, we realized that it was not that simple, as it was presented in the tutorials. We did not have the

sufficient credentials because we did not setup the system and hence, we were not aware of where to find them. Then we tried to create a REST API connection using NodeJs, but we did not succeed as we had multiple building errors. Therefore, we learned that we can install the whole environment using virtual machines and we built our solution based on this knowledge. Chapter 6 presents some of these errors in more detail.

1.6. Design Thinking Workshop

The Design Thinking Workshop was the earliest stage in our project, where we were introduced to our topic by our industry partner and were given the opportunity to untangle the problem and brainstorm some ideas for solution. The one solution we brainstormed and decided to move forward with, was to create a dashboard with a menu. This should support our persona – the store manager – to get an overview of the statistics and predictions of the sales in form of visualizations and therefore, find support for future decisions and plans. In our project, we persisted to the idea of presenting the results of our predictive model in a dashboard for both SAC and SAP HANA.

Some additional ideas we had for the dashboard were to include statistics about expiring products, reduced stocks and inventory space. However, these information were not included in our given data and therefore were not possible to implement. Another very useful feature we thought of during the design thinking workshop was to implement alerts/sms/emails with weekly reports and summaries of sales for the store manager. However, in the later phase, we realized that this feature would not make sense to implement because our data was not real-time data and also implementing it was out of the scope of this project and time limit.

2. Theoretical Background

2.1. Introduction to Data Analytics

The field of data analytics provides a decision support based on the analysis of large data sets with the help of computer systems. It is an interdisciplinary field which combines knowledge from statistics, machine learning, pattern recognition, operation research and artificial intelligence [3]. Predictive analytics is a subfield of data analytics that consists of creating and using models that make predictions by uncovering patterns in the data. Predictions not only uncover what will happen in the future but their role in data analytics can also be to determine unknown values for variables based on stored information with support of, for example, a reliable model trained with machine learning techniques. By using a supervised machine learning method, data analysts can create models, which automatically learn to find the relationships between a set of features and one or more targets. If the model shows a good performance after the training process, it can be used as a reliable tool to predict the values of the target variables for new input values for the features [2].

A data analysis process is typically divided into four phases. The first phase is Preparation and usually consists of collecting or accessing the data from a data source. The second phase is Preprocessing which is related to the cleaning or filtering of data to make the data appropriate for analysis. The third phase is Analysis where the data is analyzed by building a model using machine learning techniques. The fourth and final step is Postprocessing where the results from the analysis are interpreted and evaluated [3].

One real-world application of predictive analytics is Sales Forecasting. It is the process of estimating future sales based on analysis of historical sales during a defined period in the future, for example, in the next week, month, quarter or year. This particular technique is also known as Time series analysis as the predictions are based on historical data with a time dimension. Machine learning can be used to predict sales by analyzing patterns in historical data based on the external (holidays, trends, seasonality) and internal (promotions, competition) factors that affect sales [2]. In this case, the target value for the machine learning model is the sales which needs to be predicted and the set of features are the factors that influence sales.

2.2. Correlation Analysis

Correlation Analysis in data analytics is a technique to measure the direction and strength of the relationship between the variables. It defines the pairwise relationship between circumstances, objects or between mathematical or statistical variables which tend to vary, be

associated, or happen together in a non-coincidentally way [4].

The statistical measure that determines the direction and strength of the relationship between two variables is called the correlation coefficient. The correlation coefficient value ranges from -1 to 1 where a value of -1 shows a perfect negative correlation, while a value of 1 shows a perfect positive correlation and a value of 0 shows no correlation between the two variables. All the other values in between -1 and 1 indicate the degree of correlation, for example, a value of -0.2 shows a weak negative correlation while 0.8 shows a strong positive correlation [5].

Correlation Analysis can be used as a feature selection technique in the phase before training the machine learning model, because it uncovers dependencies between the inputs and output-variable. If variables show strong correlation to the target variable, they are no longer independent and have to be excluded from the list of features used for the training of the model.

2.3. Introduction to Machine Learning

The concept of Machine Learning originated in the 1950's when Arthur Samuel developed a computer game that became better at playing with the number of games played. As defined by him, "Machine learning gives computers the ability to learn without being explicitly programmed" [6]. Over the years, Machine Learning has evolved vastly through complex and big data and today we encounter its applications in everyday life through streaming services like Netflix or social media. In the context of data analysis, machine learning is a method to automate analytical model building. It involves the creation of computer programs that can learn from data, make predictions, and improve performance by gathering more data and experience.

There are two basic types of machine learning. A supervised learning approach is when the data consists of both the input and output values (also known as labeled data). By calculating the error from the difference between the model's predicted value and the actual output value it is possible to change the model's parameters to minimize this error. Unsupervised learning is used when there is no output data (also known as unlabeled data). Instead, the algorithm work on its own to discover the inherent structure of the data. With the input dataset used in this project and its numerical output value, it is clear that supervised learning is applicable for our project. As there are multiple time-dependent variables in the data (sales, promo, competition, holiday), the data is considered to be a multi variate time series data, where every variable depends not only on its past values but also has some dependency on other variables.

The first step in the machine learning modelling is to split the data into training, validation, and test datasets. Typical good splits are 80/10/10 or 60/20/20 depending on the dataset. Each dataset then undergoes a different phase as described below:

Training: This is the phase where the labeled data in the training dataset is used to train the machine learning algorithm to learn by pairing the input with the expected output.

Validation: This is the phase where the unseen data in the validation dataset is used to

evaluate the performance of the ML algorithm used on the training dataset. In other words, this the phase where the model is evaluated using an evaluation metric and the performance is improved by making any changes that is required.

Test: This is the final phase where the unlabeled, untouched, and unseen data in the test dataset is used to confirm that the ML algorithm was trained effectively and can make accurate predictions.

Different ML models are for different tasks and it is important to choose the correct approach for tackling the task appropriately. Since we have a multivariate time series data, there are certain algorithms that are suitable to our model for sales forecasting. The next section provides an overview of the ML algorithms used in SAC and SAP HANA to build the predictive model.

2.4. Regression Analysis

Regression is a supervised machine learning algorithm to predict a dependent variable with one or more independent variables or predictors. The dependent variables are conceived of as being a linear function of the independent variables, which are weighted by regression coefficients and then are added together:

$$y = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k + \epsilon$$

where y is the predicted value of the dependent variable, $x_1 \ldots x_k$ are the independent variables, $\beta_0 \ldots \beta_k$ are the regression coefficients (i.e. the effect that increasing the value of x has on y) and ϵ is the model error which determines how much variation there is in the estimate of y. A small error indicates less variation between the predicted and actual value and hence results in a better model and vice versa when the error is large.

2.4.1. Autoregression

Autoregression is a time series regression method that uses observations from previous time steps as input to a regression equation to predict the value at the next time step, for example:

$$y_t = \beta_0 + \beta_1 y_{t-1} + \ldots + \beta_k y_{t-(k-1)} + \epsilon_t$$

where y_t means y measured in time period t, $\beta_0 \ldots \beta_k$ are the model parameters and ϵ_t is the error for the regression. The model is written as AR(k) where k is the number of immediately preceding values in the series, which are used to predict the value at the present time. The relationship between the variables is then called autocorrelation:

$$Corr(y_t, y_{t-k})$$

where the value k is called the lag. More generally, a lag k correlation is the correlation between values that are k time periods apart. This algorithm is inherent in the time series predictive model in SAC along with regression.

2.4.2. Random Forest Regression

Random Forests use multiple decision trees/regression trees and merge their outputs, perceiving them, as if they were a committee of decision makers. This procedure is called ensemble learning and the main idea is, that if individual predictions are properly combined, the overall accuracy on average of the whole "committee" should be better than of any individual decision maker [7].

Random forests also uses the Bagging algorithm: each tree is generated from a different training dataset, which is a bootstrap of the original dataset. The resulted models are a combination of a uniform average or vote.

In order to increase the diversity between the classifiers, random forests also uses the Random Subspace Method at each branching point of the trees, to restrict each of them to work on a different random subset of size n, from the full feature set,

$$n = log_2(N + 1), \tag{2.1}$$

where N is the size of the whole feature set [7].

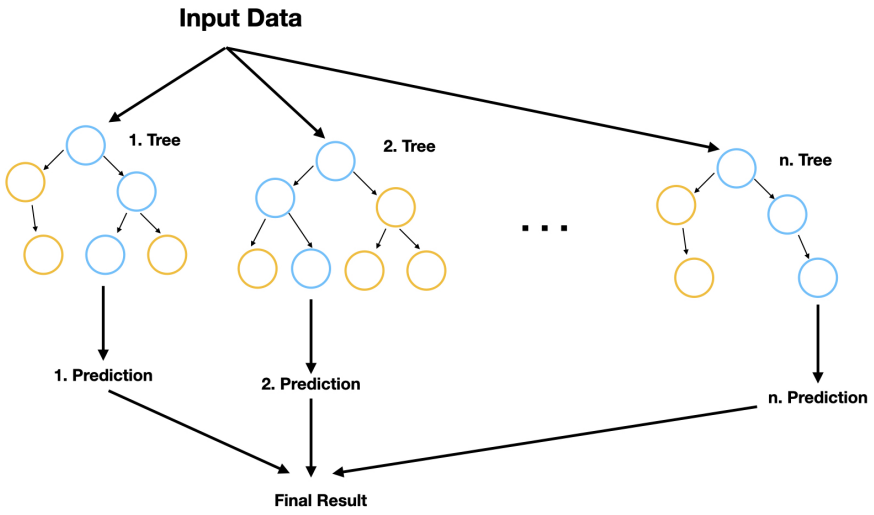


Figure 2.1.: An example of a random forest with n decision trees

This comes as a solution in case, there will be one very strong feature present in the dataset. By only using bagging for the decision trees, most or even all of the deciders would use this feature in the top split, resulting in almost all the trees looking in the end alike. This would damage the prediction accuracy overall, so random forests solve this problem by forcing each split to consider only a subset of the features by deploying the Random Subspace Method. This way, many of the splits will not receive this strong feature and the other "members of the committee" will have a chance to influence the final result [8].

2.5. Cross Validation

Cross-validation is mainly used in prediction context to evaluate the accuracy of the built predictive model. In this setting, a model usually has a training data set with known data and a testing data set with unknown data. The goal of cross-validation is to test the model's ability to predict new data that was not used in estimating it, in order to flag problems like overfitting or selection bias and give an insight on how the model will generalize to an independent data set. Among the various types of cross validation, k-fold cross-validation is mostly used. In k-fold cross-validation, the original sample is randomly partitioned into k equal sized subsamples. One single subsample will be selected to be testing data, and the remaining $k-1$ subsamples are then used as training data. This process will be repeated k times so that each of the k subsamples are used exactly once as the validation data. The k results can be then averaged to produce a single estimation.

2.6. Introduction to Gradients

In vector calculus, the gradient of a scalar-valued differentiable function f of several variables is the vector field ∇f whose value at a point p is the vector whose components are the partial derivatives of f at p. That is, for $f\colon \mathbb{R}^n \to \mathbb{R}$, its gradient $\nabla f\colon \mathbb{R}^n \to \mathbb{R}^n$ is defined at the point $p = (x_1, \ldots, x_n)$ in n-dimensional space as the vector:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial x_1}(p) \\ \vdots \\ \frac{\partial f}{\partial x_n}(p) \end{bmatrix}.$$

Under gradient vector we understand "the direction and rate of fastest increase". If the gradient of a function is non-zero at a point p, the direction of the gradient is the direction in which the function increases most quickly from p, and the magnitude of the gradient is the rate of increase in that direction, the greatest absolute directional derivative.

3. Overview of Data

The datasets analyzed in this project come from our industry partner, NTT Data. There are three datasets:

1. The store dataset has 1115 entries and consists of:
   a) Store-ID (each store receives an unique ID in range between 1 and 1115)
   b) Store-Type (in this case a,b,c,d)
   c) Assortment (in this case a = basic, b = extra, c = extended)
   d) Information about the competition (distance from each store, month and year of their opening)
   e) Promo2 and other related variables describing ongoing promotions (week and year since they were introduced or interval of their repetition)
2. The train dataset has 1017209 entries and consists of :
   a) Store-ID
   b) Day of the week
   c) Date
   d) Number of Customers on this date
   e) Open (in this case 0 = store closed and 1 = store opened)
   f) State Holiday (in this case a = public holiday, b = Easter holiday, c = Christmas, 0 = None)
   g) School Holiday (in this case 1 = the store was closed, thus open = 0 because stores are not open during holidays and case 0 = the date of current entry is not a school's holiday)
   h) Promo (in this case 0 = the store does not have a promotion on this date and 1 = the store has a promotion on this date)
   i) Sales on this date, which is the target variable to predict in this project
3. The test dataset has 41088 entries and consists of the same variables as the train dataset, except Number of Customers and Sales (Sales is the target variable, so it must not be present in the test dataset). In addition to the train dataset, this dataset has one column called ID for each unique tuples consisting of Date and Store.

Before starting the analysis, one has to first review the quality of the datasets and deal with the problem of missing data.

A look into the train dataset proves that it does not have any missing information, as it can be seen later in Chapter 5. The test dataset has a few missing values for the variable Open, which may result in more missing values once this dataset is joined with the store dataset, which unfortunately has even more missing data.

The following chapters describe the preprocessing steps conducted on all the datasets in detail, in particular how the missing data problem was approached. After this is explained, the documentation will continue by presenting the steps required to successfully train a Machine Learning model, which will achieve the goal of this project: predict the sales of the given datapoints from the test dataset.

4. Implementation for SAC

4.1. Installation and Configuration

SAP Analytics Cloud is only possible to access with an Internet connection. There are two ways to login to SAC:

1. Purchase SAC and receive a link to a Unique URL. This option is ideal for enterprise accounts.
2. Use trial account to login through the website https://saphanajourney.com using a Chrome browser.

For this project in collaboration with NTT Data, the access authorization to SAC is given by NTT Data along with the three datasets: store.csv, train.csv and test.csv.

4.2. Preparing and Preprocessing

4.2.1. Import csv file

The first step to preprocess the data in SAC is to visually analyze the data and for this, SAC provides us the smart feature of Stories. Stories in SAC help to discover insights that are hidden within our data. To do that, first import the csv files into a Story as shown step by step in Figure 4.1. Each step is marked with a number from 1 to 6 in the figure.

4.2.2. Analyze Data

After importing the csv files into Story, create charts and analyze which attributes consist of what data and how much of the data is null. Figure 4.2 shows an instance of a chart from Stories of the attribute PromoInterval from store data and how to create it step by step. The resulting chart shows that 48.57% of the PromoInterval is NaN and contains three categories 'Jan,Apr,Jul,Oct', 'Feb,May,Aug,Nov' and 'Mar,Jun,Sept,Dec' with 30.35%, 11.76% and 9.31% respectively.

4.2.3. Transform Data

4.2.3.1. Transform Store Data

After analyzing each of the attributes from the three datasets, perform the necessary transformations in the Modeler to prepare the data required as input for the predictive model. The

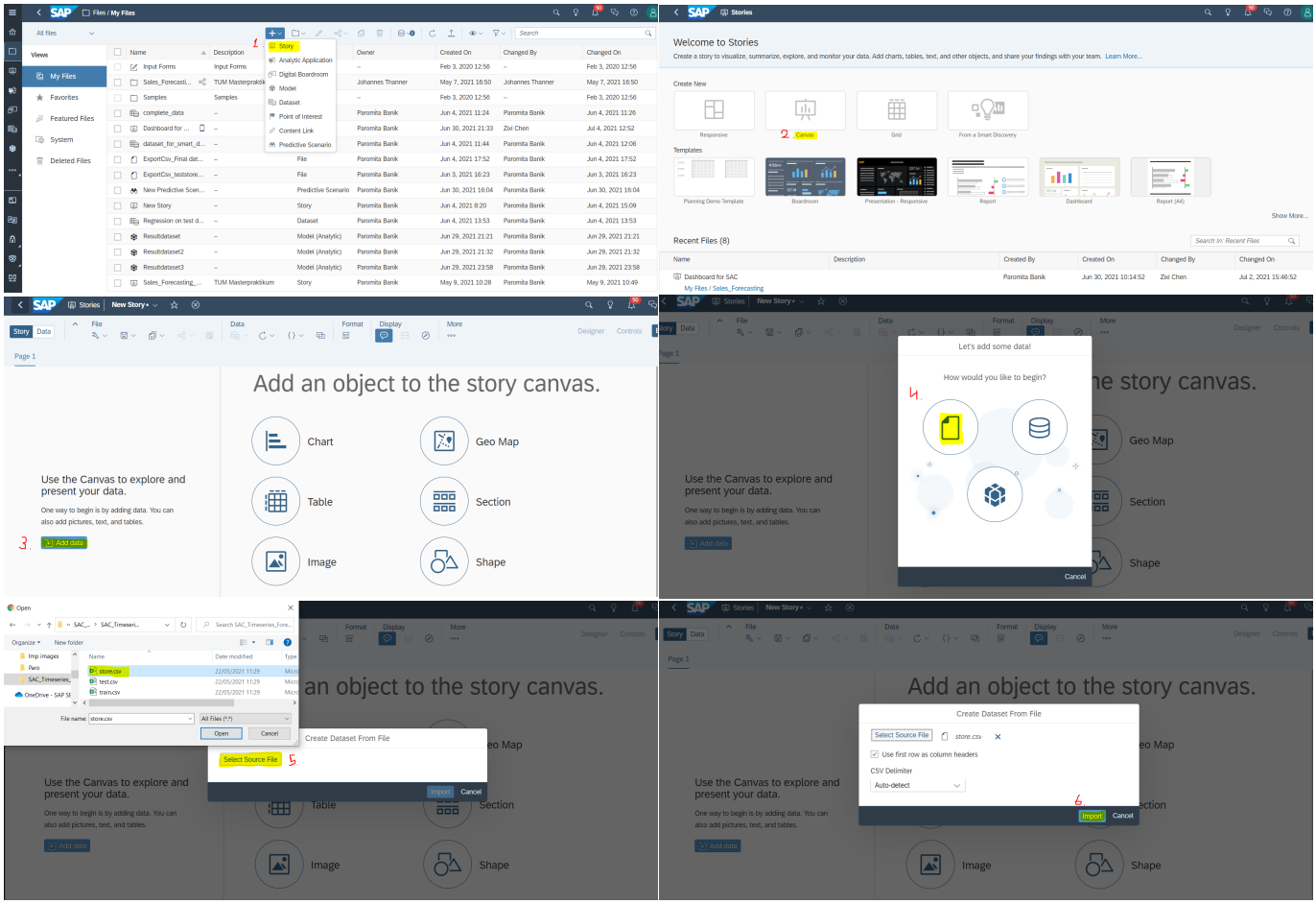


Figure 4.1.: How to import a csv file into a Story step by step

Modeler in SAC is an advanced modeling tool that allows to create, maintain, and load data into models and contains a variety of functions that offer a deeper dive into data modeling. Figure 4.3 shows how a typical transform function is done in a Modeler in step 5, where all the values matching 'Jan,Apr,Jul,Oct' in PromoInterval are replaced with value 1.

Similar to the step 5 depicted in Figure 4.3, perform the following transformations on the store dataset:

- If Promo2 is 0, replace the NaNs in Promo2SinceWeek, Promo2SinceYear and PromoInterval to 0.
- Replace the categorical data 'Jan,Apr,Jul,Oct', 'Feb,May,Aug,Nov' and 'Mar,Jun,Sept,Dec' in PromoInterval to numeric data 1,2,3.
- Replace the categorical data a,b,c,d and a,b,c in StoreType and Assortment to numeric data 0,1,2,3 and 0,1,2 respectively.

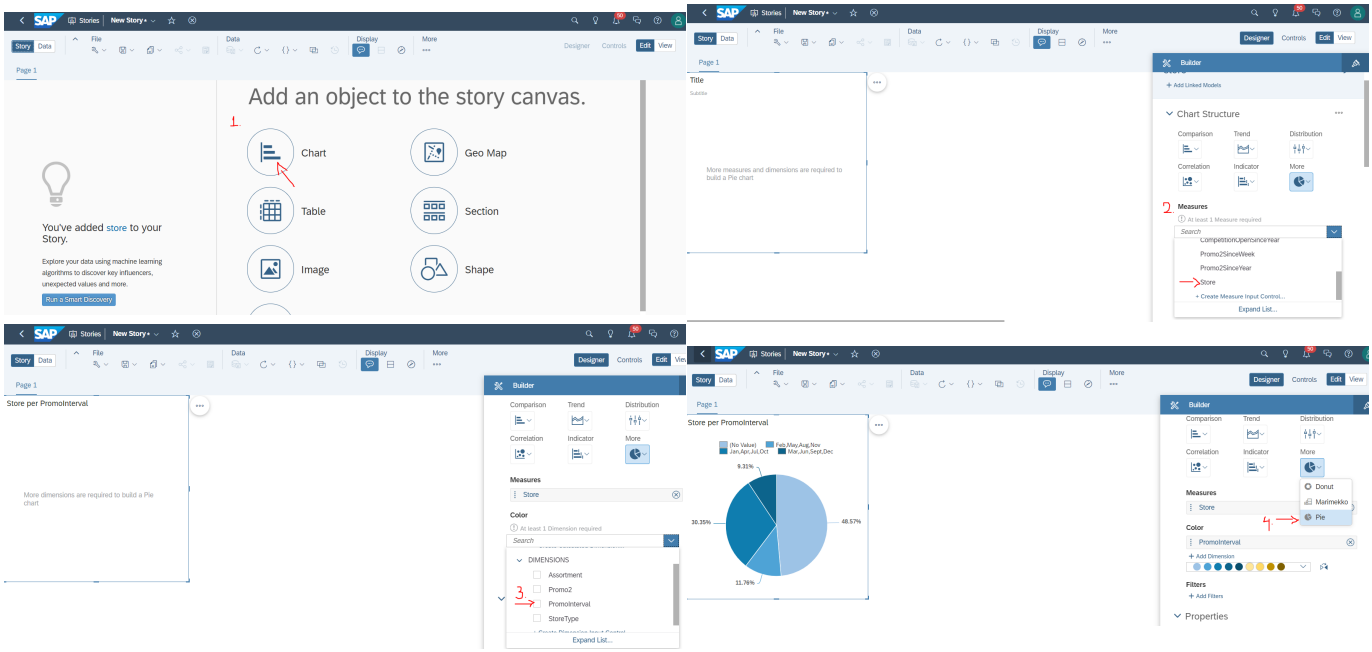


Figure 4.2.: How to create a chart in Story step by step

- Replace the NaNs in CompetitionOpenSinceMonth and CompetitionOpenSinceYear with the most frequent values in the respective columns.

4.2.3.2. Transform Train and Test Data

Perform the following transformation on the train and test datasets:

- Replace the categorical data 0,a,b,c in StateHoliday to numeric data 0,1,2,3.

Because SAC has the limitation of 1 million rows when combining datasets in Modeler, additionally perform the following transformations on the train dataset to delete rows and balance the data:

- Since the test data contains only StateHoliday with values 0 and 1, delete the rows with StateHoliday 2 and 3 in the train data and keep the ones with 0 and 1.
- Further delete the rows in train data for the 258 stores that are not in test data.

Figure 4.4 shows an instance of how to delete rows with specific values of a column in Modeler.

4.2.4. Merge data

Now that the data is within the limit of 1 million rows, merge the train and store data in Modeler by the Store column as shown in Figure 4.5 and create the model. Because a

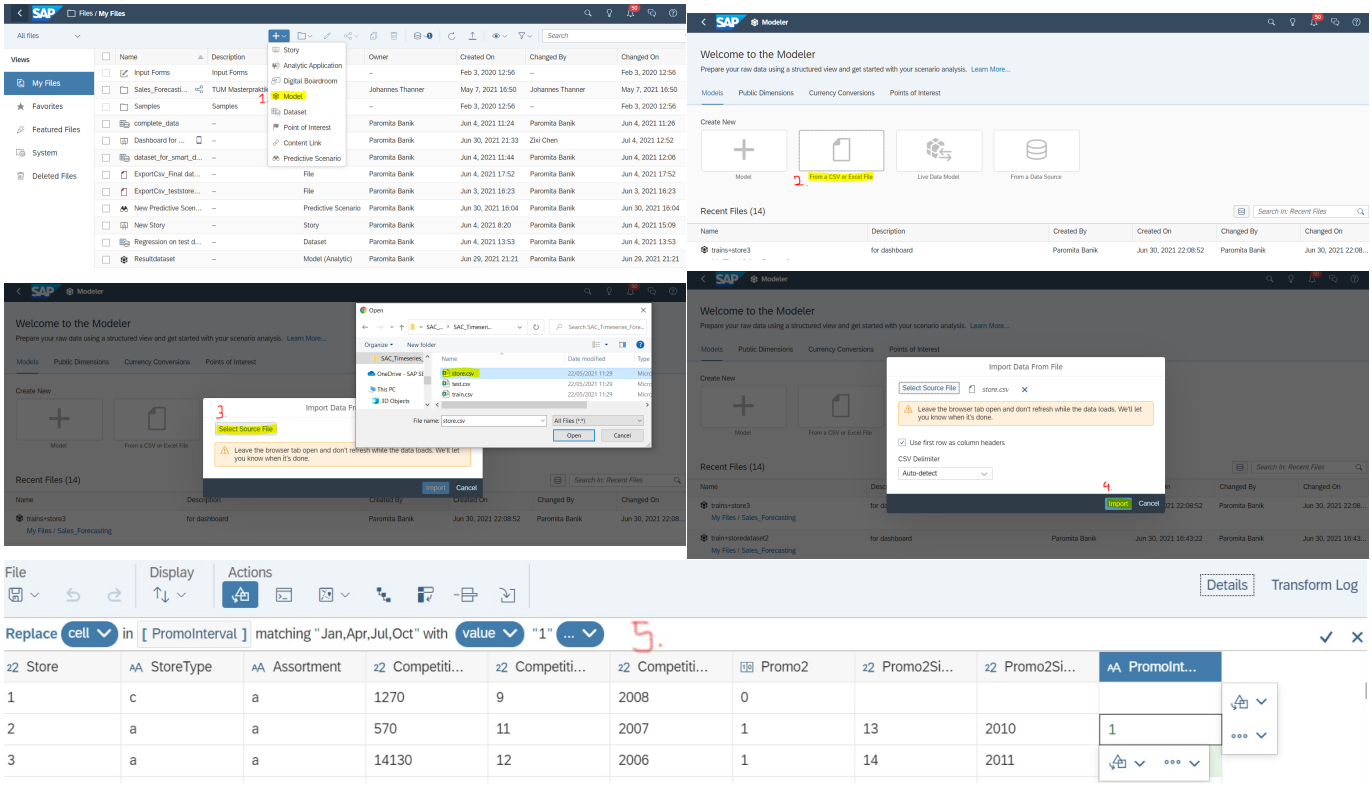


Figure 4.3.: How to import and transform data in Modeler

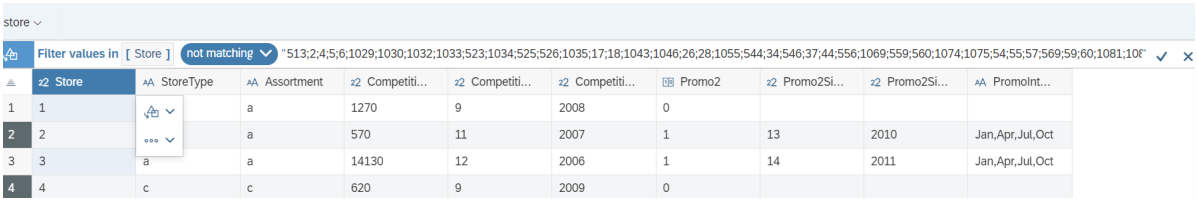


Figure 4.4.: How to delete rows in Modeler

predictive model in SAC can only be built with a Dataset and not a Model, export the Model as a csv file (shown as step 5 in Figure 4.5) and import it again into a Dataset (for example refer to Figure 4.1 or 4.3 on how to import csv files) to prepare the final input datasets for the predictive model. Now, repeat the same steps to merge the test and store data in order to get the same amount of features as the merged train and store data.

4.3. Feature Selection

The key influencers or features that impact sales is found via Smart Discovery in SAC. Smart Discovery is a powerful feature in SAC that runs automated machine learning algorithms in the backend to find out correlations between the dataset elements against the target metric.

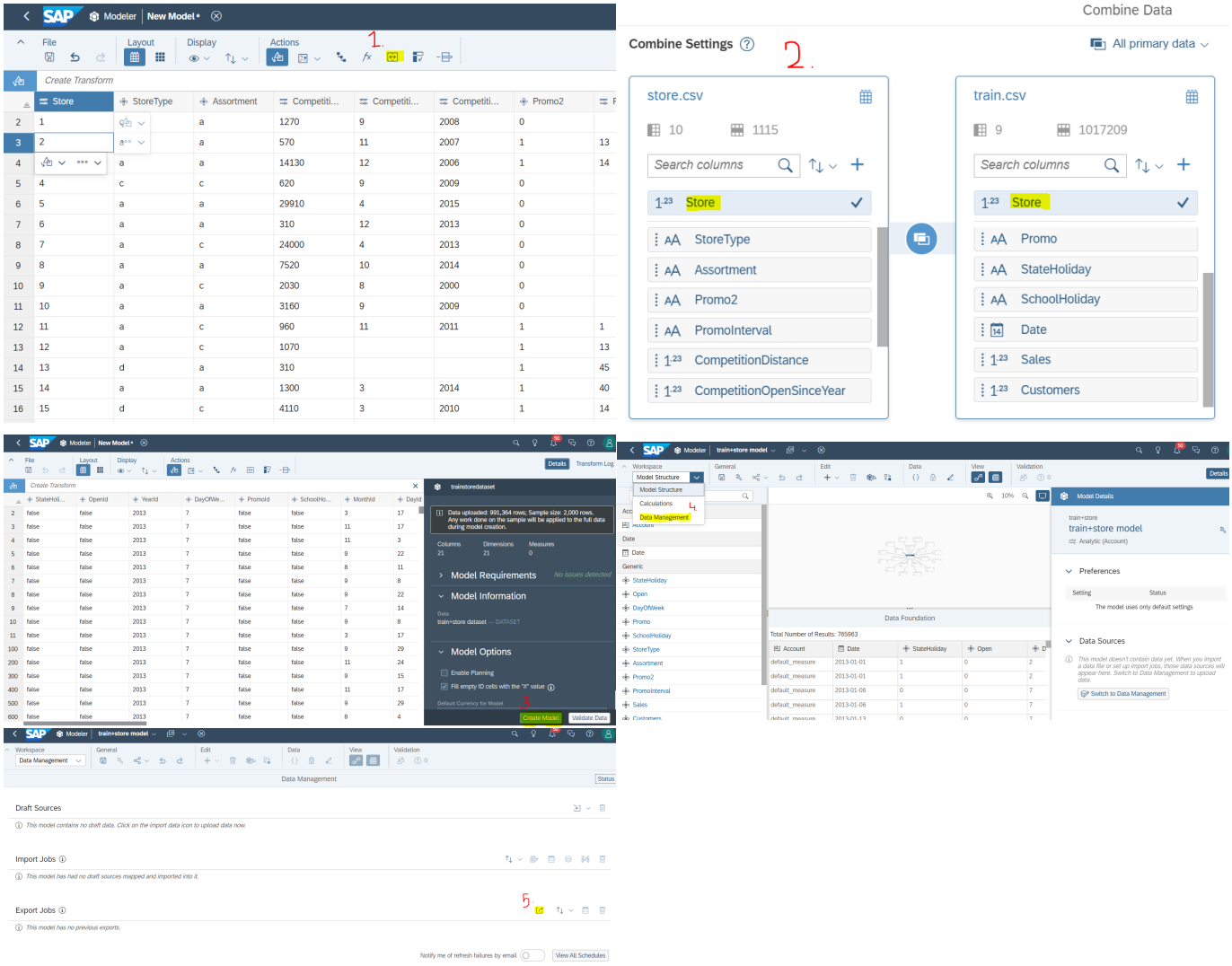


Figure 4.5.: How to combine data, create model and export model in Modeler

This is in this project equivalent to what is done as correlation analysis in SAP HANA. To launch a Smart Discovery, open a story containing the merged train and store dataset (refer to Figure 4.1 for how to open a story) and run the Smart Discovery feature as shown in Figure 4.6. Smart Discovery gives us the 10 key influencers of Sales, which as shown in Figure 4.6, are DayOfWeek,CompetitionOpenSinceMonth, CompetitionDistance, Promo, Promo2SinceYear, Promo2SinceWeek, StateHoliday,SchoolHoliday, StoreType and Assortment. The Smart Discovery also shows the columns that are highly correlated with Sales. In this case it is Customers and is automatically excluded by the Smart Discovery from the analysis because it brings the same information as sales and doesn't explain it.

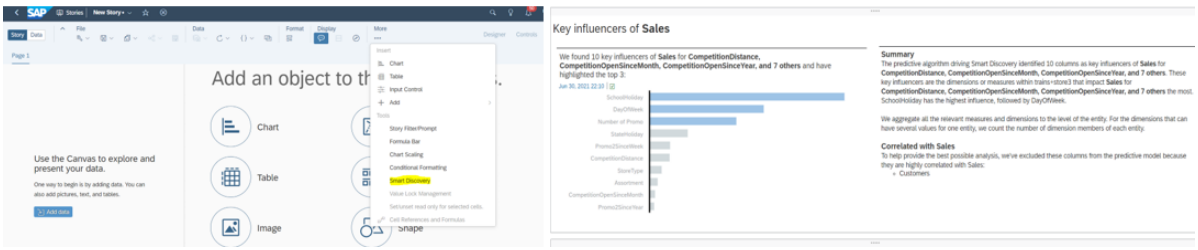


Figure 4.6.: Smart Discovery for Feature Selection

4.4. SAC Smart Predict: Predictive Scenario

4.4.1. Regression Model

Predictive Scenario in SAC Smart Predict generates a predictive model from historical data. Since our data is not usable for time-series forecast in SAC (refer to Section 6.1.1), the other suitable predictive model to try is regression as sales is a continuous quantity. In a regression, the notion of time is not necessary. Each row of the historical dataset represents an entity described by a set of variables. The value of the target of a new entity is predicted based on the values of the other variables. SAC Smart Predict generates a formula and applies that formula to calculate a value for the target measure.

As shown in Figure 4.7, you start with the creation of a predictive scenario. You choose regression and give a name to the predictive scenario. When you mention the merged train and store data as the training data source, during the training process, Smart Predict assigns randomly 75% of cases for the estimation dataset (dataset from which the predictive model is built) and the remaining 25% for the validation dataset (dataset used to calculate the performance of the model). The target is the numeric variable “Sales”. Under the “Influencers” section, exclude the variables that were not detected as key influencers in Smart Discovery. Lastly, train the model.

4.4.2. Train Outcome

Figure 4.8 shows the outcome from the training of data by regression predictive scenario. The quality of the regression model is measured by the Root Mean Square Error (RMSE). It is a statistical indicator which measures the average of the square difference between values predicted by the predictive model and actual values of the target for all cases of the validation dataset. The smaller this difference is, the better the quality of the predictive model. The mean of the target is around 5,900 for both the estimation and validation dataset and standard deviation is around 3,700. We can say that the quality of the predictive model is very good by scoring more than 99% accuracy. The predicted curve (green) and the actual curve (red) match closely, therefore the predictive model is good and can be trusted to predict the value of the unknown target.

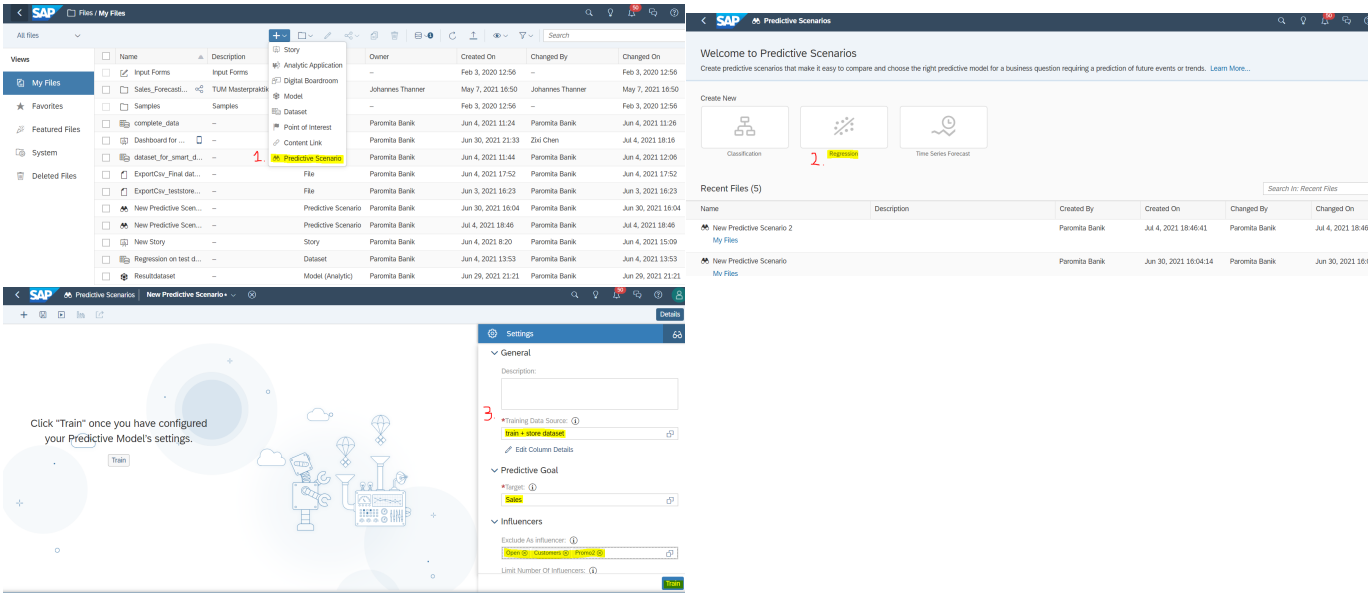


Figure 4.7.: How to create regression predictive model step by step

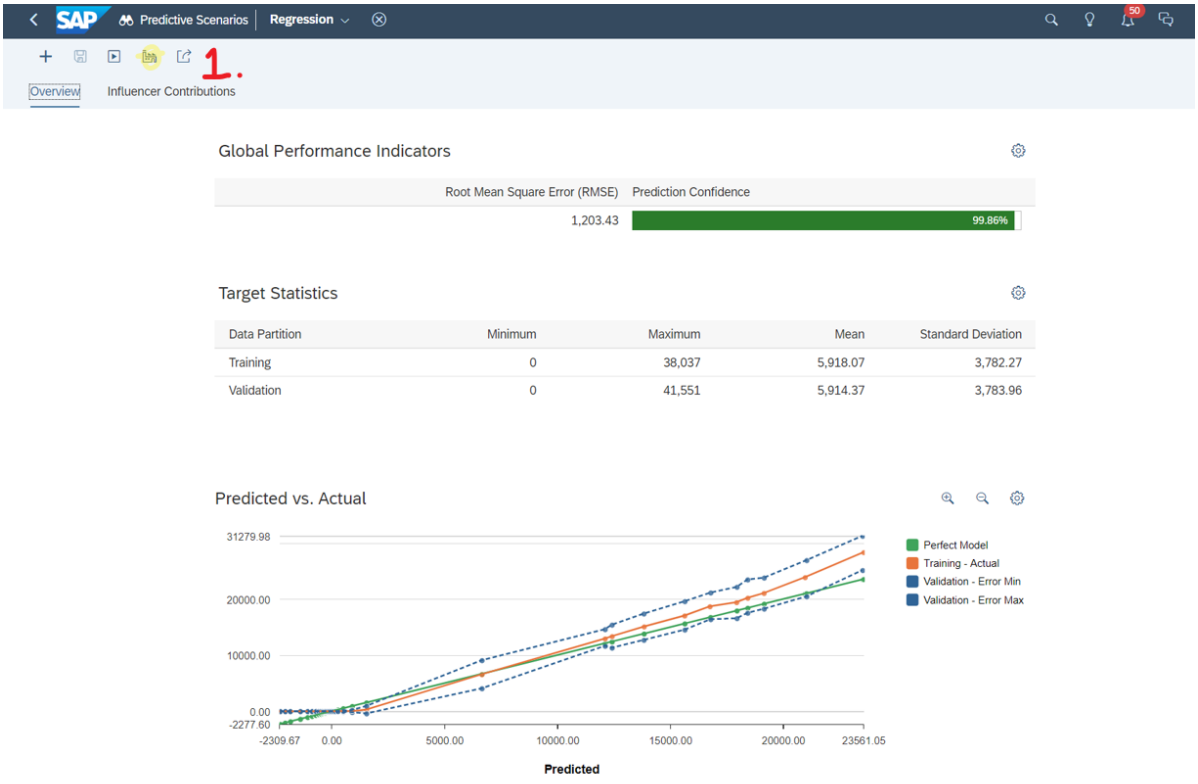


Figure 4.8.: Outcome after training model

4.4.3. Regression Model on Test Data

The primary objective of a predictive model is to get predictions when the value of the target variable is unknown. Therefore, apply the regression model on the merged test and store data by clicking on the "Apply Predictive Model" button (refer to Figure 4.8 marked yellow in step 1) and apply the settings as shown in Figure 4.9. The merged test and store dataset is a dataset with the same information as the training dataset but where the values of the target variable (in this situation the Sales) is unknown, which is what you want to determine. The replicated columns are the variables of the input dataset you want to retrieve in the output dataset. The statistical column is the column of predicted values of the target variable and the output is the name you give to the output dataset.

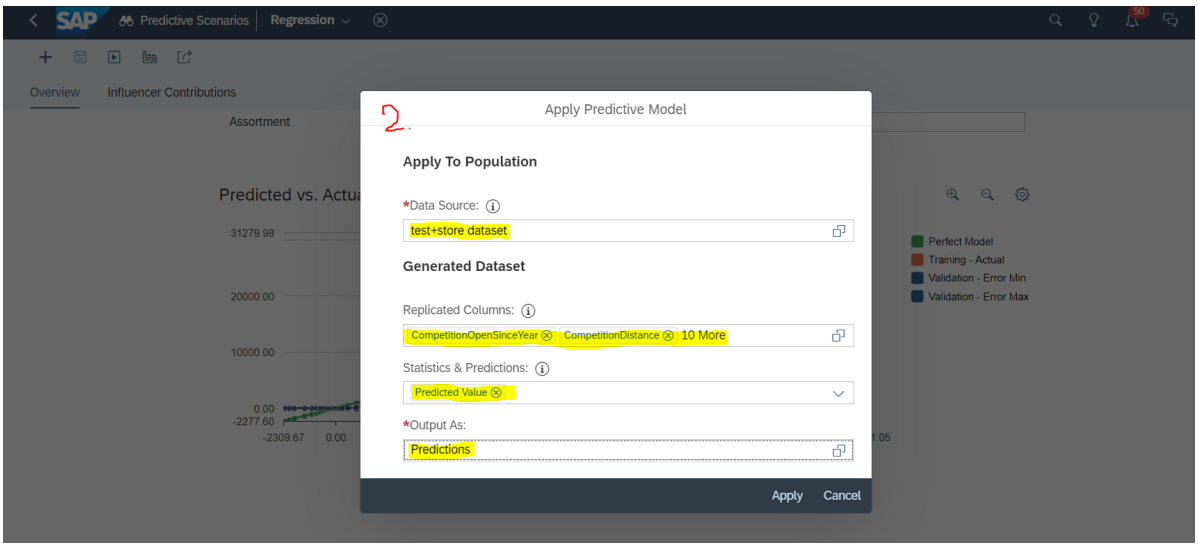
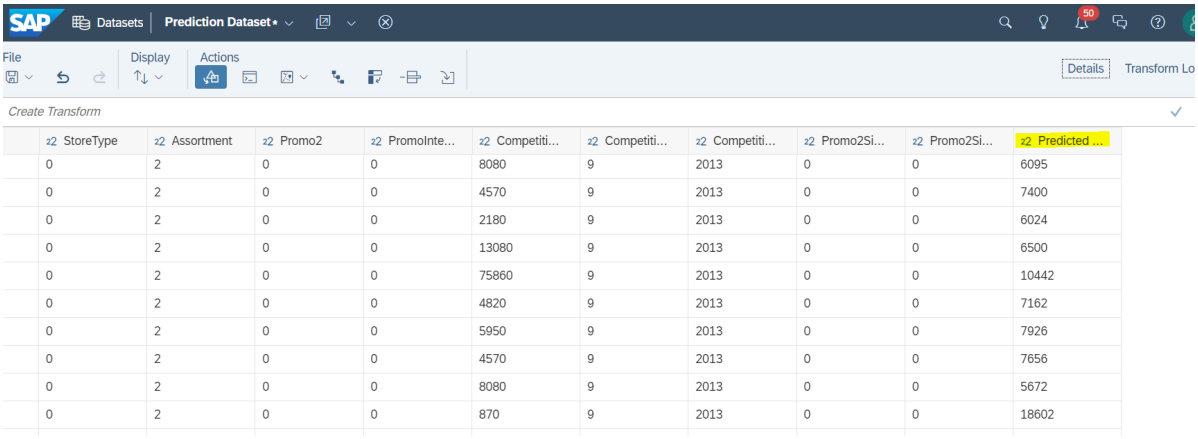


Figure 4.9.: Applying regression model on test data

Figure 4.10 shows a screenshot of the output dataset of the predicted sales which can then be used to visualize the predictions.

4.5. Visualization for SAC

According to our planned solution, we are going to use the feature of Dashboard in SAC to display the result of the sales forecast along with powerful visualizations of the data. In SAC, a dashboard is the same as a Story. To create a SAC Story Dashboard, import the resulted output dataset from section 4.4.3 to a Story (refer to Figure 4.1) and create line charts to show the predicted sales per date (refer to Figure 4.2 for example). Figure 4.11 shows an instance of the line chart. Then run Smart Discovery in the same story (refer to Figure 4.6). The Smart Discovery will give five pages as output: Overview page summarizes the results for the target variable, Key Influencers page shows the variables that influence the target variable, Unexpected Values page shows the information about outliers and Simulation page



StoreType	Assortment	Promo2	PromoInte...	Competiti...	Competiti...	Competiti...	Promo2Si...	Promo2Si...	Predicted ...
0	2	0	0	8080	9	2013	0	0	6095
0	2	0	0	4570	9	2013	0	0	7400
0	2	0	0	2180	9	2013	0	0	6024
0	2	0	0	13080	9	2013	0	0	6500
0	2	0	0	75860	9	2013	0	0	10442
0	2	0	0	4820	9	2013	0	0	7162
0	2	0	0	5950	9	2013	0	0	7926
0	2	0	0	4570	9	2013	0	0	7656
0	2	0	0	8080	9	2013	0	0	5672
0	2	0	0	870	9	2013	0	0	18602

Figure 4.10.: Output dataset with predictions from regression model

allows to test hypothetical scenarios with the data.

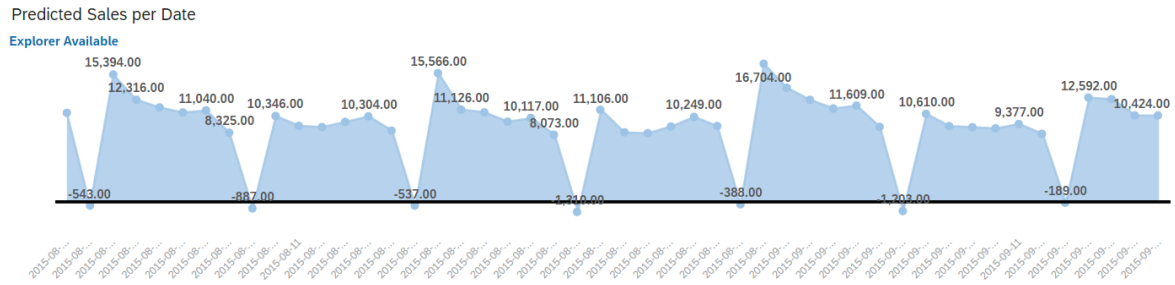


Figure 4.11.: Predicted sales per date from the result of regression model in SAC

5. Implementation for SAP HANA

5.1. Installation and Configuration

SAP HANA XSA is only accessible with an Internet connection. The tools required for this project are WebIDE, Database Explorer and XS Advanced Cockpit. If you want to learn about SAP HANA XSA, you can download it from the SAP products website, where you can find SAP HANA, express Edition as a trial [1]. This version is available for free for development and productive use. The downloaded file is of .exe format and represents a download manager, where you can choose which packages you need. For this project, we chose SAP HANA XSA, server + applications, because we need to have access to WebIDE and Database Explorer. For this package, the requirement is to have at least 16 GB RAM available. The download manager includes also an installation guide, which you have to follow in order to get the HANA environment running. Furthermore, the download manager will issue an image, which will contain SAP HANA and has to be imported by a virtual machine (VM), where you can finally start testing its functionalities. The installation of the SAP HANA, express edition image requires at least 68 GB disk space.

For this project, we decided to use the virtual machine provided by VMWare Workstation Pro, which can be downloaded from their own website and is available for free as a trial for 30 days [2].

For the users who do not meet these requirements, there is an option to use SAP HANA Cloud or receive access credentials from someone with an admin role.

The SAP HANA solution of this project includes a collaboration with a Python script, which is written in a Jupyter Notebook. In order to be able to run this script, you have to install all the necessary packages listed in the requirements file.

5.1.1. Start Using SAP HANA 2.0, express edition

5.1.1.1. Setup the Environment

After the downloads required in Section 5.1 are finished, we can start with the installation as described in the issued guide. With the command $\backslash sbin \backslash ifconfig$, you can see the IP address of your VM. This IP address has to be saved in the *etc/hosts* file, otherwise the information provided by the VM will not be accessible on the outside, meaning you will not reach WebIDE in the browser. If after this change you still have no access to the links of the xs apps (listed in terminal of the VM using the command 'xs apps'), even though the instances are running 1/1,

[1] https://www.sap.com/products/hana/express-trial.html
[2] https://www.vmware.com/products/workstation-pro/workstation-pro-evaluation.html

you have to check if the network connection is set right. The advantage of using VMWare Workstation Pro is that we are allowed to change these settings by going to the Edit tab of the VM and clicking on Virtual Connection Editor like shown in Figure 5.1. Next, go to settings and change the connection to bridged like presented in Figure 5.2.

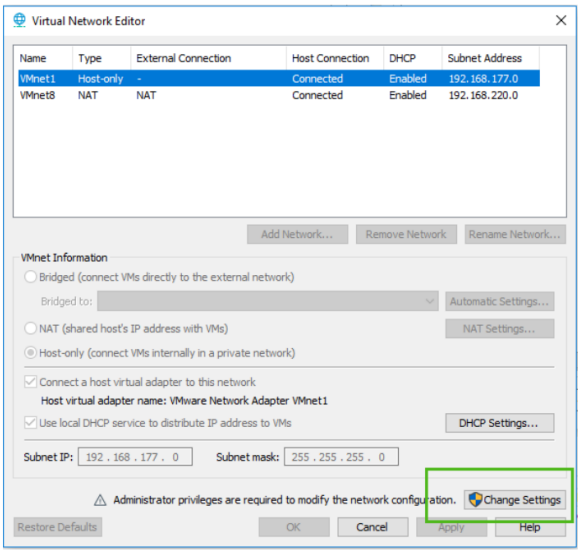


Figure 5.1.: Check the connection type

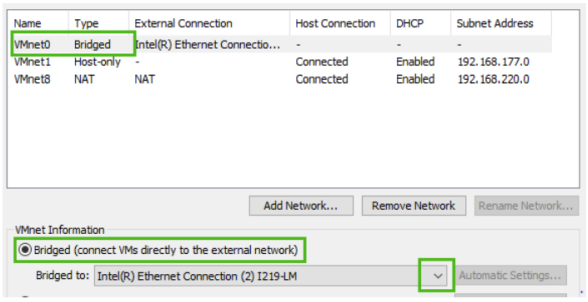


Figure 5.2.: Change the connection to Bridged

If there are still problems reaching the links listed by the xs apps, possible causes are:

1. Some VPN is active on your laptop and blocks the communication between the client and VM.
2. You did not set in the settings of the VM the right amount of RAM. It works best if the VM is given 32 GB RAM.
3. There is not enough disk space.
4. The IP address of the VM changed. Check this in the VM cli and make the update in the hosts file

5.1.1.2. Upload the Data required for the Data Analytics Project

If you have finally access to WebIDE, in our case it can be reached at $https://hxehost:53075/$, there are two possible scenarios:

1. You can not sign in as described in the installation guide: using the master-password set during the setup and the username XSA_DEV. This happens if the XSA_DEV role is not activated for development yet. In order to solve this, you will have to go on XSA-Cockpit (in our case at $https://hxehost:51036/$), by logging in with the XSA_ADMIN username and the master-password and activate the development for the XSA_DEV role.

2. You can sign in and see an empty workspace, ready to be used and also have access to the Database Explorer.

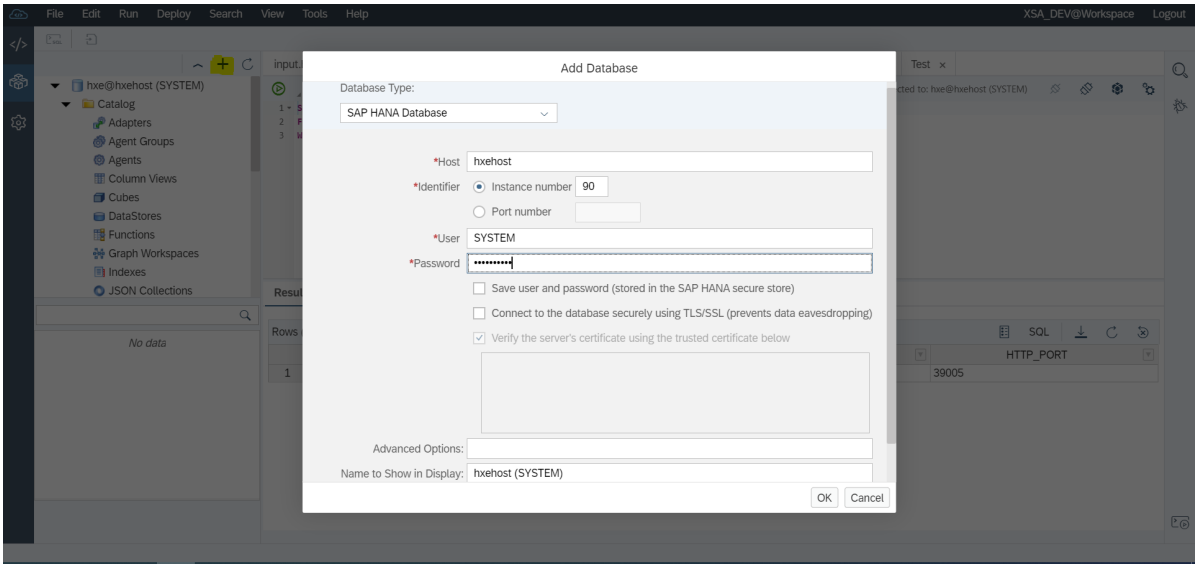


Figure 5.3.: Add Database

The next step is to add a database in the Database Explorer and to create a schema, which will include all your tables. This is done by clicking on the plus sign as shown in Figure 5.3. In our case, the host is *hxehost*, the instance number is 90 and we login as *SYSTEM* and with the master-password [3]. In the SQL Editor presented in Figure 5.5, you can write the commands to create a schema with tables, where you can also specify the datatypes of their columns as exemplified in code listings at A.1.

Now that you have a schema with empty tables and defined columns datatypes, you can import the data from the .csv files provided by our industry partner NTT Data. Under catalog, click on tables and you will see them as a list like shown in the bottom-left corner of the Figure 5.5. Right click on each one of them and click on Import Data. This opens an import assistant such as displayed in Figure 5.4, where you can upload directly the files from your own laptop. The next step maps the columns of the uploaded file with the corresponding columns of the table in HANA, where this new data will be uploaded. The last step checks if there are any errors in this process.

In our case, the columns had to be set to the datatypes written in the schema, because otherwise the information of the files was not processed without any errors. One issue for example are entries were the columns StateHoliday and SchoolHoliday are both "0". In this case, SAP HANA had a few problems processing these entries. We solved this issue by transforming the .csv file to split the text at each comma into columns by using Microsoft Office Excel. The new version of the .csv files was uploaded in SAP HANA.

[3]https://developers.sap.com/tutorials/xsa-explore-basics.html7d44b439-524d-4c53-99a5-c36cf226ff5c

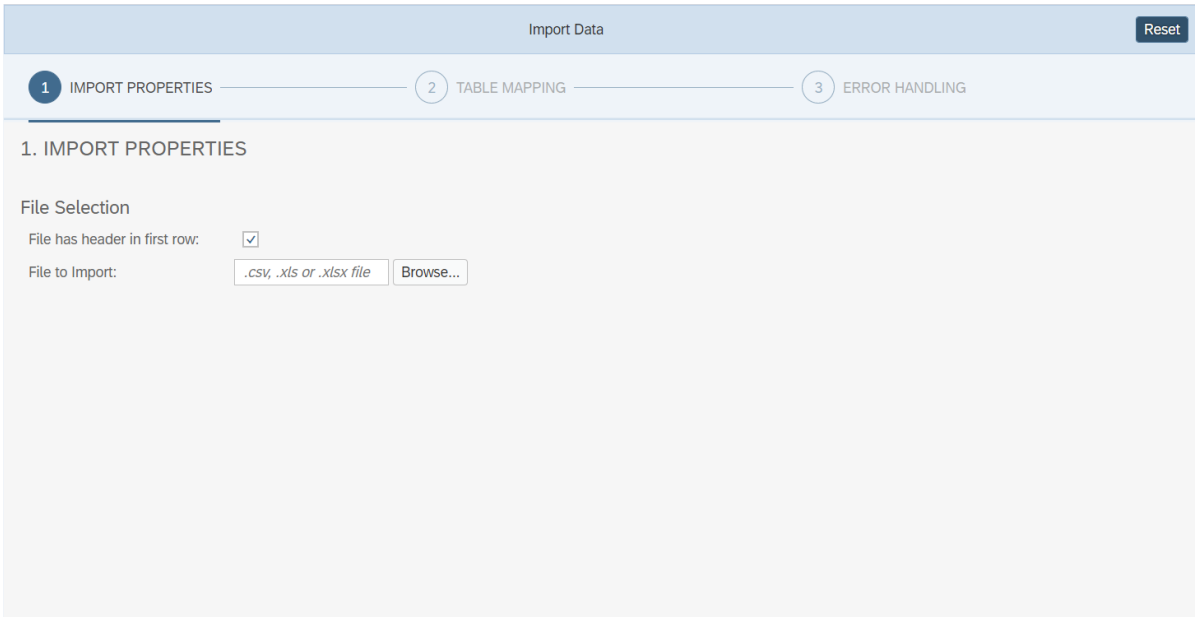


Figure 5.4.: Import Data to Tables

5.1.1.3. Create User for Python Script

Now that you have a schema and the tables containing all the data necessary from the .csv files, you have to create a user and grant it permissions to access the information saved in this schema. These credentials are later necessary to make it possible to access the data from the Jupyter Notebook.

By running the commands written in code listings at A.2, you can create a role with username ML_DEV and give it all the privileges to access your schema, which in our case is called input.

The last thing the Data Analytics script needs, is to know on which port to request the data. You can discover this by running the command displayed in code listing A.3 (in our case it is 39015).

Finally you can write all this information into the Python script and see if the connection was successful. If you receive a result similar to the one shown in Figure 5.6, then the communication between both platforms was established the right way and you can go on reviewing the implementation of the sales forecast task.

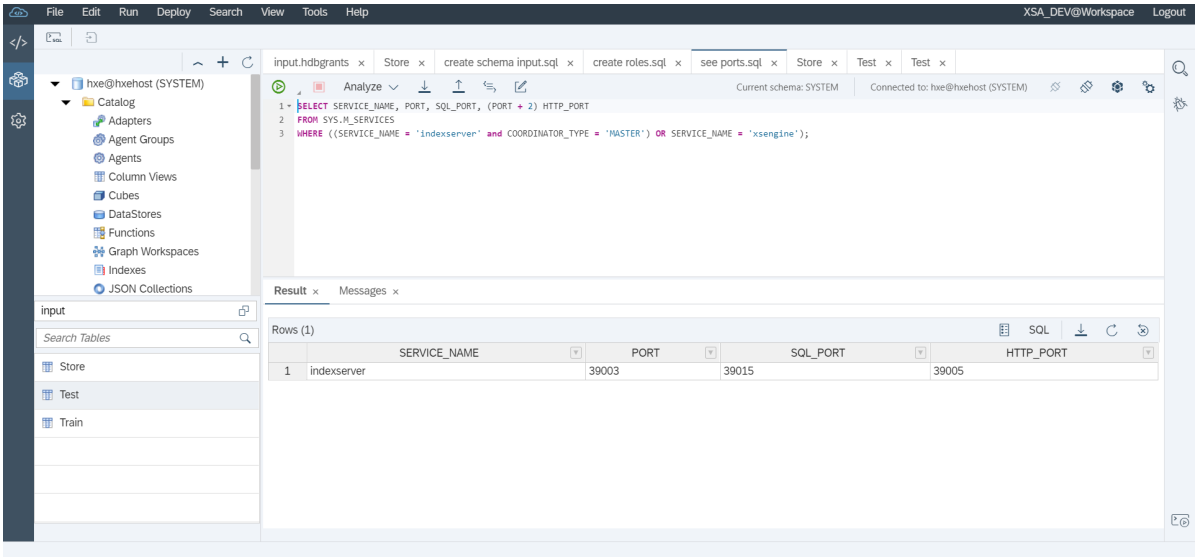
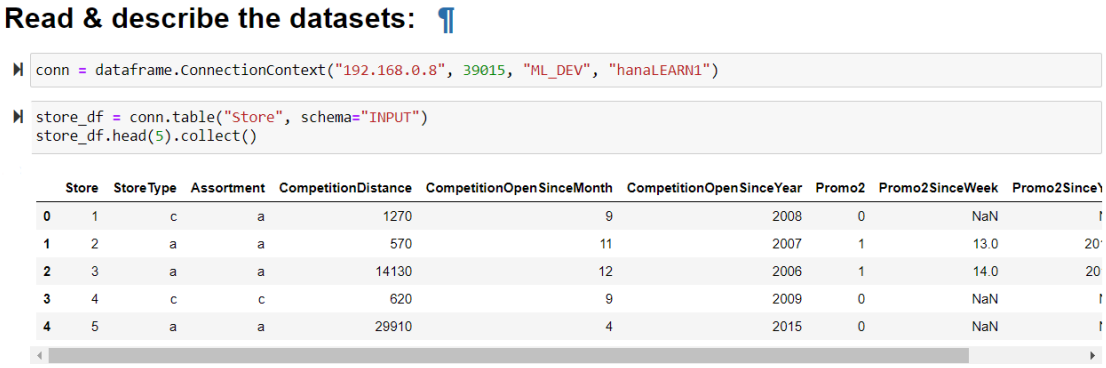


Figure 5.5.: See the information about the ports required by the Python script to connect to SAP HANA



Read & describe the datasets: ¶

```
conn = dataframe.ConnectionContext("192.168.0.8", 39015, "ML_DEV", "hanaLEARN1")
```

```
store_df = conn.table("Store", schema="INPUT")
store_df.head(5).collect()
```

	Store	StoreType	Assortment	CompetitionDistance	CompetitionOpenSinceMonth	CompetitionOpenSinceYear	Promo2	Promo2SinceWeek	Promo2Since
0	1	c	a	1270	9	2008	0	NaN	
1	2	a	a	570	11	2007	1	13.0	20
2	3	a	a	14130	12	2006	1	14.0	20
3	4	c	c	620	9	2009	0	NaN	
4	5	a	a	29910	4	2015	0	NaN	

Figure 5.6.: Example of a successful connection

5.2. Backend: Python Data Analysis Module

5.2.1. Preparing and Preprocessing

The steps for preprocessing the data in Python were done in the same manner as presented in Section 4.2.

The percentage of missing data per column was computed using the method listed at B.2 in Code Listings. This way, we realized that the train dataset had no missing entries, while some variables of the store and test dataset had missing data. This insights can be reviewed in the generated tables in Figures 5.7, 5.8 and 5.9.

Similar to the SAC solution described in Section 4.2, the Nans in the variables Promo2SinceYear, Promo2SinceWeek and PromoInterval were replaced by 0 because they were triggered

	Missing Values	% of Total Values
Store	0	0.000000
StoreType	0	0.000000
Assortment	0	0.000000
CompetitionDistance	3	0.269058
CompetitionOpenSinceMonth	354	31.748879
CompetitionOpenSinceYear	354	31.748879
Promo2	0	0.000000
Promo2SinceWeek	544	48.789238
Promo2SinceYear	544	48.789238
PromoInterval	544	48.789238

Figure 5.7.: Missing values in Store dataset

	Missing Values	% of Total Values
Store	0	0.0
DayOfWeek	0	0.0
Date	0	0.0
Sales	0	0.0
Customers	0	0.0
Open	0	0.0
Promo	0	0.0
StateHoliday	0	0.0
SchoolHoliday	0	0.0
Month	0	0.0
Day	0	0.0
Year	0	0.0

Figure 5.8.: Missing values in Train dataset

	Missing Values	% of Total Values
Id	0	0.000000
Store	0	0.000000
DayOfWeek	0	0.000000
Date	0	0.000000
Open	11	0.026772
Promo	0	0.000000
StateHoliday	0	0.000000
SchoolHoliday	0	0.000000

Figure 5.9.: Missing values in Test dataset

by the value 0 of Promo2. The reason for this is that Promo2 has two values: 1 means there is a promotion, 0 means there is no promotion. If there is no promotion, then there is also no promotion interval and no week, where the promotion is available, thus the missing values. So replacing the Nans with 0 makes sense in this case, because we are not replacing lost unknown data. The next missing values come from the CompetitionOpenSince and CompetitionOpenDistance variables and in this case, because we could not replace by a simple value like we did for the Promo-variables, the recommended solution is to replace each with their most frequent value because these variables are categorical. The methods doing these preprocessing steps are represented at B.3 in Code Listings.

We also created the columns Day, Month and Year by extracting their values form the Date column. This step was necessary, because otherwise this information would go lost during the analysis, since only numerical values can be processed and Date as it is in the original dataset now, can only be read as string or object datatype. This step was not necessary in SAC because SAC can read dates as date type.

5.2.2. Correlation Analysis

After learning about the Correlation Analysis in Section 2.2, we can now have look at its implementation which is shown in Code Listings at B.5. We used the function .corr() provided by the Pandas library to compute the correlation matrix. This method computes the pairwise correlation of the columns in a dataframe, which means, we can receive the computed correlation coefficient for each pair of the variables from our dataset. The default value of this method is $method =' pearson'$, but it can also compute other coefficients such as kendall or spearman. For this project, we used the default value.

The computed correlation matrix can be visualized with the help of a heatmap, where the

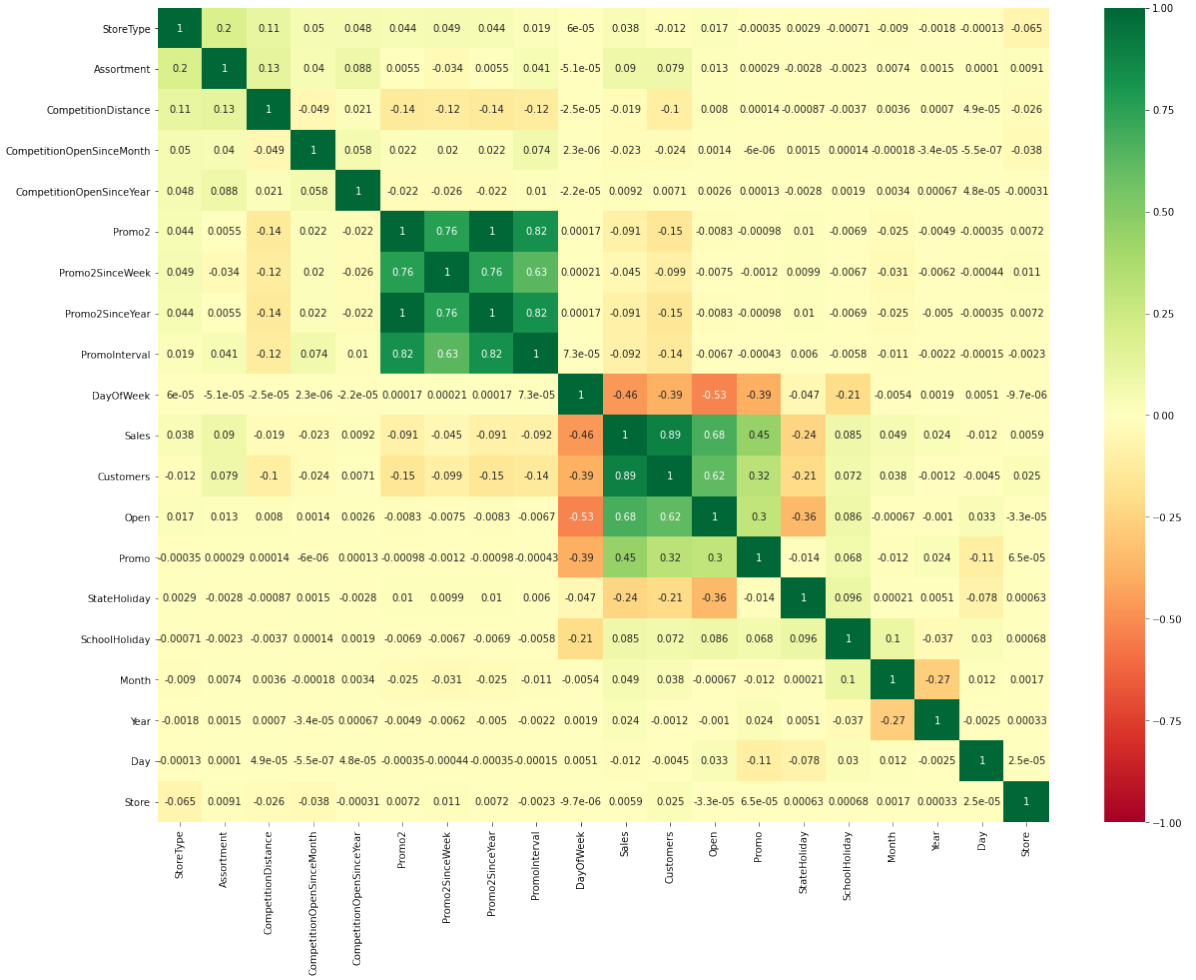


Figure 5.10.: Result of the Correlation Analysis in Python. Will be represented in SAC with provided heatmap-diagram

intensity of the colors represent the strength of the relationship between the variables on the x-axis and y-axis. Figure 5.10 was generated with the help of the Seaborn library, which is a statistical visualization Python library. As we can see in this figure, the variables Promo2, Promo2SinceWeek, Promo2SinceYear and PromoInterval are highly correlated to each other, thus using them all for training the model would not be meaningful since they are redundant and they would not provide more information than just one of these variables. Hence, we decided to select only Promo2 for the training.

The next insight we get from this heatmap is that the inputs Open and Customers are highly correlated to the Sales variable, which in our case is the output. This means, we have to exclude these two variables from the training of the model, otherwise the model would copy their behavior and not actually learn from the data. We decided to keep the inputs DayOfWeek and Promo for training because their coefficient is lower than 0.5, which means

they may only have a moderated correlation to the output.

In conclusion, the inputs used for the training of the model in the following sections are: Store, StoreType, Assortment, CompetitionDistance, CompetitionOpenSinceMonth, CompetitionOpenSinceYear, Promo2, DayOfWeek, Promo, StateHoliday, SchoolHoliday, Day, Month and Year.

5.2.3. Regression Analysis

This section will discuss the implementation of the training of the prediction model required for this project. Since we want to test the same functionalities and methods for both SAC and SAP HANA, we will also implement in Python Regression Analysis instead of time-series forecast as it was already mentioned in Section 4.4.1.

The Python module contains two regression models: Random Forest Regressor and XGBoost. Section 5.2.4 will describe which model will be deployed for the prediction of the unseen data from the test dataset.

The complete dataset was split in 70% for the training and 30% for the validation of the model.

5.2.3.1. Random Forest Regressor

After learning about Random Forests in Section 2.4.2, we can have a look at its implementation which was possible thanks to the sklearn.ensamble library. The code for this section can be found at B.6 in Code Listings. As we can see, the data was first split for training and validation like explained in the previous Section 5.2.3. Afterwards, we used 100 decision trees to make the predictions. The next steps are to fit the model and make predictions on the data contained in the validation set.

We do not need to implement cross validation here in order to get an unbiased estimate, because as we learned in the introduction section, Random forest already uses bagging to train the model. This means, the issue is solved internally because each tree is trained on a different bootstrap sample from the original data, therefore we do not need cross validation to shuffle the data into random splits anymore.

The accuracy of the predictions was computed using the the code listed at B.8 in Code Listings and represents the mean absolute percentage error. This model has an accuracy of 98.5% and the results of the prediction can be visualized in SAC.

5.2.3.2. XGBoost

Extreme Gradient Boosting (XGBoost) is a flexible and efficient library, which implements machine learning algorithms using the Gradients Boosting framework [4]. XGBoost is a supervised learning method, which uses gradient boosted trees. This means, both Random Forest and XGBoost have models based on decision tree ensembles, but XGBoost trains the

[4] https://xgboost.readthedocs.io/en/latest/

trees differently. In this case, XGBoost does not train all trees at once but adds them one by one, by choosing each time the optimal tree to bring us in the most efficient way to our goal [5].

For training this model, we used 1000 gradient boosted trees (which represents the number of boosting rounds) and a learning rate of 0.05, as it can be seen in the code listing at B.7. We let the model be fitted on the data selected for the training, with the option to stop early if 100 consecutive rounds do not improve the model and using the validation data for its evaluation.

Afterwards, we needed to add k-fold cross validation, because otherwise this method would only have one chance to train the data on this dataset. Therefore, we used k-fold and divided the dataset in 10 splits, so that the model can train on 10 different sets and thus minimize bias. The computed k-fold cross validation average score is 0.92. The accuracy of the predictions made by XGBoost was computed using the code listed at B.8 in Code Listings and represents the mean absolute percentage error. This model has an accuracy of 96.54% and the results of the prediction can be visualized in SAC.

5.2.4. Regression Model on Test Data

This is the final step of the implementation of the Python module: prediction of the sales of the datapoints from the test dataset, which has not been used in the analysis until now. For this step, we need to make the same preparation on the dataset described in the previous sections, in order to get the same amount of features. If the dataframe created for the testdata does not provide all of them, we can not predict using the saved models, because they require the same number of input variables. Therefore, we join the test dataset with the store data, the same way we did with the train dataset.

Afterwards, we continued with the preprocessing steps like using the encode_factor method on the variable StateHoliday and creating the columns Day, Month and Year from the Date column, as we did for the train dataset.

We already mentioned in Section 3 that we have missing data in the test dataset. Now, if we want to use the model created by the Random Forest Regressor, it will not compile successfully because of the present Nans. This means, that unless we solve the issue by filling these empty entries, it will not be possible to predict using this model.

This is the reason why we decided to use XGBoost for the prediction of the test dataset, because it works also without any preprocessing of missing data. This is possible because when XGBoost finds a missing feature on a decision node, it can take a default direction of the branch and continue the path without interruption. [6]. The final results can now be seen in SAC.

[5] https://xgboost.readthedocs.io/en/latest/tutorials/model.html
[6] https://towardsdatascience.com/xgboost-is-not-black-magic-56ca013144b4

6. Limitations

6.1. For SAC

6.1.1. Time-Series Forecast

Time Series Forecast is a predictive scenario in SAC which is useful for estimating future values of a measure that has a time dimension, which is also why it fit best to our use case of sales prediction. The algorithms in time series forecast analyze the curve composed from the values of the measure taken at different time and break down this signal in elementary components: trend, cycles and fluctuation. Then it combines these components to predict what will be the future values of the measure.

Unlike the regression predictive scenario in SAC, where the trained predictive model was possible to apply on the test data, the time series forecast in SAC required the train and test data to be merged which was not within the scope of SAC. As a result, the rows of test data needed to be appended to the train data separately in an external tool such as Excel or Python (e.g. how to append in Python: result = train.append(test)).

Unfortunately even after that, it was not possible to train the time series forecast on our given dataset because it required strictly increasing date values with no duplicates and our data contained non-increasing date values. Figure 6.1 shows a screenshot of the error thrown by time series forecast in SAC.

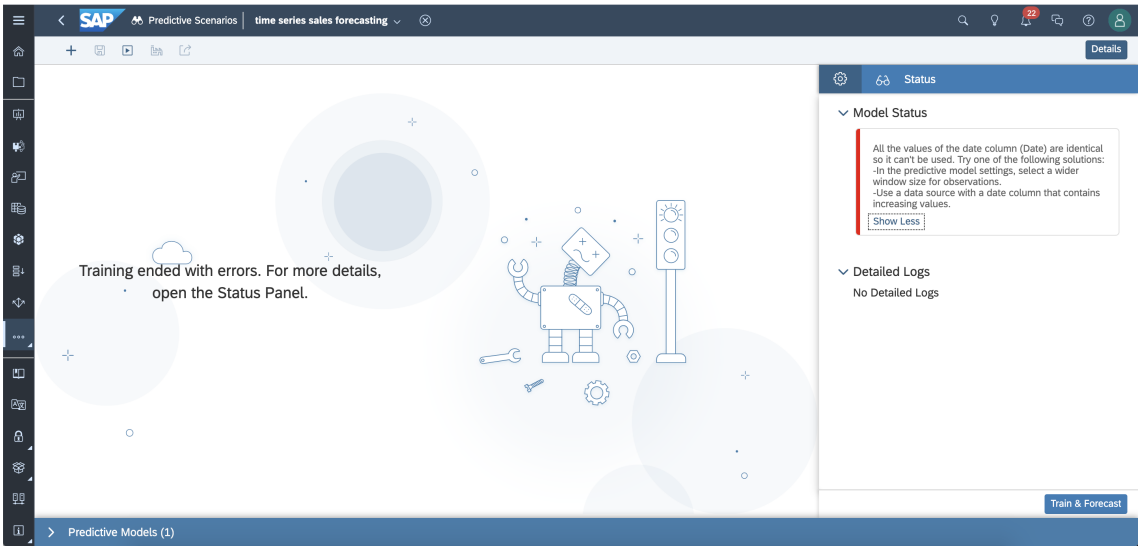


Figure 6.1.: Error: Dates are identical

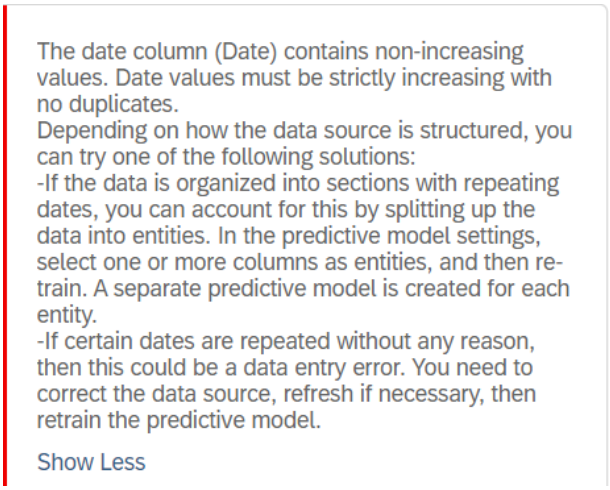


Figure 6.2.: Error: Non-unique Dates

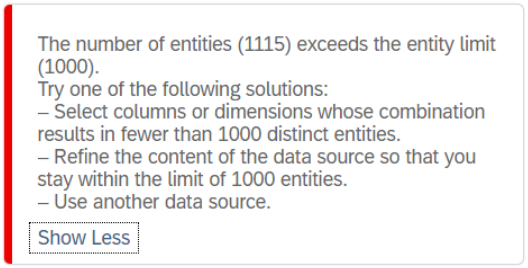


Figure 6.3.: Error: Exceeded limit of 1000 entities

Following the suggestion provided in Figure 6.2, we split up the data by the entity "store" which meant that a separate predictive model would be created for each store. This lead us to the error (refer to Figure 6.3) that number of stores cannot exceed 1000 whereas our data contained 1115 stores.

After deleting the stores in train data that were not in test data and limiting the number of stores to 1000, we attempted to run the time series forecast again. As an example, the results for Store 1 can be seen in Figure 6.4, where a warning suggests to increase the training data source, because there are not enough confidence intervals.

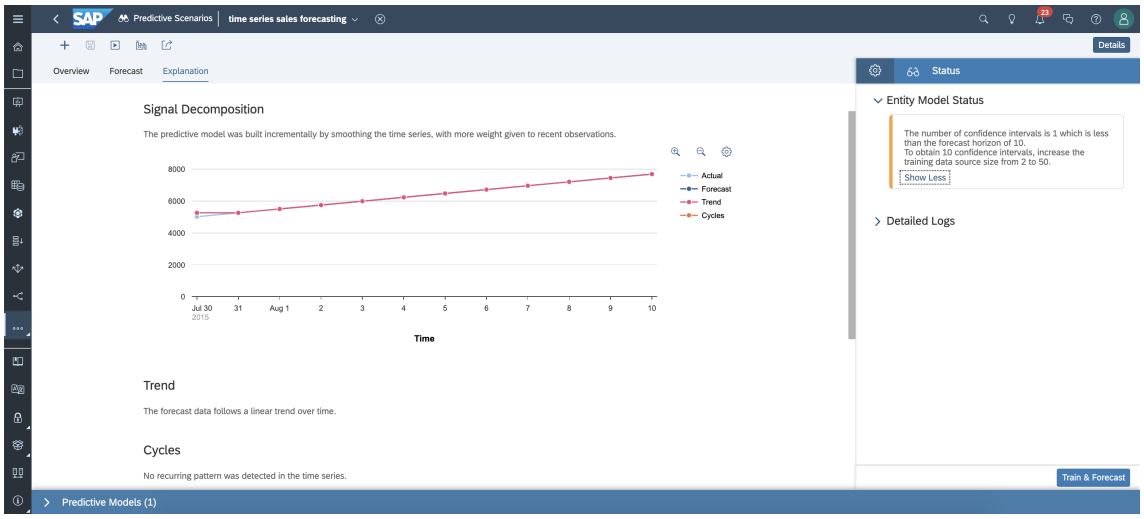


Figure 6.4.: Entity model status: increase the training data source size

We can see in Figure 6.5, that overall the training has a computed MAPE close to 0% (meaning mean loss is amlost 0). This performance is similar to the prediction confidence of the regression model, which had a score higher than 99%. The warnings are hinting that the

method could not extract enough information to provide us with solid predictions: there are warnings for every store entered in this process.

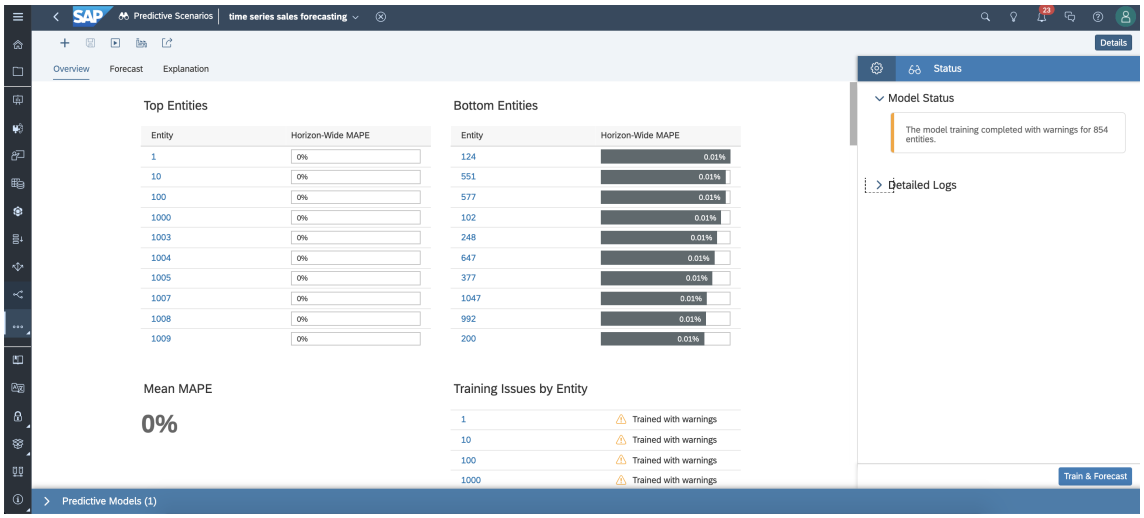


Figure 6.5.: Model training completed with warnings

Therefore, even if the forecast is trained per entity, there is not enough amount of information per store to predict daily sales for each store individually.

6.1.2. Other Limitations

Here all the other limitations of SAC are summarized:

1. Cannot get an overview of more than 2000 rows of data
2. Combined data cannot contain more than 1 million rows, a screenshot of the error is shown in Figure 6.6.
3. Cannot compare two datasets for example to see which stores from train data were not in test data.
4. Cannot convert a model after combining datasets back into a dataset, need to export as csv and import again to create dataset.
5. Cannot display correlation matrix as heatmap. The heatmap creation in SAC intends to visualize value distribution with multiple dimensions defined, an example shown in Figure 6.7.

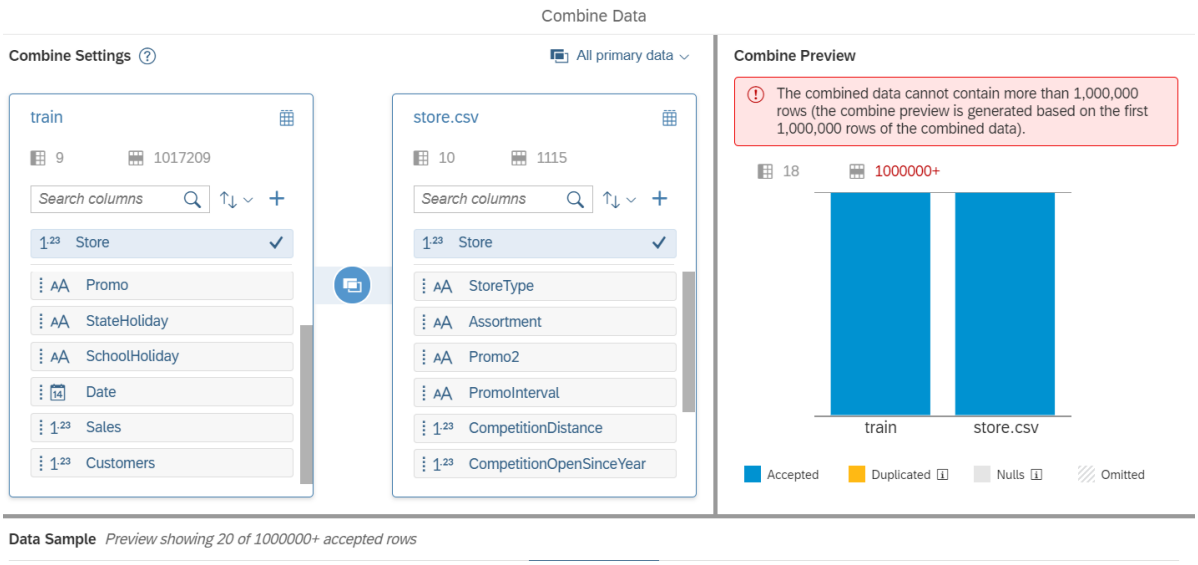


Figure 6.6.: Error for combining data with more than 1 million rows

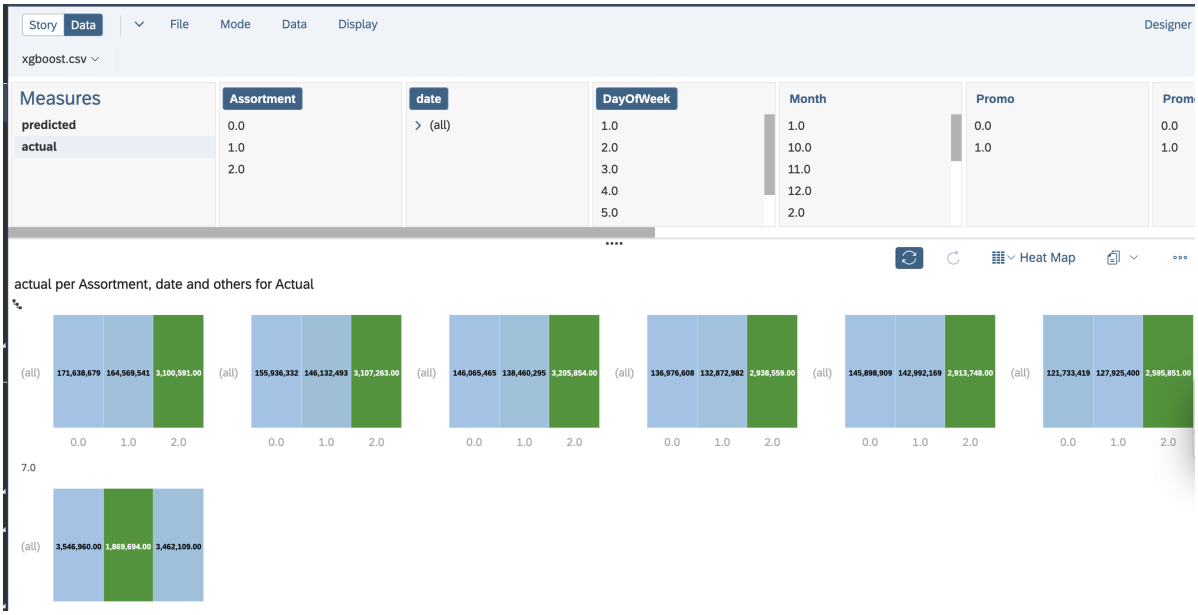


Figure 6.7.: Outlook of heatmap on SAC

6.2. For SAP HANA

6.2.1. REST API Issues when using SAP HANA applications without having Express Edition installed

In order to use SAP HANA predictive analysis functions, we found in the early steps of the project the SAP HANA Python Client API for machine learning algorithms. This enables data

scientists to access data sources from SAP HANA and continue building the machine learning models in Python. The tutorials asked directly for information and credentials, which were not available to us and since we only had access to WebIDE from SAP HANA, the other suggested options by forums to solve this issue were to switch to SAP HANA Cloud or install SAP HANA on Premise, so that we can have full control of the environment.

Since this would have meant, that we had to shift from the provided software by the Chair I17, we searched for other possible solutions and learned that we could open the ports for communication outside of HANA using NodeJs. NodeJs was already available in WebIDE and can be deployed to create a REST API connection. Thus, we continued trying to connect the Python module with HANA by using this method.

We followed multiple tutorials online from the https://developers.sap.com/ platform, video tutorials and books. However, we encountered many difficulties and errors. The following section describe a few of the problems we encountered.

6.2.1.1. Consume the OData service in a basic HTML5 module

For one of the first tutorial we followed, we launched SAP HANA WebIDE and created a GitHub repository for our project called "PYAPP". We cloned the repository into our workspace and configured the working space to X55. We added a HTML5 Module called web to our project and the UAA services. We had to pay attention to the mta.yaml file, because it must contain information related to different modules within our project: ID, version, description, resources, modules and their parameters. Next, we created an HDI module. Inside this module, we created database artifact like "store.hdbtable". We could successfully build objects in the database in this step. To upload data, we created file called "stored.hdbtabledata" and import "store.csv" and the others datasets from NTT Data. Afterwards, we created a NodeJs module and tried to send information via routes to the HTML5 module. Unfortunately, we could not see the same text as presented in the tutorial and received instead this message from the HTML5 module: "An error occurred. Please open the SAP Web IDE Run console to view more information about the error." The console message is presented in Figure 6.8. We were not able to solve this error and gave up on this tutorial.

6.2.1.2. Extend the Node.js model with SAP HANA XS Advanced

In this tutorial, we would use packages.json file to define dependencies to modules inside our project to make the installation easier. We also wanted use express to handle multiple HTTP handlers like shown in Figure 6.9. We had no prior experience in NodeJs and could not understand its logic and syntax, in order to edit the contents in files to make it work in our case. So we stopped before creating route handler and creating destination module for route in following the tutorials.

```
7/11/21 3:12:08.390 PM [APP/4-0] ERR throw new Error('No UAA configuration is
provided in non-anonymous mode.');
7/11/21 3:12:08.390 PM [APP/4-0] ERR ^
7/11/21 3:12:08.390 PM [APP/4-0] ERR
7/11/21 3:12:08.390 PM [APP/4-0] ERR Error: No UAA configuration is provided
in non-anonymous mode.
7/11/21 3:12:08.390 PM [APP/4-0] ERR at checkOptions
(/hana/shared/X55/xs/app_working/z55lp1/executionroot/ba904cda-eaaa-4fa0-8f41-
0f30ee560ad5/app/node_modules/@sap/xsjs/lib/index.js:183:11)
7/11/21 3:12:08.390 PM [APP/4-0] ERR at module.exports
(/hana/shared/X55/xs/app_working/z55lp1/executionroot/ba904cda-eaaa-4fa0-8f41-
0f30ee560ad5/app/node_modules/@sap/xsjs/lib/index.js:27:3)
7/11/21 3:12:08.390 PM [APP/4-0] ERR at Object.<anonymous>
(/hana/shared/X55/xs/app_working/z55lp1/executionroot/ba904cda-eaaa-4fa0-8f41-
0f30ee560ad5/app/server.js:29:1)
```

Figure 6.8.: Build error in WebIDE

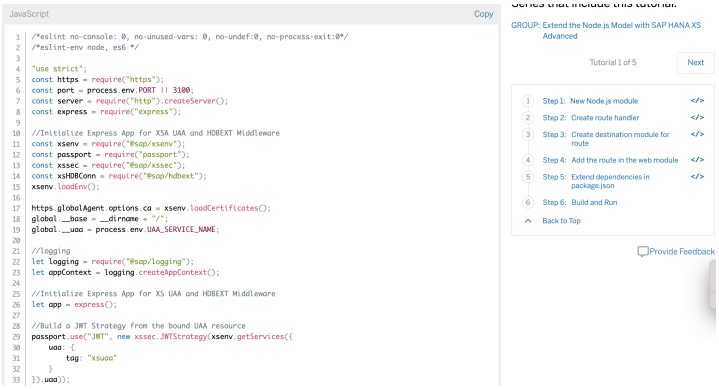


Figure 6.9.: Example of using Express to handle Http Requests

6.2.1.3. Error building a Python Application in WebIDE

Another interesting approach was to build a Python application and deploy it into HANA. We tried to follow a tutorial, where this was very good explained [1]. We followed almost every step, even though we could not run the commands which implied having SAP HANA on premise. Unfortunately, this tutorial also failed, because the self-written MTA-file could not be processed, as it can be seen in Figure 6.10.

At least, this approach made us realize that installing HANA on premise would solve our connection issue and it would not imply that we used another kind of software, than the one provided by the Chair I17 (unlike SAP HANA Cloud).

[1] https://sapbazar.com/articles/item/863-accessing-data-using-rest-apis-with-python-on-xsa

```
5:12:03 PM (Builder) Build of "/lab-course/pyapp" started.
5:12:05 PM (DIBuild) [INFO] Target platform is XSA
[INFO] Reading mta.yaml
[ERROR] MTA build failed unexpectedly.
5:12:05 PM (Builder) Build of /lab-course/pyapp failed.
```

Figure 6.10.: Application failed in WebIDE

6.2.1.4. Build the XSJS and XSODATA services to expose the data model to the user interface

We also tried to follow this approach [2], but its result generated an Excel File for which we would have to update the path in the Python script. This would not have been a good option for us, because we wanted the path to the data to be independent and not the user to search for the downloaded file and write that path in the Python script each time.

Still, when we tried this approach, we only managed to get an empty Excel File, without any data from our datasets.

[2] https://developers.sap.com/tutorials/xsa-xsjs-xsodata.html#8a4b9131-69ba-4607-9ed3-3caf23dfbfec

7. Exercises

7.1. Create your own Use Case

In order to try out all the functionalities we presented here, find yourself another dataset and try to test what we implemented in Chapters 4 and 5. As a first step, go online [1] and search for a dataset, which includes more than 3 months of data and unique hourly timestamps for transactions. This way, you can try out the time series forecast in SAC and compare it to the Random Forest and XGBoost models in Python.

7.2. Send events of Data to HANA

Now that you have HANA installed with a VM and Python connected to it, insert data row by row in a table stored in HANA using a Python module. Write another script to retrain the regression models implemented in Chapter 5 every day by using the extended dataset. This way, you can simulate real-time transactions and help our persona get insights about daily sales.

[1] https://www.kaggle.com/datasets

8. Conclusion

In our project we leveraged the power of two of SAP's innovative solutions - SAP HANA & SAP Analytics Cloud - and discovered that both solutions have their own limitations and strengths. Smart features in SAC automates the creation of predictions by providing automated classification, regression and time series techniques. Therefore it gives the ability to end-users, who are not professional data scientists, to leverage the power of making predictions. However, it does not offer the possibility for end-users to change the logic of the algorithms being used. On the other hand, SAP HANA provides the privilege of directly connecting Python script to HANA, where they can implement their own logic and algorithms for predictive functionalities, but this might only be applicable for professional data scientists who have adequate programming skills.

In our opinion, SAP HANA Cloud and SAP Analytics Cloud would be the best solution for future projects in the area of Data Analytics and Forecasts. The main benefit of using SAP HANA Cloud is that the user can always access the data source, without starting a virtual machine or searching for it on a local machine. Therefore, the results of analysis would be delivered fast back to its users, or even better, be directly stored in HANA, without manual importing. Then, SAC could directly access these results using its Connection to Live Data function, which can access the instance of HANA, where this data is stored. This was a feature we actually wanted to apply from the beginning, but did not have the opportunity.

For future releases, it would be practical to have a console in SAC, which would contain a Python or Spark environment already installed. This way, the user could solve the limitations we described in Section 6.1, without exiting the platform, and have a blend of both automated features and the possibility to implement its own logic. The future is about processing Big Data and such a smart Cloud platform for Analytics should therefore be able to work with more than 1 Million rows at a time if it wants to be competitive.

Bibliography

[1] S. Parsons. "Principles of Data Mining by David J. Hand, Heikki Mannila and Padhraic Smyth, MIT Press, 546 pp., £34.50, ISBN 0-262-08290-X." In: *Knowledge Eng. Review* 19.2 (2004), pp. 183–184. URL: http://dblp.uni-trier.de/db/journals/ker/ker19.html#Parsons04c.

[2] J. D. Kelleher, B. MacNamee, and A. D'Arcy. *Fundamentals of Machine Learning for Predictive Data Analytics: Algorithms, Worked Examples, and Case Studies*. Cambridge, MA: MIT Press, 2015. ISBN: 978-0-262-02944-5.

[3] T. Runkler. *Data Analytics: Models and Algorithms for Intelligent Data Analysis*. Lehrbuch - Springer. Vieweg+Teubner Verlag, 2012. ISBN: 9783834825889. URL: https://books.google.de/books?id=PaylsdmgkkIC.

[4] H. AKOĞLU. "User's guide to correlation coefficients". In: *TURKISH JOURNAL OF EMERGENCY MEDICINE* (2018), pp. 91–93.

[5] S. Bolboaca and L. Jäntschi. "Pearson versus Spearman, Kendall's Tau Correlation Analysis on Structure-Activity Relationships of Biologic Active Compounds". In: *Leonardo Journal of Sciences* 9 (July 2006).

[6] A. L. Samuel. "Some Studies in Machine Learning Using the Game of Checkers". In: 3.3 (1959). ISSN: 0018-8646. DOI: 10.1147/rd.33.0210. URL: https://doi.org/10.1147/rd.33.0210.

[7] C. Sammut and G. I. Webb. *Encyclopedia of Machine Learning*. 1st. Springer Publishing Company, Incorporated, 2011. ISBN: 0387307680.

[8] G. James, D. Witten, T. Hastie, and R. Tibshirani. *An Introduction to Statistical Learning: with Applications in R*. Springer Texts in Statistics. Springer New York, 2013. ISBN: 9781461471387. URL: https://books.google.de/books?id=qcI%5C_AAAAQBAJ.

A. Code Listings in SQL

Listing A.1: Create Schema and Tables

```
create schema input;

set schema input;

create column table "Store" (
        "Store" INTEGER ,
        "StoreType" CHARACTER,
        "Assortment" CHARACTER not null,
        "CompetitionDistance" INTEGER,
        "CompetitionOpenSinceMonth" INTEGER,
        "CompetitionOpenSinceYear" INTEGER,
        "Promo2" INTEGER,
        "Promo2SinceWeek" INTEGER,
        "Promo2SinceYear" INTEGER,
        "PromoInterval" NVARCHAR(40),
        PRIMARY KEY ("Store")
);

create column table "Test" (
        "Id" INTEGER,
        "Store" INTEGER,
        "DayOfWeek" INTEGER,
        "Date" VARCHAR(20),
        "Open" INTEGER,
        "Promo" INTEGER,
        "StateHoliday" VARCHAR(1),
        "SchoolHoliday" INTEGER
);

create column table "Train" (
        "Store" INTEGER,
        "DayOfWeek" INTEGER,
        "Date" VARCHAR(20),
        "Sales" INTEGER,
```

```
        "Customers" INTEGER,
        "Open" INTEGER,
        "Promo" INTEGER,
        "StateHoliday" VARCHAR(1),
        "SchoolHoliday" INTEGER
);
```

Listing A.2: Create User Roles

```
create user ML_DEV password hanaLEARN1 no force_first_password_change;
GRANT ALL PRIVILEGES ON SCHEMA INPUT TO ML_DEV;
```

Listing A.3: View Ports

```
SELECT SERVICE_NAME, PORT, SQL_PORT, (PORT + 2) HTTP_PORT
FROM SYS.M_SERVICES
WHERE ((SERVICE_NAME = 'indexserver' and COORDINATOR_TYPE = 'MASTER')
      OR SERVICE_NAME = 'xsengine');
```

B. Code Listings in Python

Listing B.1: Connect to SAP HANA and get Data for Analysis

```
conn = dataframe.ConnectionContext("192.168.0.8", 39015, "ML_DEV", "hanaLEARN1")
#Get the data from the tables in schema input
store_df = conn.table("Store", schema="INPUT")
store_df.head(5).collect()
train_df = conn.table("Train", schema="INPUT")
train_df.head(5).collect()
test_df = conn.table("Test", schema="INPUT")
test_df.head(5).collect()

#transform hana_dataframe to pandas
store_df = store_df.collect()
train_df = train_df.collect()
test_df = test_df.collect()
```

Listing B.2: Preprocessing: Missing Data

```
#View how much information is missing per column
def nans_per_columns(df):
    nans = df.isnull().sum(axis=0)
    nans_percent = 100 * df.isnull().sum() / len(df)
    nans_table = pd.concat([nans, nans_percent], axis=1)
    nans_table_new = nans_table.rename(columns={0: 'Missing Values',
                                                1: '% of Total Values'})
    return nans_table_new
```

Listing B.3: Preprocessing: Clean Store Dataset

```
#All the nans came because Promo2=0 -> replace these Nans with 0
store_df.loc[store_df["Promo2"] == 0 & store_df['Promo2SinceWeek'].isna() &
             store_df['Promo2SinceYear'].isna() &
             store_df['PromoInterval'].isna(),
             ["Promo2SinceWeek", "Promo2SinceYear", "PromoInterval"]]=0

#We delete the 3 rows that miss values at CompetitionDistance
store_df_clean = store_df.dropna(subset=["CompetitionDistance"])
```

```
#Fill in missing data for CompetitionOpenSince with the most frequent values
store_df_clean = store_df_clean.fillna(store_df_clean.mode().iloc[0])
```

Listing B.4: Preprocessing: Encode Characters to Numbers for categorical Variables

```
def encode_factor(df, colnames):
    for colname in colnames:
        i = 0
        for val in df[colname].unique():
            df.loc[df[colname]== val, colname] = i
            i +=1
        df[colname] = df[colname].astype(int, copy=False)

store_df_clean['PromoInterval'].unique()
encode_factor(store_df_clean,['PromoInterval','StoreType','Assortment'])
encode_factor(train_df,['StateHoliday'])
```

Listing B.5: Correlation Analysis

```
#Complete_train_df is cleaned store & train dataset joint on store(ID)
corr_df = complete_train_df.corr()
plt.figure(figsize = (20,15))
sn.heatmap(complete_train_df.corr(), annot=True,vmin=-1, vmax=1, cmap="RdYlGn")
```

Listing B.6: Training Model using Random Forest Regression

```
from sklearn.ensemble import RandomForestRegressor

X_train_rf, X_test_rf, y_train_rf, y_test_rf = train_test_split(features_data,
                                                                target_data_np,
                                                                test_size=0.30,
                                                                random_state=42)

regressor = RandomForestRegressor(n_estimators=100, random_state=42)
regressor.fit(X_train_rf, y_train_rf)
y_pred_rf = regressor.predict(X_test_rf)

#Cross Validation
cv = KFold(random_state=42, n_splits = 10)#default n_splits = 5
cv_scores = cross_val_score(regressor, X_test_rf, y_test_rf,
                            scoring='neg_mean_absolute_error',
                            cv = cv, n_jobs = -1, error_score='raise')
```

Listing B.7: Training Model using XGBoost

```
from xgboost import XGBRegressor

model = XGBRegressor(n_estimators=1000, learning_rate=0.05)
model.fit(X_train, y_train, early_stopping_rounds=100,eval_set=[(X_test,y_test)])
y_prediction = model.predict(X_test)

#Cross Validation
cv = KFold(n_splits=10, shuffle=True)
scores = cross_val_score(model, X_test, np.ravel(y_test), cv= cv)
```

Listing B.8: Computation of the Performance of the trained Models

```
df_acc_pred["predicted"] = y_prediction #predicted values
df_acc_pred["actual"] = y_test #original hidden values

#Delete entries where actual value = 0. Division by 0 results in infinity
df = df_acc_pred.loc[df_acc_pred["actual"] != 0]
#Mean absolute percentage error
mape = 100 * ((df["actual"] - df["predicted"])/ df["actual"])
accuracy = 100 - abs(np.mean(mape))
```