Pro

Arka Ghosh

arkaghosh.nb@gmail.com

Abstract

This document outlines the steps undertaken to build various ML models for predicting the Sales for the problem statement defined in MachineHack 2021 competition.

Analytics olympiad ‘21

Machine Learning Process Document

Contents

[**1.** **Import Python Libraries** 1](#_Toc87696386)

[**2.** **Instantiate H2O server** 1](#_Toc87696387)

[**Attempts to start and/or connect to and H2O instance.** 1](#_Toc87696388)

[**3.** **Read data from Train and Test datasets** 2](#_Toc87696389)

[**4.** **Convert H2O frame to Pandas dataframe** 2](#_Toc87696390)

[**5.** **Exploratory Data Analysis** 3](#_Toc87696391)

[**Profile Report** 3](#_Toc87696392)

[**Correlation Matrix** 3](#_Toc87696393)

[**Check Missing values** 4](#_Toc87696394)

[**View Interactions** 4](#_Toc87696395)

[**Frequency Plot(Histograms)** 5](#_Toc87696396)

# **Import Python Libraries**

Firstly, we will need to install many libraries in Python that we are going to use in our analysis. Apart from a few common ones like Pandas, NumPy, ScipySklearn, we also need to import modules for particular algorithms like Linear Regression, XGBoost, H20 etc. Each of these libraries need to be first installed on our computer by doing ‘pip install <python-module>. I have done this analysis on my Anaconda – Jupyter Notebook. So, the installation can be done by launching a command prompt in the same tool.

# **Instantiate H2O server**

H2O is an open source, in-memory, distributed, fast, and scalable machine learning and predictive analytics platform that allows you to build machine learning models on big data and provides easy productionization of those models in an enterprise environment.

H2O’s AutoML can be used for automating the machine learning workflow, which includes automatic training and tuning of many models within a user-specified time-limit.

## **Attempts to start and/or connect to and H2O instance.**

**# max\_mem\_size** - A character string specifying the maximum size, in bytes, of the memory allocation pool to H2O. This value must a multiple of 1024 greater than 2MB.

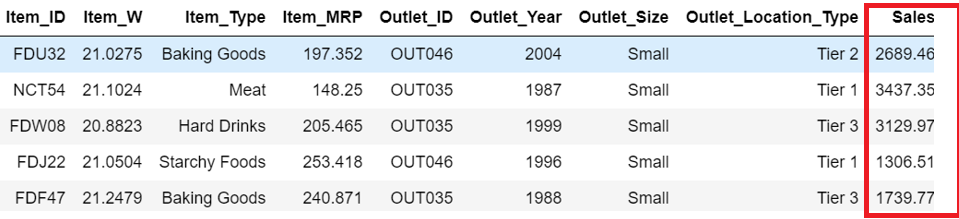
# Append the letter m or M to indicate megabytes, or g or G to indicate gigabytes.

**# nthreads -** Number of threads in the thread pool. This relates very closely to the number of CPUs used. -1 means use all CPUs on the host (Default). A positive integer specifies the number of CPUs directly.

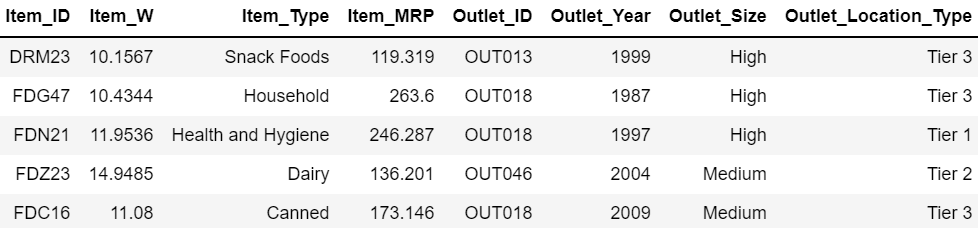
# **Read data from Train and Test datasets**

The import function is a parallelized reader and pulls information from the server from a location specified by the client. The path is a server-side path. This is a fast, scalable, highly optimized way to read data. H2O pulls the data from a data store and initiates the data transfer as a read operation.

**train**:



**test**:



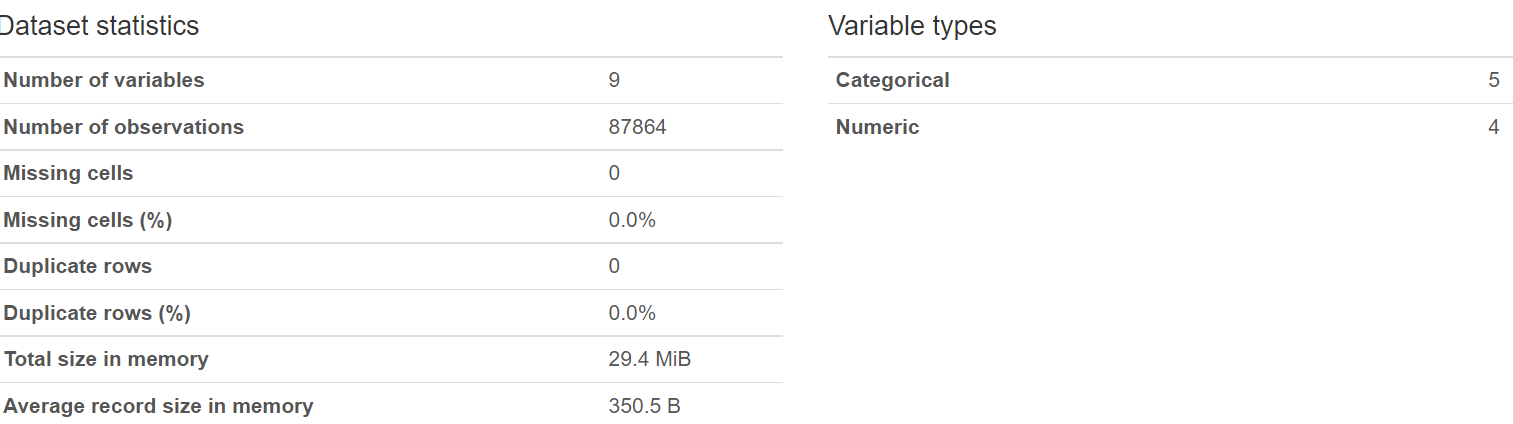
# **Convert H2O frame to Pandas dataframe**

To do data processing and feature engineering, we would like to work on a normal Pandas dataframe. We would also like to apply some base algorithms on this data. So we will convert H2O dataframe to a Pandas dataframe. In the final stage, when we will do a stacked-ensemble modelling, we will convert it back to a H2O dataframe.

# **Exploratory Data Analysis**

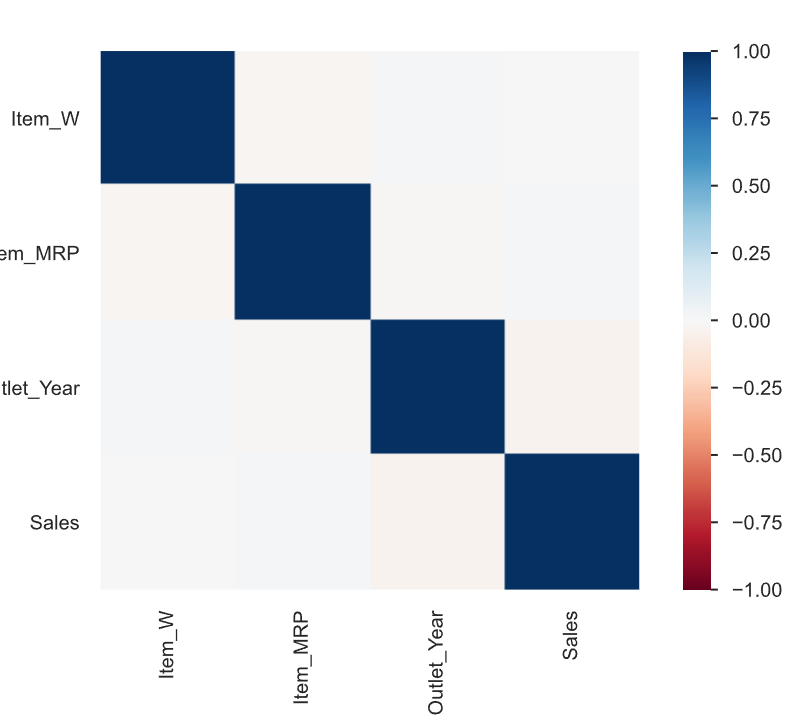
## **Profile Report**

The pandas df.describe() function is great but a little basic for serious exploratory data analysis. pandas\_profiling extends the pandas DataFrame with df.profile\_report() for quick data analysis.



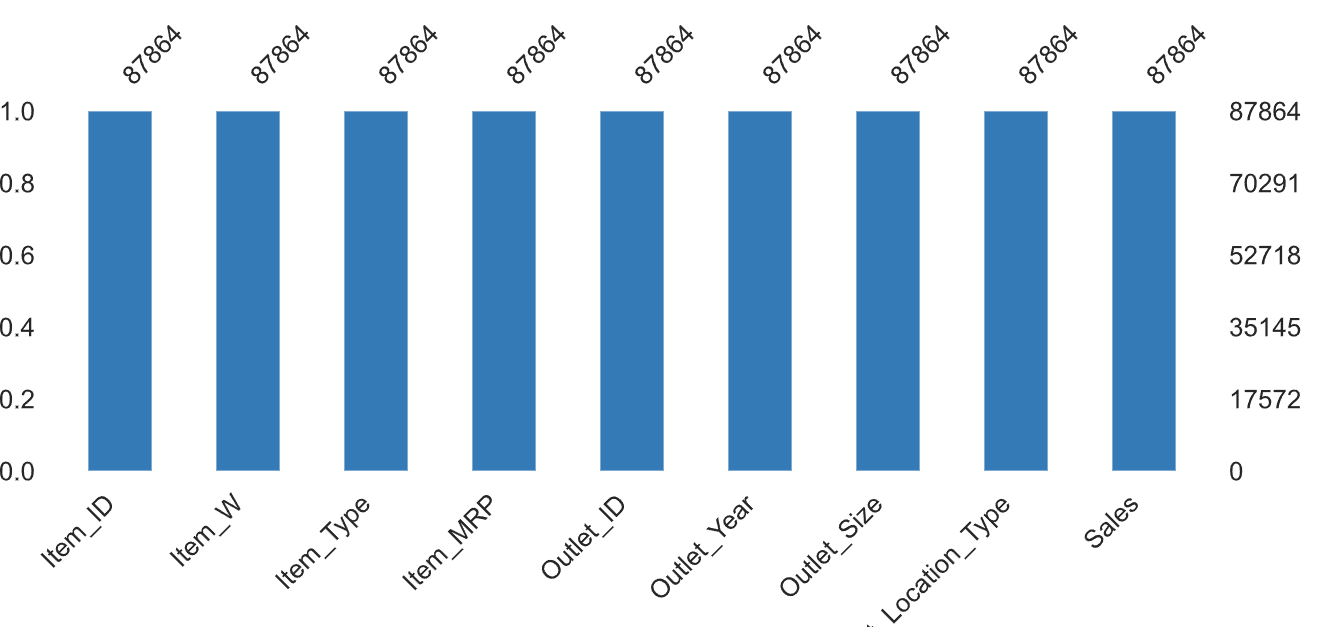
## **Correlation Matrix**

For the numeric features, we will build a correlation matrix, just to verify if there is any particular features that has a high degree of association(positive or negative) with Sales.

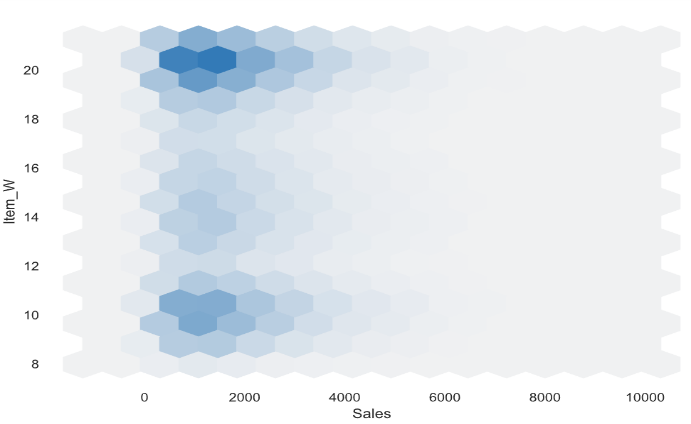
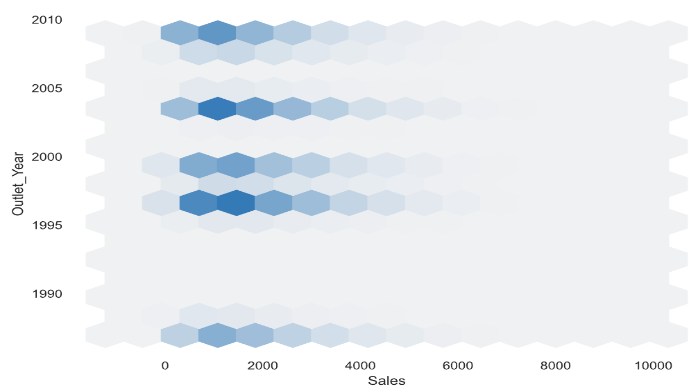
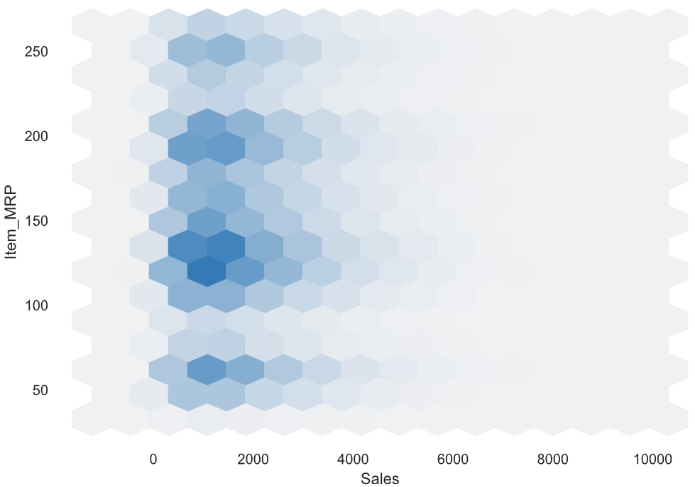
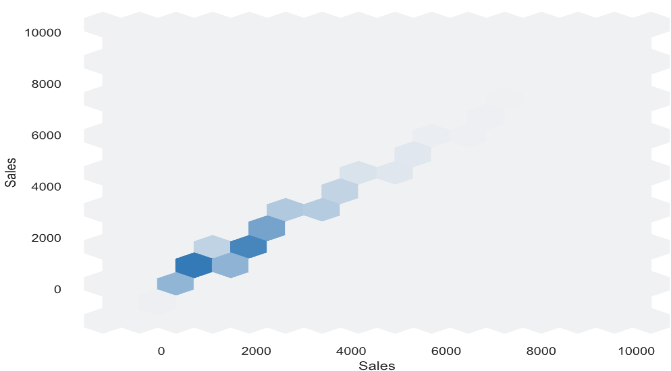


## **Check Missing values**

Sometimes, datasets have few columns with high percentage of missing values or NaN values. These values need to be treated before applying any ML algorithm. We can do a mean-imputation if the count of such missing values is small. But if a column has too many NA values, its better to drop it from the dataframe. Here, we don’t see any missing values , since all columns have exactly 87864 records.



## **View Interactions**

Plot any 2 columns to see how the data is distributed on x-axis and y-axis. If we want to see Sales plotted against other numeric features, we can do it.  

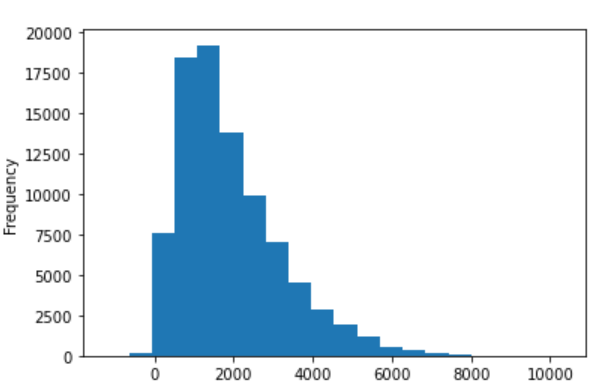
## **Frequency Plot(Histograms)**

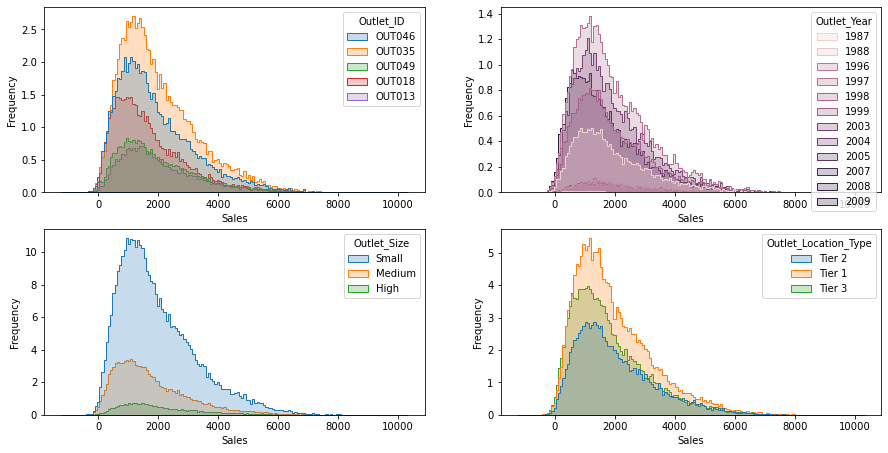
### **Sales**

A histogram is a bar graph-like representation of data that buckets a range of outcomes into columns along the x-axis. The y-axis represents the number count or percentage of occurrences in the data for each column and can be used to visualize data distributions.

We have created a histogram with 20 bins for Sales. Most frequent occurrence is the range between Rs.1000 - 2000. This means that maximum products that are being sold fall in this range. Beyond this , the frequency of sales is on a downward slope.

Next, we plot the frequency charts of Sales as per Outlet\_ID, Outlet\_Year, Outlet\_Size and Outlet\_Location\_Type.





### **Item Weight**

We have created a histogram with 20 bins for Item Weight. Most frequent occurrence is when the items weighs 10, 14 and 22. This means that maximum item-weights of products that are being sold fall in this range. Beyond this , the frequency of Item Weight can vary.

Next, we plot the frequency charts of Item Weight as per Outlet\_ID, Outlet\_Year, Outlet\_Size and Outlet\_Location\_Type.

