Pro

Arka Ghosh

arkaghosh.nb@gmail.com

Abstract

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

Analytics olympiad ‘21

Machine Learning Solution Overview

Contents

[**Import Python Libraries** 3](#_Toc87719745)

[**Instantiate H2O server** 3](#_Toc87719746)

[Attempts to start and/or connect to and H2O instance. 3](#_Toc87719747)

[**Read data from Train and Test datasets** 3](#_Toc87719748)

[Convert H2O frame to Pandas dataframe 4](#_Toc87719749)

[**Exploratory Data Analysis** 4](#_Toc87719750)

[Profile Report 4](#_Toc87719751)

[Correlation Matrix 4](#_Toc87719752)

[Check Missing values 5](#_Toc87719753)

[View Interactions 5](#_Toc87719754)

[**Frequency Plot(Histograms)** 6](#_Toc87719755)

[Sales 6](#_Toc87719756)

[Item Weight 7](#_Toc87719757)

[Item MRP 8](#_Toc87719758)

[**Box n Whiskers Plot** 9](#_Toc87719759)

[**Times Series Analysis** 10](#_Toc87719760)

[Combined Sales 10](#_Toc87719761)

[Sales by Outlet 11](#_Toc87719762)

[Sales by Outlet Size 11](#_Toc87719763)

[Sales by Outlet Location Type 12](#_Toc87719764)

[Sales by Item Type 12](#_Toc87719765)

[**Time Series Forecast – Prophet** 12](#_Toc87719766)

[5-year predictions 13](#_Toc87719767)

[Upcoming Trend 14](#_Toc87719768)

[**Statistical Analysis** 14](#_Toc87719769)

[Normality Distribution Tests 14](#_Toc87719770)

[Quantile-Quantile Plot 14](#_Toc87719771)

[Shapiro-Wilk Test 15](#_Toc87719772)

[Anderson-Darling Test 15](#_Toc87719773)

[Spearman Rank Correlation 16](#_Toc87719774)

[Kruskal-Wallis H Test 17](#_Toc87719775)

[Relationship of Sales & Outlet Size 17](#_Toc87719776)

[Relationship of Sales & Outlet Location Type 17](#_Toc87719777)

[Kruskal-Wallis Test Effect Size 18](#_Toc87719778)

[Outlet Size: 18](#_Toc87719779)

[Outlet Location Type: 18](#_Toc87719780)

[**Handle -ve Sales figures** 18](#_Toc87719781)

[**Outlier Detection** 18](#_Toc87719782)

[IQR method of outlier detection 18](#_Toc87719783)

[Outlier Handling by Winsorization 19](#_Toc87719784)

[**Feature Engineering** 19](#_Toc87719785)

[Derive the Item\_Group based on Item\_Type 19](#_Toc87719786)

[Derive the Outlet\_Age based on Outlet\_Year 19](#_Toc87719787)

[Check no NULL values in train/test 19](#_Toc87719788)

[Categorical to Numeric Conversion 19](#_Toc87719789)

[Label Encoding 19](#_Toc87719790)

[One Hot Encoding 19](#_Toc87719791)

[**Machine Learning Modelling** 20](#_Toc87719792)

[Multivariate Linear Regression 20](#_Toc87719793)

[Random Forest Regressor 20](#_Toc87719794)

[Deep Neural Net Regressor 20](#_Toc87719795)

[XGBoost Regressor 20](#_Toc87719796)

[Feature Importance – XGBoost 21](#_Toc87719797)

[Hyptertuned XGBoost Regressor 22](#_Toc87719798)

[Best Model: Stacked Ensemble Learning 23](#_Toc87719799)

[Model Summary 23](#_Toc87719800)

[AutoML Model Leaderboard 24](#_Toc87719801)

[Ensemble Exploration 24](#_Toc87719802)

[**Conclusion** 27](#_Toc87719803)

# **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.

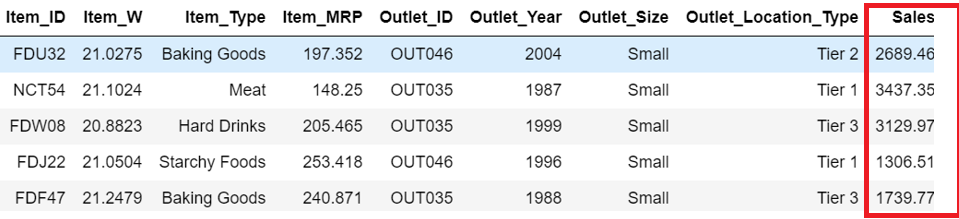
**# max\_mem\_size** - A character string specifying the maximum size, in bytes, of the memory allocation pool to H2O. This value must a multiple of 1024 greater than 2MB.

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

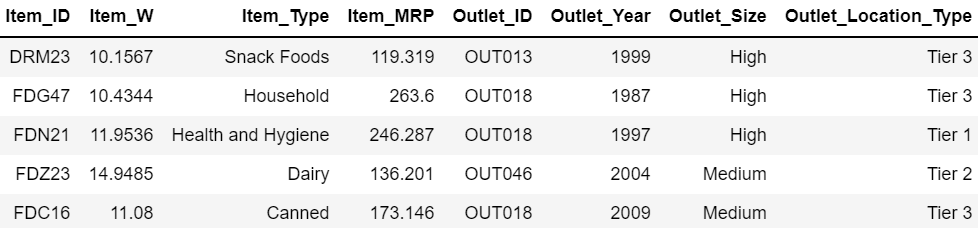
**# nthreads -** Number of threads in the thread pool. This relates very closely to the number of CPUs used. -1 means use all CPUs on the host (Default). A positive integer specifies the number of CPUs directly.

# **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**:



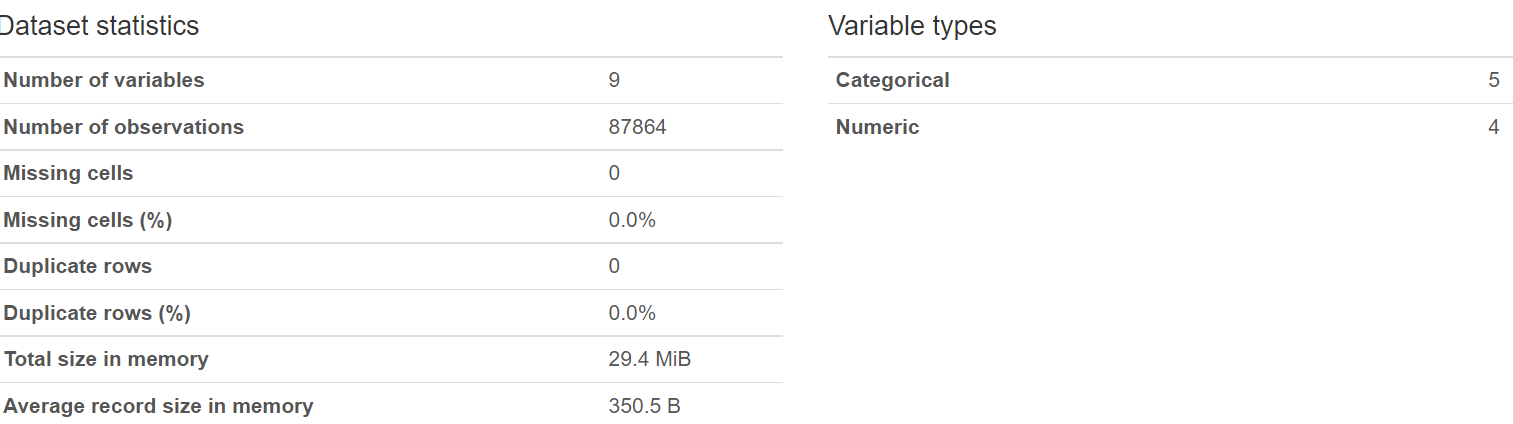
## Convert H2O frame to Pandas dataframe

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**

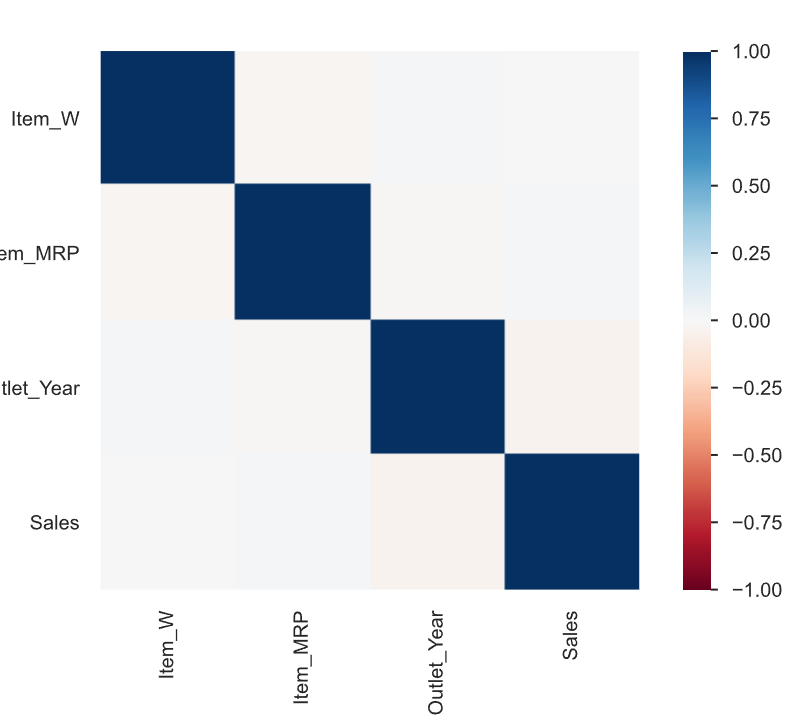
## Profile Report

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.



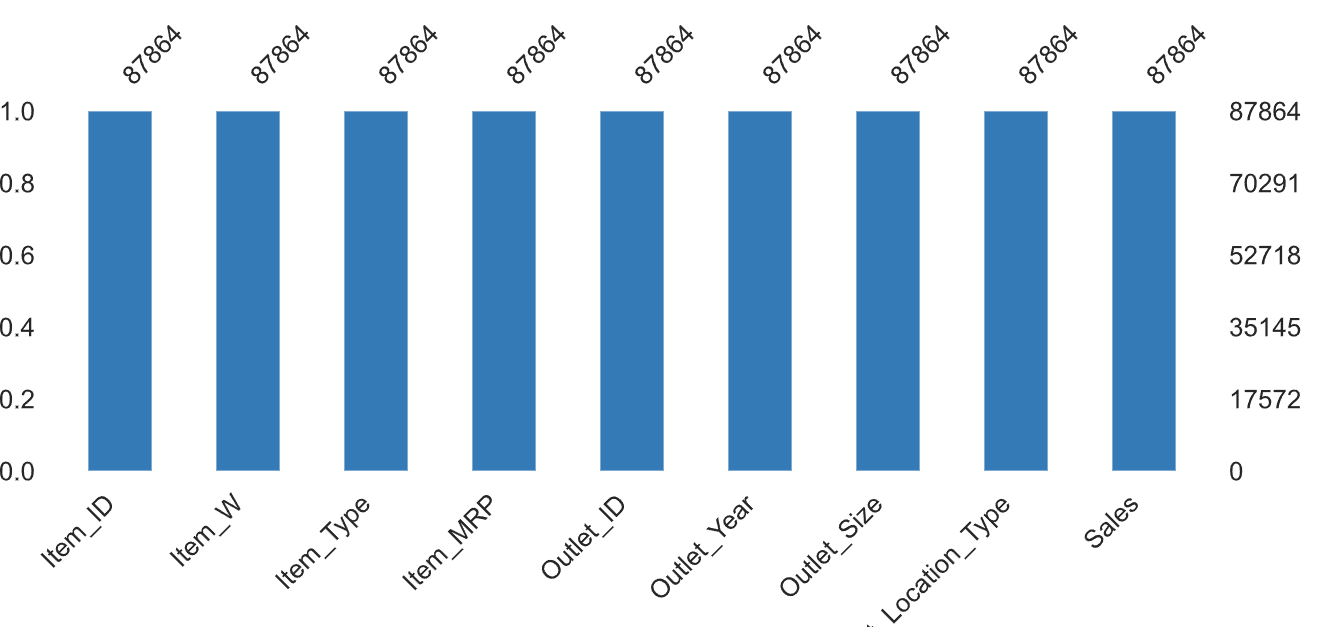
## Correlation Matrix

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.

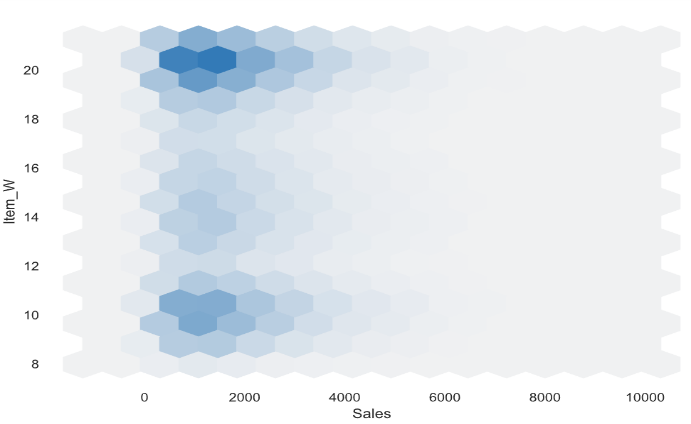
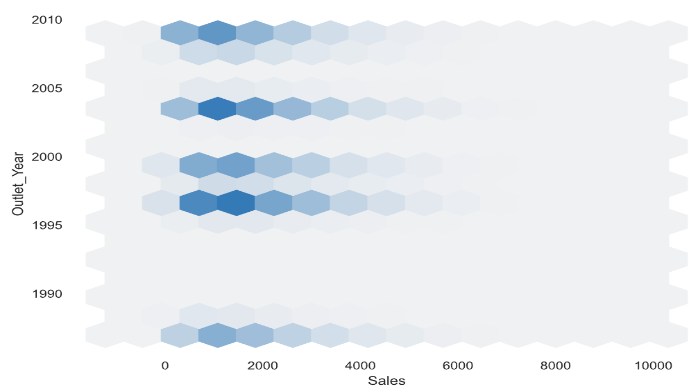
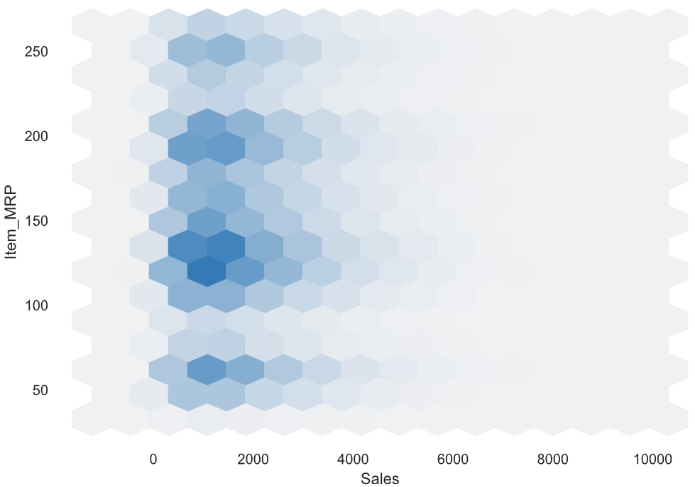
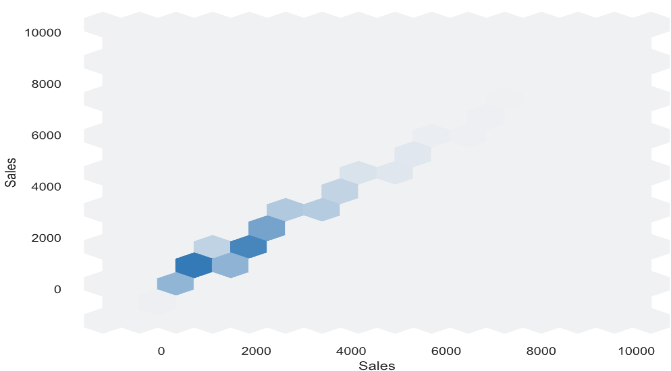


## Check Missing values

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.



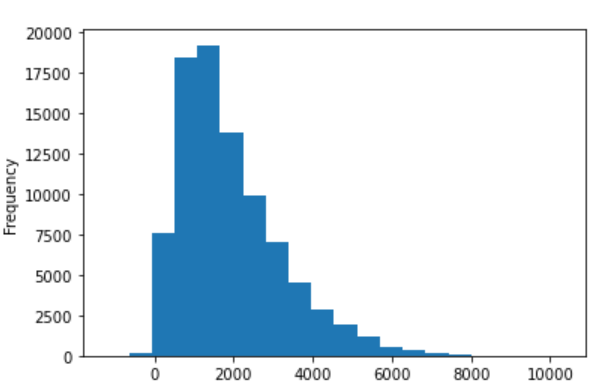
## View Interactions

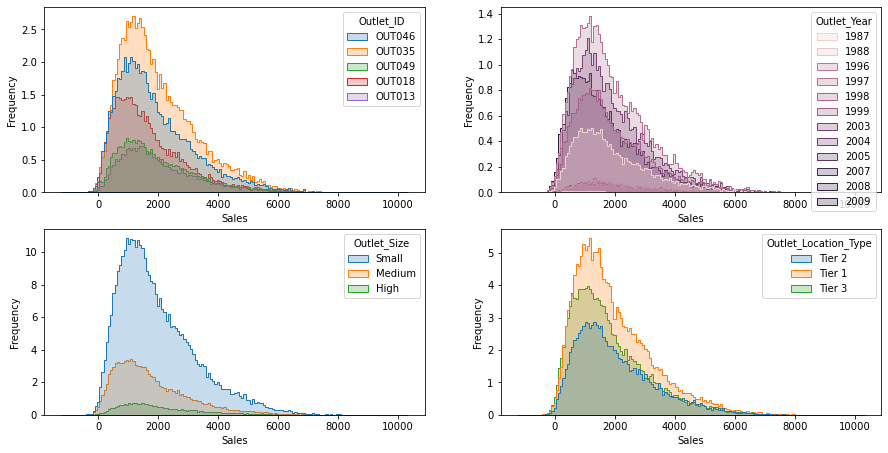
Plot any 2 columns to see how the data is distributed on x-axis and y-axis. If we want to see Sales plotted against other numeric features, we can do it.  

## **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.

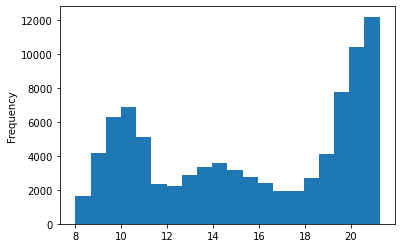
We have created a histogram with 20 bins for Sales. Most frequent occurrence is the range between Rs.1000 - 2000. This means that maximum products that are being sold fall in this range. Beyond this , the frequency of sales is on a downward slope. Next, we plot the frequency charts of Sales as per Outlet\_ID, Outlet\_Year, Outlet\_Size and Outlet\_Location\_Type.

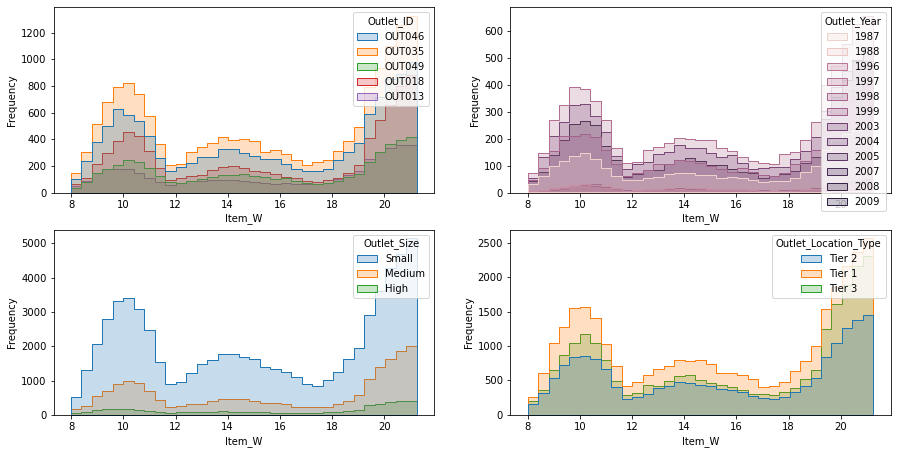


### Item Weight

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.

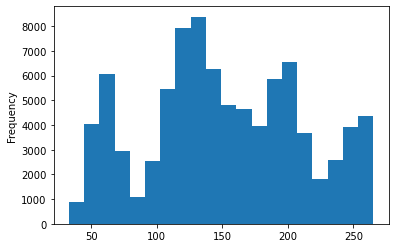


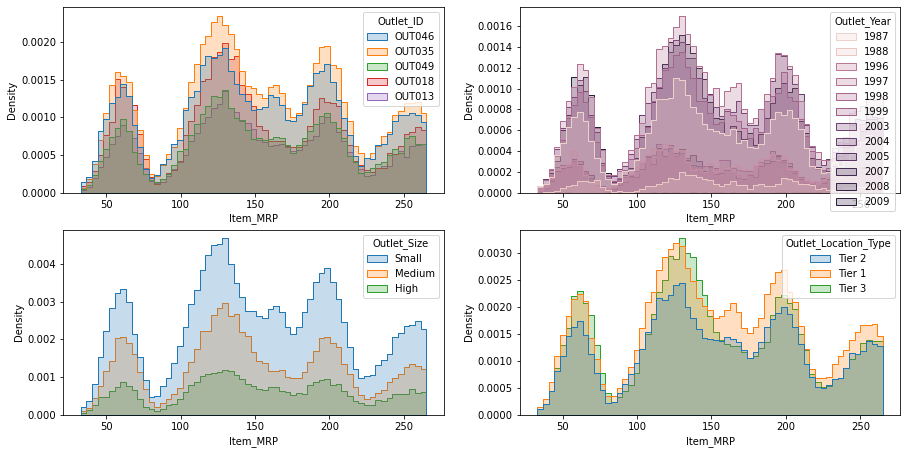


### Item MRP

We have created a histogram with 20 bins for Item MRP. Most frequent occurrence is when the items MRPS is around 60, 125 and 200. This means that maximum item-MRP of products that are being sold fall in this range. Beyond these values, the MRP can vary.

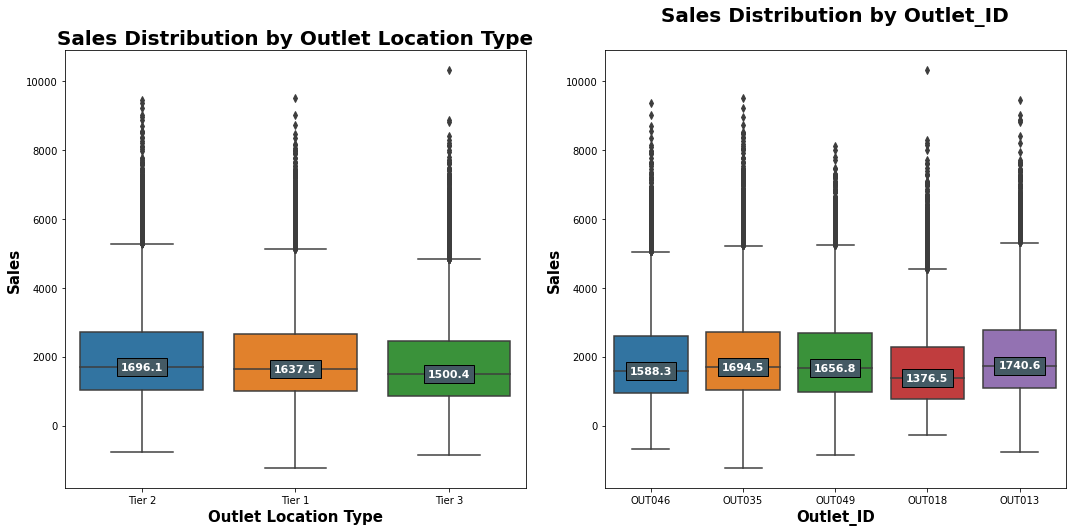
Next, we plot the frequency charts of Item MRP as per Outlet\_ID, Outlet\_Year, Outlet\_Size and Outlet\_Location\_Type.

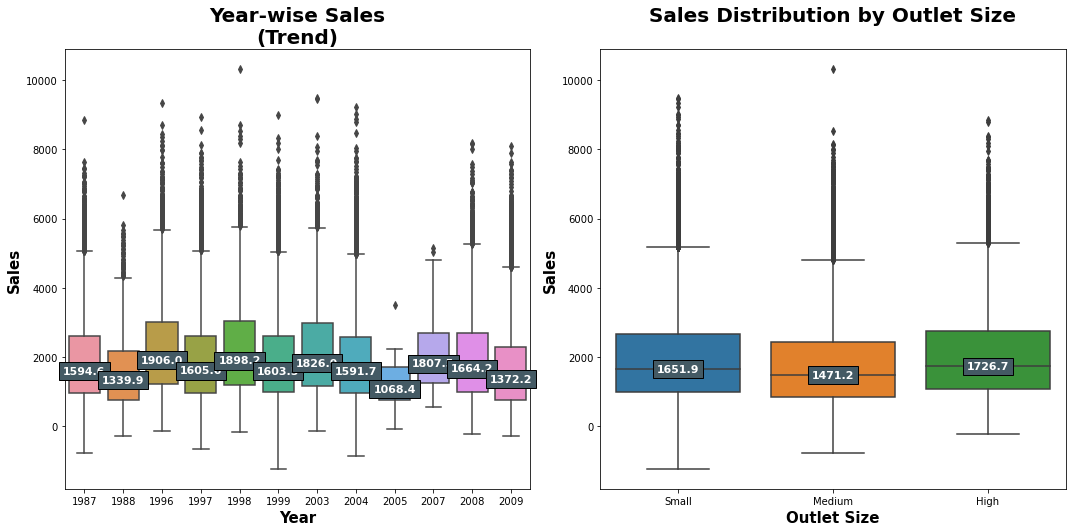




## **Box n Whiskers Plot**

We plot Sales numbers to check KPI : Year-wise Sales(Trend) and Sales Distribution by Outlet Size**.** The numbers here indicate the median values of Sales every year and by Outlet size. Median sales was highest in the year 1996 and the lowest in 2005. Big outlets have maximum median sales and medium sized outlets have minimum median sales. Tier 2 cities have highest median sales and OUT013 had the highest median sales.

****

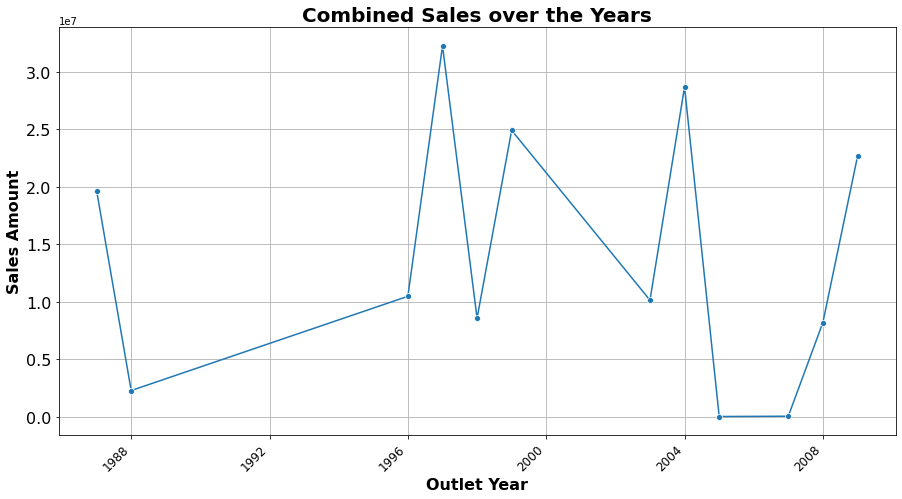


## **Times Series Analysis**

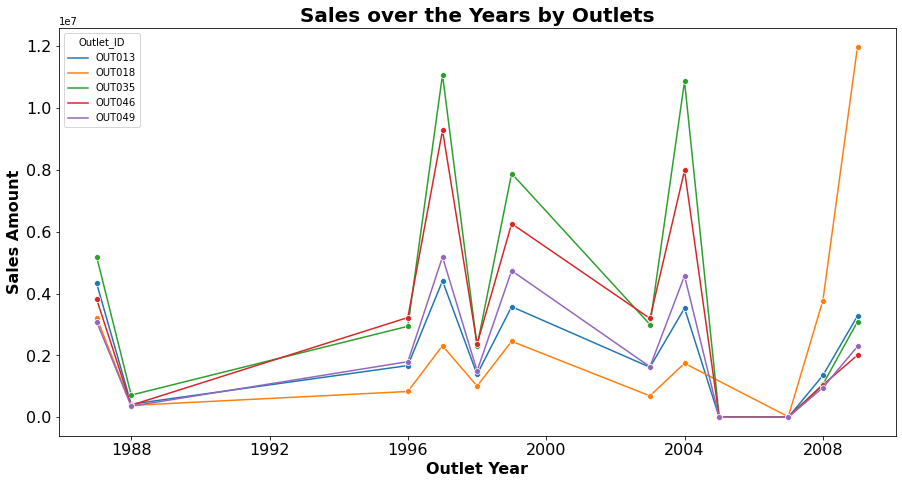
Time series analysis is a specific way of analyzing a sequence of data points collected over an interval of time. In time series analysis, analysts record data points at consistent intervals over a set period of time rather than just recording the data points intermittently or randomly.

Here, we see that combined sales amount varied a lot over the years although in 2005 there was a sharp drop followed by a steep increase in 2006-2007 period.

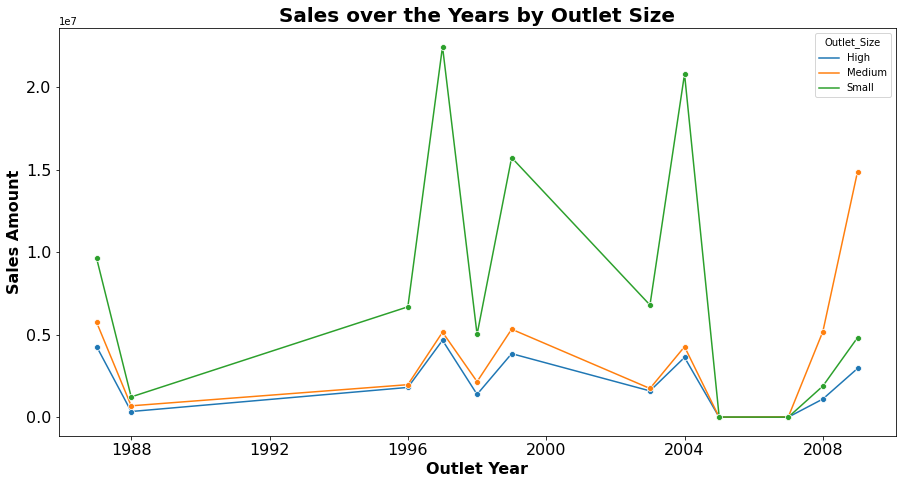
### Combined Sales



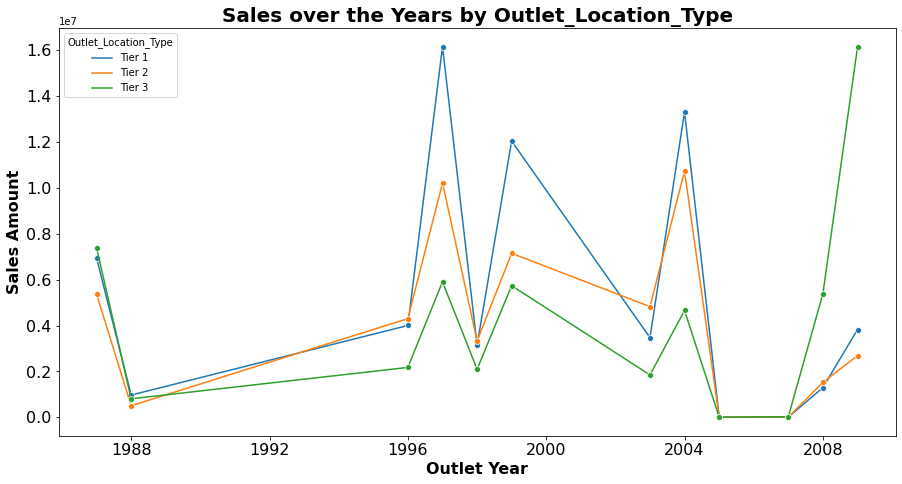
### Sales by Outlet



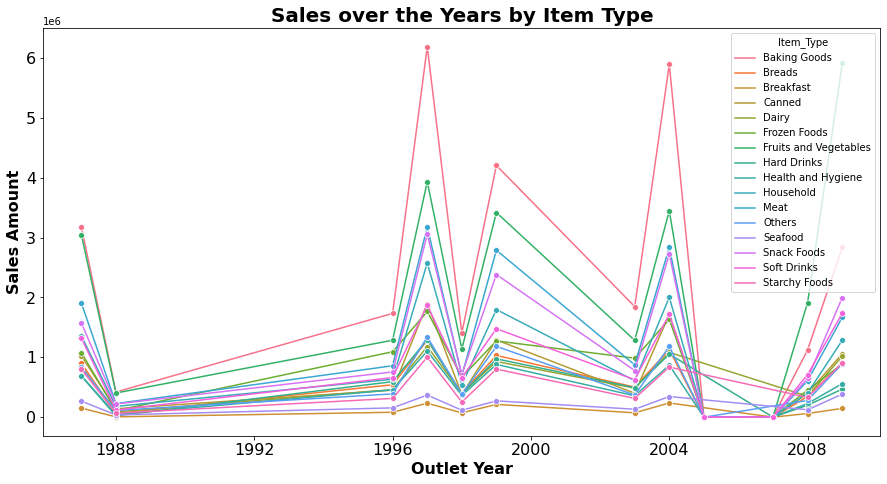
### Sales by Outlet Size



### Sales by Outlet Location Type



### Sales by Item Type



## **Time Series Forecast – Prophet**

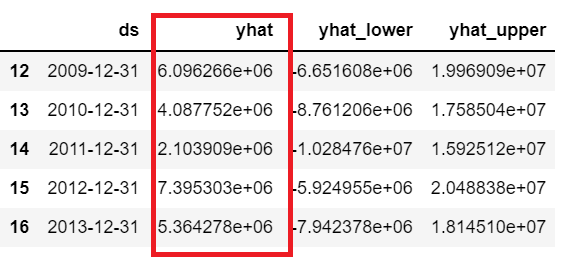
We will try and forecast the combined sales value for the next 5 years based on Facebook’s Prophet algorithm. Prophet is a procedure for forecasting time series data based on an additive model where non-linear trends are fit with yearly, weekly, and daily seasonality, plus holiday effects. It works best with time series that have strong seasonal effects and several seasons of historical data. Prophet is robust to missing data and shifts in the trend, and typically handles outliers well.

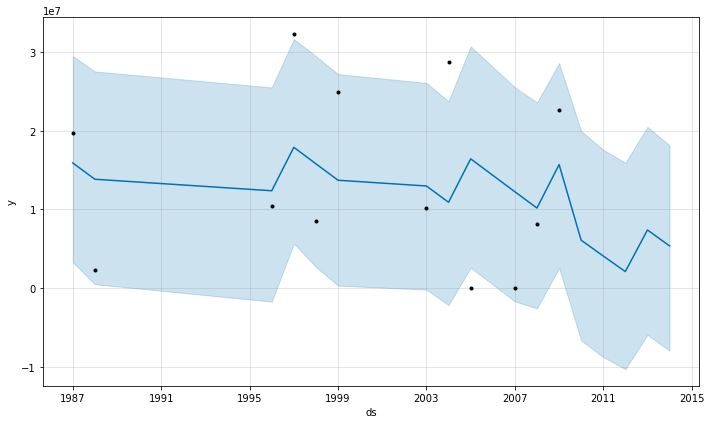
When creating the prophet models, I set the changepoint prior to 0.15, up from the default value of 0.05. This hyperparameter is used to control how sensitive the trend is to changes, with a higher value being more sensitive and a lower value less sensitive. This value is used to combat one of the most fundamental trade-offs in machine learning: bias vs. variance.

Here, under the predictions table, we are only concerned with ds, yhat\_lower, yhat\_upper, and yhat because these are the variables that will give us the predicted results with respect to the date specified.

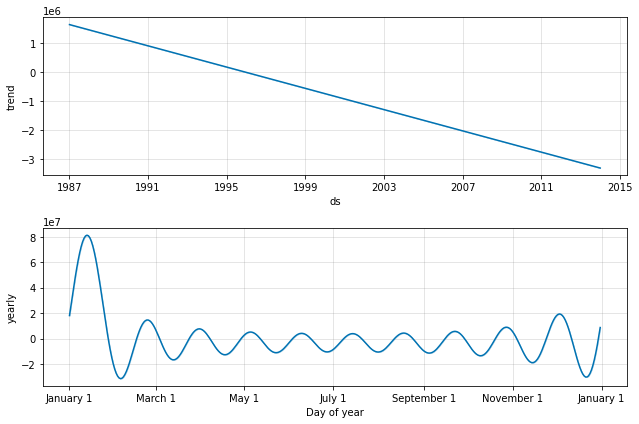
### 5-year predictions

**yhat means the predicted output** based on the input fed to the model, yhat\_lower, and upper means the upper and lower value that can go based on the predicted output that is, the fluctuations that can happen.





### Upcoming Trend

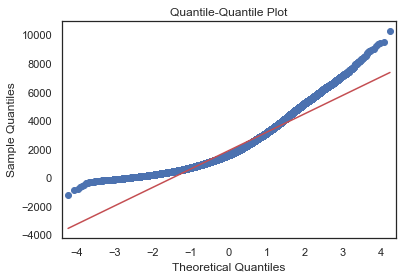
Here, we can see the trends with respect to year and cyclicity in a year. The first graph represents a strong decreasing trend as we progress through the 5 years from 2009 to 2013 and the latter shows a fluctuating trend in the monthly sales. For most months it is steady but towards the start of the year from January to February there is strong downward fluctuation.

# **Statistical Analysis**

## Normality Distribution Tests

The given statistical tests prove that Sales data is not normally distributed. We can try some type of transformations like log, exponential, inversions etc. But those conversions are not helping improve the accuracy in this case.

## Quantile-Quantile Plot

The Q-Q plot, or quantile-quantile plot, is a graphical tool to help us assess if a set of data possibly came from some theoretical distribution such as a Normal or exponential. Q-Q plots are used to find the type of distribution for a random variable whether it be a Gaussian Distribution, Uniform Distribution, Exponential Distribution or even Pareto Distribution, etc. We can tell the type of distribution using the power of the Q-Q plot just by looking at the plot.

## Shapiro-Wilk Test

The Shapiro-Wilk test evaluates a data sample and quantifies how likely it is that the data was drawn from a Gaussian distribution, named for Samuel Shapiro and Martin Wilk.

**Results:**

*H0 : Sample was drawn from a Gaussian distribution , Ha : Sample was not drawn from a Gaussian distribution*

*Statistics=0.9218, p-value=0.0000*

Conclusion:

Sample does not look Gaussian (reject Null Hypothesis H0)

## Anderson-Darling Test

Anderson-Darling Test is a statistical test that can be used to evaluate whether a data sample comes from one of among many known data samples, named for Theodore Anderson and Donald Darling.

**Results:**

*H0 : Sample was drawn from a Gaussian distribution , Ha : Sample was not drawn from a Gaussian distribution*

*Statistic: 1819.1218*

*Significance Level 15.0000: Critical Value 0.5760, Data does not look normal (reject Null Hypothesis H0)*

*Significance Level 10.0000: Critical Value 0.6560, Data does not look normal (reject Null Hypothesis H0)*

*Significance Level 5.0000: Critical Value 0.7870, Data does not look normal (reject Null Hypothesis H0)*

*Significance Level 2.5000: Critical Value 0.9180, Data does not look normal (reject Null Hypothesis H0)*

*Significance Level 1.0000: Critical Value 1.0920, Data does not look normal (reject Null Hypothesis H0)*

## Spearman Rank Correlation

Spearman rank correlation coefficient measures the monotonic relation between two variables. Its values range from -1 to +1 and can be interpreted as:

+1: Perfectly monotonically increasing relationship

+0.8: Strong monotonically increasing relationship

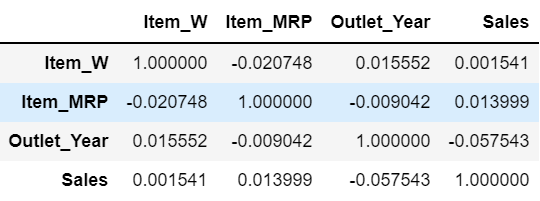
+0.2: Weak monotonically increasing relationship

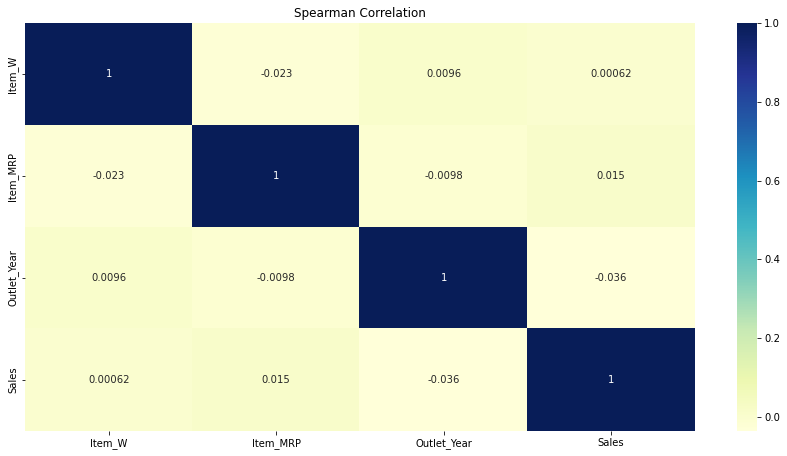
0: Non-monotonic relation

-0.2: Weak monotonically decreasing relationship

-0.8: Strong monotonically decreasing relationship

-1: Perfectly monotonically decreasing relationship





## Kruskal-Wallis H Test

### Relationship of Sales & Outlet Size

The Kruskal-Wallis test is a nonparametric version of the one-way analysis of variance test or ANOVA for short. A Kruskal-Wallis test is used to determine whether or not there is a statistically significant difference between the medians of three or more independent groups. It is considered to be the non-parametric equivalent of the One-Way ANOVA.

*Statistics for Kruskal-Wallis Test is 631.364, p=0.0000*

*Conclusion:*

*Different distributions: One or more sample distributions(small\_sales, medium\_sales & high\_sales) are not equal(reject H0)*

### Relationship of Sales & Outlet Location Type

*Statistics for Kruskal-Wallis Test is 452.281, p=0.0000*

*Conclusion:*

*Different distributions: One or more sample distributions(tier1\_sales, tier2\_sales, tier3\_sales) are not equal(reject H0)*

## Kruskal-Wallis Test Effect Size

The eta squared, based on the H-statistic, can be used as the measure of the Kruskal-Wallis test effect size.

It is calculated as follow : eta2[H] = (H - k + 1)/(n - k); where H is the value obtained in the Kruskal-Wallis test; k is the number of groups; n is the total number of observations (M. T. Tomczak and Tomczak 2014). The eta-squared estimate assumes values from 0 to 1 and multiplied by 100 indicates the percentage of variance in the dependent variable explained by the independent variable.

The interpretation values commonly in published literature are:

0.01- < 0.06 (small effect),

0.06 - < 0.14 (moderate effect)

>= 0.14 (large effect)

In this case we see that the test effect size is 0.005 and 0.007 for Outlet Location Type and Outlet Size. So it’s even below the ‘small effect’ range.

### Outlet Size:

Statistics for Kruskal-Wallis Test is 0.007

Conclusion:

Unexplained variance

### Outlet Location Type:

Statistics for Kruskal-Wallis Test is 0.005

Conclusion:

Unexplained variance

# **Handle -ve Sales figures**

We find there are sone -ve sales values which is not clear. Negative sales number might mean that these are losses , however this is not clearly defined in the problem statement. First, we try by dropping the -ve sales records but its leads to reduced RMSE values. Finally, we found that by turning these -ve numbers to positive , we can get a marginally better RMSE.

# **Outlier Detection**

## IQR method of outlier detection

Calculate the interquartile range for the data.

Multiply the interquartile range (IQR) by 1.5 (a constant used to discern outliers).

Add 1.5 x (IQR) to the third quartile. Any number greater than this is a suspected outlier. Subtract 1.5 x (IQR) from the first quartile. Any number less than this is a suspected outlier.

## Outlier Handling by Winsorization

Winsorization is a way to minimize the influence of outliers in your data by either: Assigning the outlier a lower weight Changing the value so that it is close to other values in the set. The data points are modified, not trimmed/removed.

# **Feature Engineering**

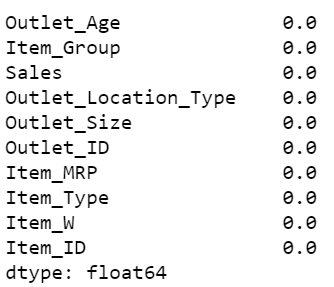
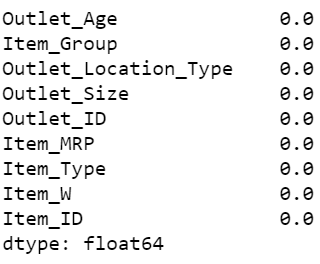
## Derive the Item\_Group based on Item\_Type

Here, we notice that Item-Type values can be grouped into some common categories of data like Drinks, Non Consumables and Food. Creating these new features help us better train the model in later stage.

## Derive the Outlet\_Age based on Outlet\_Year

In the given dataset, we have a feature called Outlet\_Year but this by itself is not going to be very useful. We know that the age of an outlet can have some impact on the sales, an older more well-known outlet might have more sales than a newer one.

## Check no NULL values in train/test

## Categorical to Numeric Conversion

Many machine learning algorithms cannot operate on label data directly. They require all input variables and output variables to be numeric. Since we will be applying a regression algorithm, all the features must be numeric in nature. We can do it by converting the existing categorical columns by applying:

### Label Encoding

Here, each unique category value is assigned an integer value. We convert the labels into a numeric form so as to convert them into the machine-readable form.

### One Hot Encoding

For categorical variables where no ordinal relationship exists, the label encoding is not enough. In fact, using this encoding and allowing the model to assume a natural ordering between categories may result in poor performance or unexpected results (predictions halfway between categories).

|  |  |
| --- | --- |
| **Encoding** | **Feature** |
| One-Hot Encode | Item\_Type |
| Label Encode | Outlet\_Size |
| Label Encode | Outlet\_Location\_Type |
| One-Hot Encode | Outlet\_ID |
| One-Hot Encode | Item\_Group |
| One-Hot Encode | Item\_ID |

# **Machine Learning Modelling**

We will implement some well-known ML models and check their performance. The RMSE values are compared against those in the public leaderboard. Lastly, we will try out a stacked ensemble model using H2O library to give us the best performance metric.

## Multivariate Linear Regression

**Train**

Root Mean Squared Error: 1273.09 , R2 Score: 0.0281

**Test**

Root Mean Squared Error: 1285.84 , R2 Score: -0.0013

## Random Forest Regressor

**Train**

Root Mean Squared Error: 1040.87 , R2 Score: 0.3503

**Test**

Root Mean Squared Error: 1285.84 , R2 Score: -0.0013

## Deep Neural Net Regressor

**Train**

Root Mean Squared Error: 1273.85 , R2 Score: N/A

**Test**

Root Mean Squared Error: 1283.41 ,R2 Score: N/A

## XGBoost Regressor

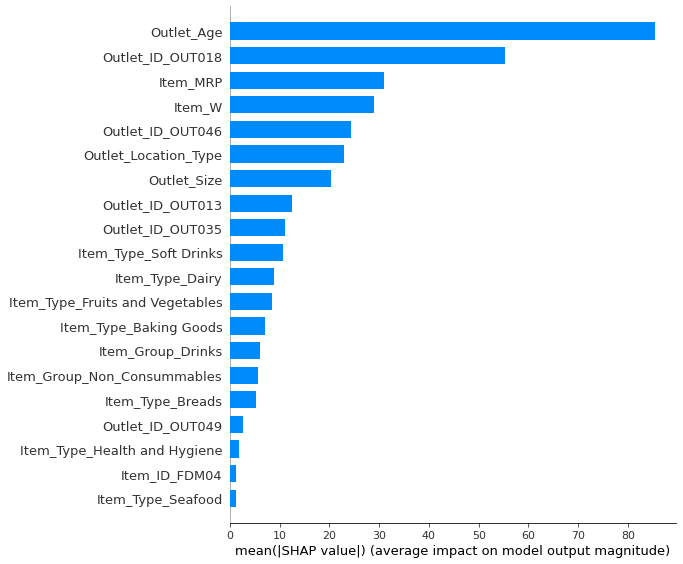
**Train**

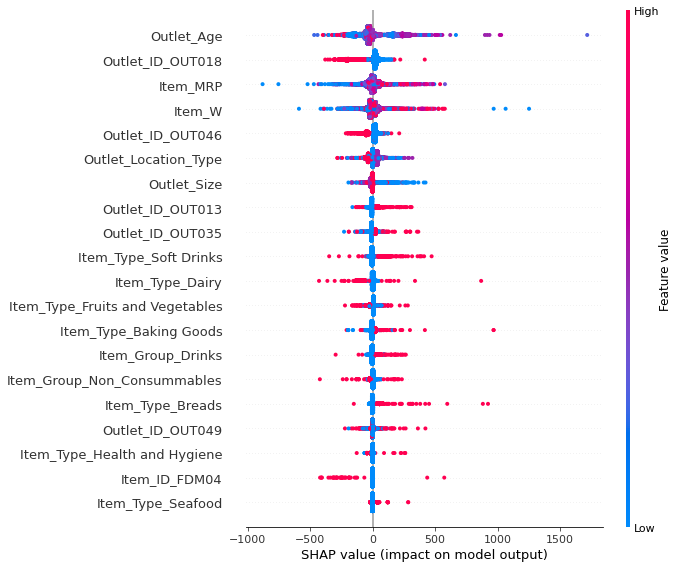
Root Mean Squared Error: 1217.39 , R2 Score: 0.1129

**Test**

Root Mean Squared Error: 1287.29 ,R2 Score: -0.0013

### Feature Importance – XGBoost





## Hyptertuned XGBoost Regressor

Of all the base models we tried XGBoost gave us the best scores. So we have tried XGBoost algorithm by tuning certain hyper-parameters:

***nthread*** *[default to maximum number of threads available if not set] : This is used for parallel processing and number of cores in the system should be entered.*

***objective*** *[default=reg:linear] : This defines the loss function to be minimized. It can have values like 'reg:squarederror', 'binary:logistic', 'multi:softmax' and 'multi:softprob'.*

***learning rate****: The learning rate is the shrinkage you do at every step you are making. If my learning rate is 0.10, you will either land on either 52 or 53 computation steps.*

***max\_depth*** *: This should be between 3-10. I’ve started with 5 but we can choose a different number as well.*

***min\_child\_weight*** *: A smaller value is chosen because it is not a imbalanced class problem and leaf nodes can have smaller size groups.*

***subsample, colsample\_bytree*** *= 1 : This is a commonly used used start value. Typical values range between 0.5-1.*

***n\_estimators*** *: Number of gradient boosted trees, here we have tried with 100.*

**Train**

Root Mean Squared Error: 1458.79 , R2 Score: -0.2737

**Test**

Root Mean Squared Error: 1287.29, R2 Score: -0.0013

## Best Model: Stacked Ensemble Learning

Ensemble machine learning methods use multiple learning algorithms to obtain better predictive performance than could be obtained from any of the constituent learning algorithms. H2O’s Stacked Ensemble method is a supervised ensemble machine learning algorithm that finds the optimal combination of a collection of prediction algorithms using a process called stacking. Like all supervised models in H2O, Stacked Ensemble supports regression, binary classification, and multiclass classification.

### Model Summary

*AutoML progress: |*

*06:43:51.383: AutoML: XGBoost is not available; skipping it.*

*06:43:51.383: Step 'best\_of\_family\_xgboost' not defined in provider 'StackedEnsemble': skipping it.*

*06:43:51.383: Step 'all\_xgboost' not defined in provider 'StackedEnsemble': skipping it.*

*███████████████████████████████████████████████████████████████| 100%e) 100%*

*Model Details*

*=============*

*H2OStackedEnsembleEstimator : Stacked Ensemble*

*Model Key: StackedEnsemble\_Best1000\_1\_AutoML\_4\_20211107\_64351*

*No model summary for this model*

*ModelMetricsRegressionGLM: stackedensemble*

***\*\* Reported on train data. \*\****

*MSE: 1616513.1736752875*

*RMSE: 1271.4217135456227*

*MAE: 997.614318280515*

*RMSLE: 0.8488591943425776*

*R^2: 0.052036319180277624*

*Mean Residual Deviance: 1616513.1736752875*

*Null degrees of freedom: 9936*

*Residual degrees of freedom: 9913*

*Null deviance: 16947730287.698263*

*Residual deviance: 16063291406.811333*

*AIC: 170307.17039749262*

*ModelMetricsRegressionGLM: stackedensemble*

***\*\* Reported on cross-validation data. \*\****

*MSE: 1620921.740375437*

*RMSE: 1273.154248461449*

*MAE: 999.1295029658132*

*RMSLE: 0.8492642990325258*

*R^2: 0.026072930599253752*

*Mean Residual Deviance: 1620921.740375437*

*Null degrees of freedom: 87863*

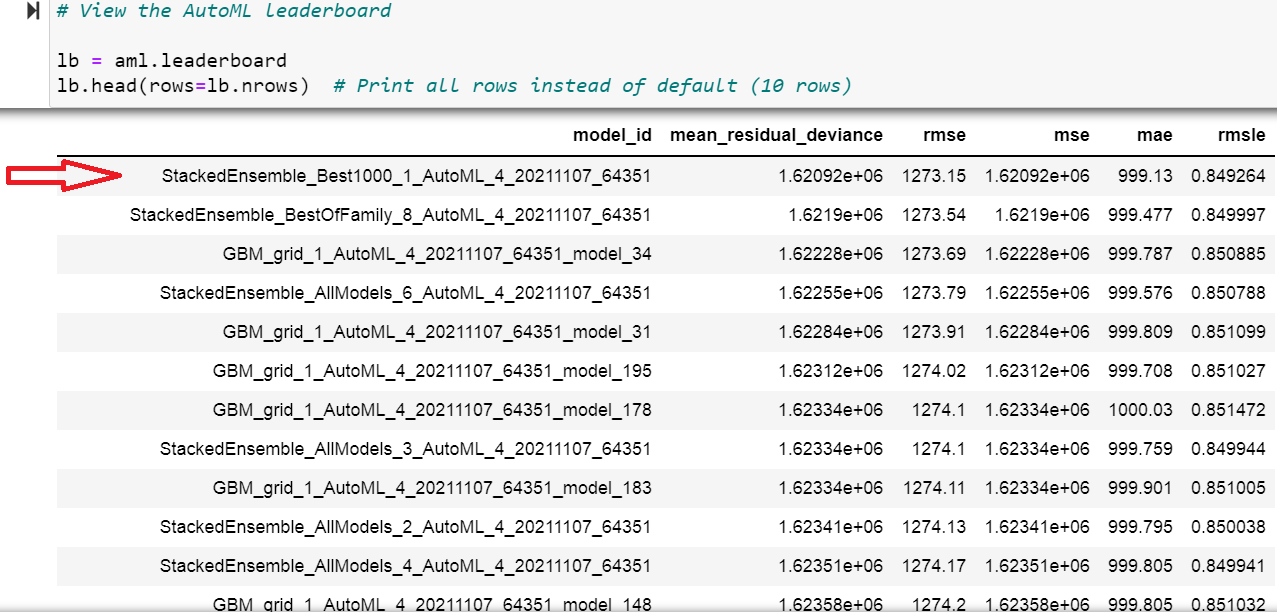
*Residual degrees of freedom: 87845*

*Null deviance: 146240242761.22968*

*Residual deviance: 142420667796.3474*

*AIC: 1505711.1196551926*

### AutoML Model Leaderboard



### Ensemble Exploration

To understand how the ensemble works, let's take a peek inside the Stacked Ensemble "All Models" model. The "All Models" ensemble is an ensemble of all of the individual models in the AutoML run. This is often the top performing model on the leaderboard

Examine the variable importance of the metalearner (combiner) algorithm in the ensemble. This shows us how much each base learner is contributing to the ensemble. The AutoML Stacked Ensembles use the default metalearner algorithm (GLM with non-negative weights), so the variable importance of the metalearner is actually the standardized coefficient magnitudes of the GLM.

*{'Intercept': 1912.7163409313366,*

*'GBM\_lr\_annealing\_selection\_AutoML\_4\_20211107\_64351\_select\_model': 18.640456226574376,*

*'GBM\_2\_AutoML\_4\_20211107\_64351': 11.529158207654246,*

*'GBM\_1\_AutoML\_4\_20211107\_64351': 59.69773448020982,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_12': 13.883106092709891,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_2': 11.269978696041612,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_6': 5.507848738480781,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_3': 8.181268430450519,*

*'GBM\_5\_AutoML\_4\_20211107\_64351': 24.529119631061555,*

*'GBM\_4\_AutoML\_4\_20211107\_64351': 8.01995060233593,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_9': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_14': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_7': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_4': 20.89921292767838,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_10': 10.352665311715477,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_15': 11.438192194746453,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_11': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_1': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_8': 0.0,*

*'DRF\_1\_AutoML\_4\_20211107\_64351': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_5': 0.0,*

*'GBM\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_13': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_14': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_14': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_5': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_14': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_5': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_12': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_6': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_15': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_2': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_16': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_20': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_17': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_2': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_13': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_11': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_15': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_17': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_13': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_6': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_3': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_13': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_12': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_9': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_12': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_18': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_7': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_19': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_1': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_6': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_18': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_15': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_11': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_9': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_5': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_3': 0.0,*

*'DeepLearning\_1\_AutoML\_4\_20211107\_64351': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_1': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_8': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_16': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_9': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_1': 0.0,*

*'GLM\_1\_AutoML\_4\_20211107\_64351': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_3': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_4': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_11': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_4': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_10': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_2': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_7': 0.0,*

*'DeepLearning\_grid\_1\_AutoML\_4\_20211107\_64351\_model\_16': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_10': 0.0,*

*'DeepLearning\_grid\_3\_AutoML\_4\_20211107\_64351\_model\_8': 0.0,*

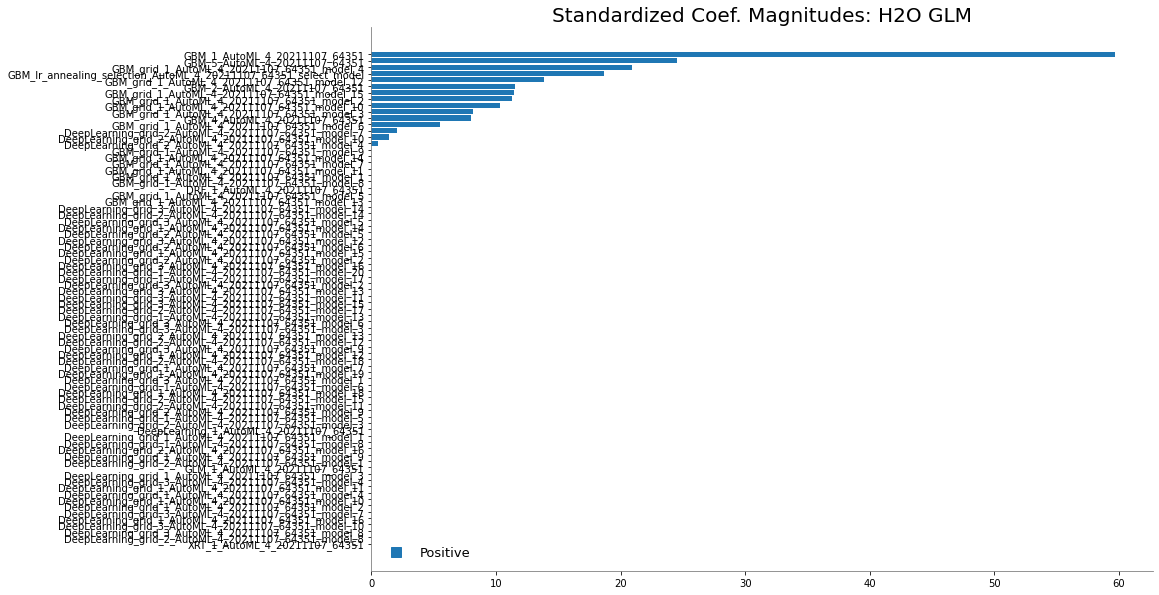
*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_10': 1.391330659698167,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_8': 0.0,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_7': 2.0985756168623912,*

*'DeepLearning\_grid\_2\_AutoML\_4\_20211107\_64351\_model\_4': 0.5489182047643926,*

*'XRT\_1\_AutoML\_4\_20211107\_64351': 0.0}*



# **Conclusion**

In this hackathon, I have applied various methods to cleanse and preprocess the dataset. The methods have also been applied on the test dataset in parallel. Data visualization techniques have shown some key insights into the data. Thereafter, by applying statistical hypothesis tests , we can conclude the relationships between sales and categorical and continuous fields. Finally, we have applied some popular ML algorithms on the data to get best value of RMSE. The models have given us decent performance, but the best result is from the stacked ensemble learning method.