

LP1 Assignment DA R3

Bigmart Sales Analysis

Date - 10th September, 2020.

Assignment Number - DA R3

Title

Bigmart Sales Analysis

Problem Definition

For data consisting of transaction records of a sales store. The data has 8523 rows of 12 variables. Predict the sales of a store.

Learning Objectives

- Learn Regression algorithms
- Learn to summarize the properties in the training dataset.
- Learn to split the dataset into training and test datasets.
- Learn to develop a predictive regression model

Learning Outcomes

I will be able to develop a predictive model for sales of an item at BigMart.

Software Packages and Hardware Apparatus Used

- Operating System : 64-bit Ubuntu 18.04
- Programming Language : Python 3
- Jupyter Notebook Environment : Google Colaboratory
- Python Libraries : Kaggle, Kaggle CLI, Sklearn, Pandas, Matplotlib, Graphviz, PyCaret

Programmer's Perspective

Let S be the system set:

$S = \{s; e; X; Y; Fme; DD; NDD; Fc; Sc\}$

where Dataset is loaded into the dataframe

s=start state

e=end state

- predicted sales

X=set of inputs

- $X = \{X1\}$
 - where X1 = BigMart Sales Dataset (8523 records, 12 columns)

Y=set of outputs

- $Y = \{Y1, Y2\}$
 - Y1 = Predicted Values
 - Y2 = Accuracy Score (Metric - RMSE)

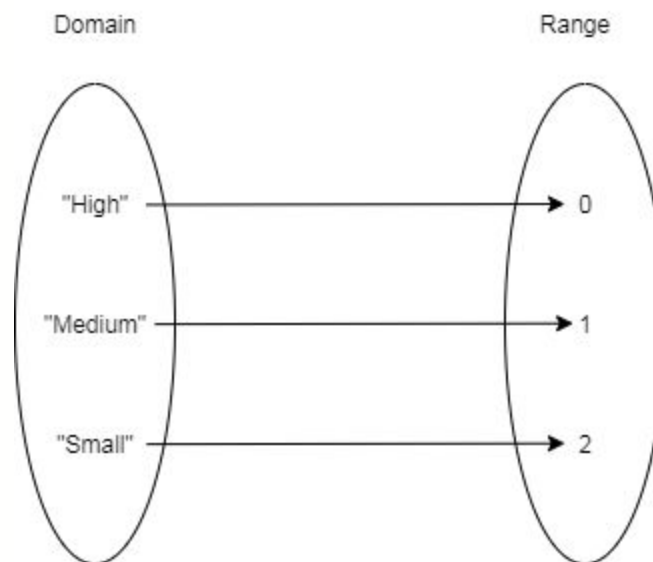
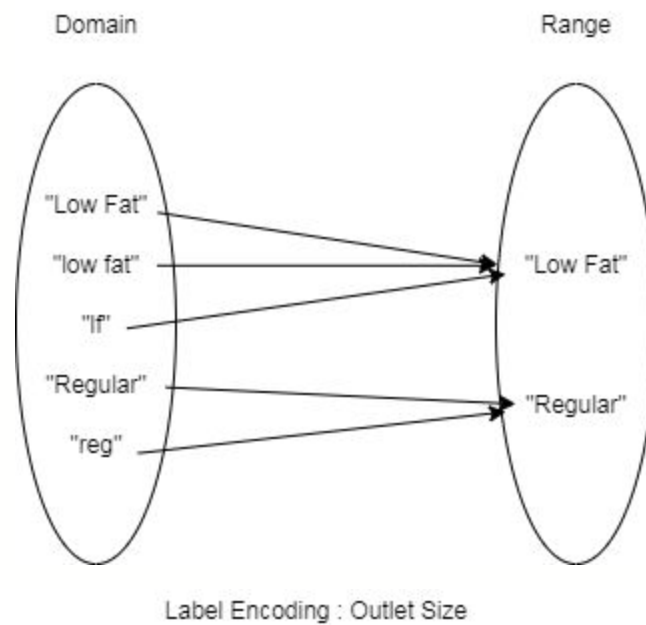
Fme is the set of main functions

- $Fe = \{f0\}$
 - f0 = Main Display Function

Ff is the set of friend functions

- $Ff = \{f1, f2, f3, f4, f5, f6\}$ where
 - f1 = function to load dataset into dataframe
 - f2 = function to handle null values
 - f3 = function to handle redundant values
 - f4 = function to generate label encoding of string values
 - f5 = function to split dataset into test and train data
 - f6 = function to train the model

Function to handle Redundant Values



DD = Deterministic Data

- BigMart Sales Dataset

NDD = Non-deterministic data (Eg - Null Values in Dataset)

- 1463 null values in the the feature Item_Weight
- 2410 null values in the the feature Outlet_Size

Fc = failure case

- High Value of RMSE
- Low Value of R2

Concepts related Theory

Linear Regression

In statistics, linear regression is a linear approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple linear regression. For more than one explanatory variable, the process is called multiple linear regression.

The relationships are modeled using linear predictor functions whose unknown model parameters are estimated from the data. Such models are called linear models.

If the goal is prediction, forecasting, or error reduction, linear regression can be used to fit a predictive model to an observed data set of values of the response and explanatory variables. After developing such a model, if additional values of the explanatory variables are collected without an accompanying response value, the fitted model can be used to make a prediction of the response.

Given a dataset of n statistical units, a linear regression model assumes that the relationship between the dependent variable y and the p -vector of regressors x is linear. This relationship is modeled through a disturbance term or error variable ε — an unobserved random variable that adds "noise" to the linear relationship between the dependent variable and regressors.

Dataset = $\{y_i, x_{i1}, \dots, x_{ip}\}_{i=1}^n$

Model Equation :

$$y_i = \beta_0 + \beta_1 x_{i1} + \dots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^T \boldsymbol{\beta} + \varepsilon_i, \quad i = 1, \dots, n,$$

Matrix Notation : $\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}$,

$$\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix},$$

$$X = \begin{pmatrix} \mathbf{x}_1^\top \\ \mathbf{x}_2^\top \\ \vdots \\ \mathbf{x}_n^\top \end{pmatrix} = \begin{pmatrix} 1 & x_{11} & \cdots & x_{1p} \\ 1 & x_{21} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_{n1} & \cdots & x_{np} \end{pmatrix},$$

$$\boldsymbol{\beta} = \begin{pmatrix} \beta_0 \\ \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}.$$

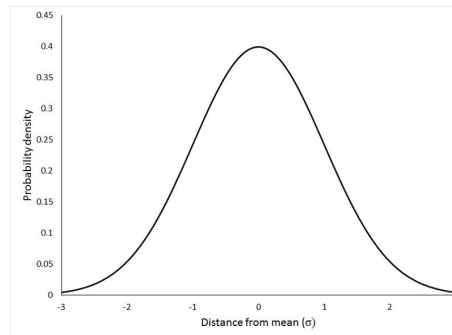
Dataset Description

The data scientists at BigMart have collected 2013 sales data for 1559 products across 10 stores in different cities. Also, certain attributes of each product and store have been defined. The aim is to build a predictive model and find out the sales of each product at a particular store.

- **Item_Identifier:** Unique product ID
- **Item_Weight:** Weight of product
- **Item_Fat_Content:** Whether the product is low fat or not
- **Item_Visibility:** The % of total display area of all products in a store allocated to the particular product
- **Item_Type:** The category to which the product belongs
- **Item_MRP:** Maximum Retail Price (list price) of the product
- **Outlet_Identifier:** Unique store ID
- **Outlet_Establishment_Year:** The year in which store was established
- **Outlet_Size:** The size of the store in terms of ground area covered
- **Outlet_Location_Type:** The type of city in which the store is located
- **Outlet_Type:** Whether the outlet is just a grocery store or some sort of supermarket
- **Item_Outlet_Sales:** Sales of the product in the particular store. This is the outcome variable to be predicted.

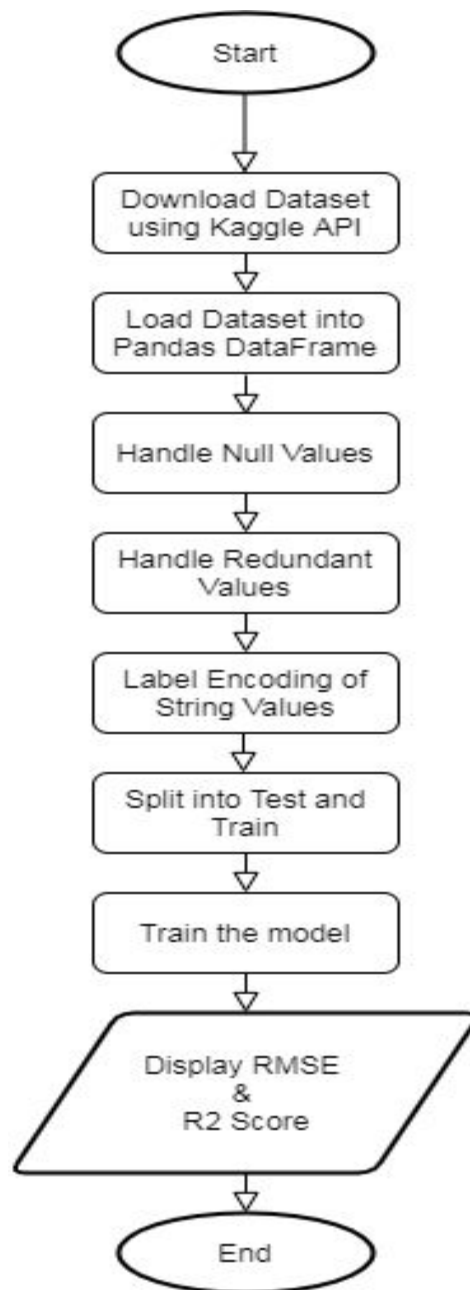
Gaussian Distribution

It is a symmetric distribution where most of the observations cluster around the central peak and the probabilities for values further away from the mean taper off equally in both directions.



To deal with missing values in a numerical feature with gaussian distribution, we can calculate the mean of the feature and replace it with the missing values. This is an approximation which can add variance to the data set. But the loss of the data can be negated by this method which yields better results compared to removal of rows and columns. Replacing with mean, median or mode is a statistical approach of handling the missing values. This method is also called leaking the data while training.

Flowchart



Output Screenshots

Regression Models - RMSE and R2 Score

The first screenshot shows the results for several linear and ensemble models. The second screenshot shows the results for a neural network, a decision tree, and a graphviz visualization of the decision tree.

```
[ ] from sklearn.linear_model import LinearRegression
RegressionAlgorithm(LinearRegression)()
Root Mean squared error: 1152.13
R2 score: 0.51

[ ] from sklearn.linear_model import Ridge
RegressionAlgorithm(Ridge)()
Root Mean squared error: 1152.23
R2 score: 0.51

[ ] from sklearn.linear_model import Lasso
RegressionAlgorithm(Lasso)()
Root Mean squared error: 1152.89
R2 score: 0.51

[ ] from sklearn.linear_model import ElasticNet
RegressionAlgorithm(ElasticNet)()
Root Mean squared error: 1183.13
R2 score: 0.48

[ ] from sklearn.ensemble import RandomForestRegressor
RegressionAlgorithm(RandomForestRegressor, n_estimators=100, max_depth=6, min_samples_leaf=50, n_jobs=4)()
Root Mean squared error: 1044.38
R2 score: 0.59

[ ] from sklearn.svm import LinearSVR
RegressionAlgorithm(LinearSVR, max_iter=10000)()
Root Mean squared error: 1181.76
R2 score: 0.48
/usr/local/lib/python3.6/dist-packages/sklearn/svm/_base.py:947: ConvergenceWarning: Liblinear failed to converge, increase the number of iterations.
"the number of iterations.", ConvergenceWarning)
```

```
[ ] from sklearn.neural_network import MLPRegressor
RegressionAlgorithm(MLPRegressor, max_iter=500)()
Root Mean squared error: 1149.64
R2 score: 0.51

[ ] from sklearn.tree import DecisionTreeRegressor
RegressionAlgorithm(DecisionTreeRegressor, max_depth=15, min_samples_leaf=300)()
Root Mean squared error: 1076.45
R2 score: 0.57

!pip install graphviz
import graphviz
from sklearn.tree import DecisionTreeRegressor, export_graphviz

model = DecisionTreeRegressor(max_depth=15, min_samples_leaf=300)
model.fit(X_train, y_train)

y_pred = model.predict(X_test)

rmse = math.sqrt(mean_squared_error(y_test, y_pred))

print("Root Mean squared error: %.2f" % rmse)
print("R2 score: %.2f" % r2_score(y_test, y_pred))

dot_data = export_graphviz(model, out_file=None, filled=True, rounded=True, special_characters=True)
graph = graphviz.Source(dot_data)
graph.render("bigmart_decision_tree")

Requirement already satisfied: graphviz in /usr/local/lib/python3.6/dist-packages (0.10.1)
Root Mean squared error: 1144.83
R2 score: 0.56
"bigmart_decision_tree.pdf"
```

Modification

```
df.info()

<class 'pandas.core.frame.DataFrame'>
```


The image displays a decision tree visualization for a file named 'bigmart_decision_tree.pdf'. The tree is a hierarchical structure of nodes, each containing a split condition (e.g., $X_5 \leq 157.079$), the mean squared error (mse), the number of samples, and the value. The root node splits on $X_5 \leq 157.079$. The left branch (True) leads to a node with $X_{10} \leq 0.5$, which further splits into $X_5 \leq 98.771$ and $X_{10} \leq 0.5$. The right branch (False) leads to a node with $X_{10} \leq 0.5$, which further splits into $X_5 \leq 206.914$ and $X_5 \leq 235.976$. The tree continues to split down to leaf nodes, with the final mse and value for each branch. The interface includes a file explorer at the top showing the file path 'C:\Users\Shikhar\Downloads\bigmart_decision_tree.pdf' and a taskbar at the bottom with various application icons and the system clock showing 8:48 PM on 9/11/2020.

```

graph TD
    Root["X5 ≤ 157.079  
mse = 2902969.377  
samples = 6818  
value = 2180.846"]
    Root -- True --> Node1["X10 ≤ 0.5  
mse = 1187134.843  
samples = 4031  
value = 1499.695"]
    Root -- False --> Node2["X10 ≤ 0.5  
mse = 3743023.887  
samples = 2787  
value = 3166.034"]
    Node1 -- True --> Node3["X5 ≤ 62.718  
mse = 464849.789  
samples = 1645  
value = 1135.198"]
    Node1 -- False --> Node4["X7 ≤ 1992.0  
mse = 1116929.373  
samples = 1859  
value = 2181.688"]
    Node2 -- True --> Node5["X5 ≤ 143.497  
mse = 891185.042  
samples = 1403  
value = 2022.638"]
    Node2 -- False --> Node6["X10 ≤ 2.5  
mse = 3072362.443  
samples = 2427  
value = 3561.915"]
    Node3 -- True --> Node7["X5 ≤ 51.517  
mse = 193695.321  
samples = 846  
value = 815.638"]
    Node3 -- False --> Node8["X7 ≤ 1998.0  
mse = 529342.697  
samples = 799  
value = 1473.556"]
    Node4 -- True --> Node9["X5 ≤ 1494184.99  
samples = 456  
value = 2671.047"]
    Node4 -- False --> Node10["X5 ≤ 119.579  
mse = 736096.456  
samples = 1030  
value = 1877.634"]
    Node6 -- True --> Node11["X5 ≤ 180.765  
mse = 1644501.653  
samples = 1244  
value = 2899.213"]
    Node6 -- False --> Node12["mse = 4105067.415  
samples = 305  
value = 3315.507"]
    Node7 -- True --> Node13["mse = 110160.093  
samples = 546  
value = 712.099"]
    Node7 -- False --> Node14["mse = 290708.075  
samples = 300  
value = 1004.08"]
    Node8 -- True --> Node15["mse = 709679.932  
samples = 319  
value = 1681.537"]
    Node8 -- False --> Node16["mse = 361641.259  
samples = 480  
value = 1335.335"]
    Node10 -- True --> Node17["mse = 590325.176  
samples = 965  
value = 1731.687"]
    Node10 -- False --> Node18["mse = 655897.695  
samples = 465  
value = 2054.968"]
    Node11 -- True --> Node19["mse = 1275673.253  
samples = 327  
value = 2594.244"]
    Node11 -- False --> Node20["mse = 1571152.2  
samples = 345  
value = 2855.39"]
    Node12 -- True --> Node21["mse = 1804954.345  
samples = 572  
value = 3099.989"]
    Node12 -- False --> Node22["mse = 261619.499  
samples = 414  
value = 3649.251"]
    Node16 -- True --> Node23["mse = 3184827.586  
samples = 464  
value = 4133.826"]
    Node16 -- False --> Node24["mse = 2975556.786  
samples = 305  
value = 3905.356"]

```

BE1_LP1_P1_41106_DA_R3_Apoov_Dixit.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Files

BE1_LP1_P1_41106_DA_R3_Apoov_Dixit.ipynb

Train.csv

bigmart_decision_tree

bigmart_decision_tree.pdf

+ Code + Text

```

[ ] !pip install pycaret

[ ] from pycaret.regression import setup, compare_models
    experiment = setup(df, target = 'Item_Outlet_Sales')

[ ] compare_models()

```

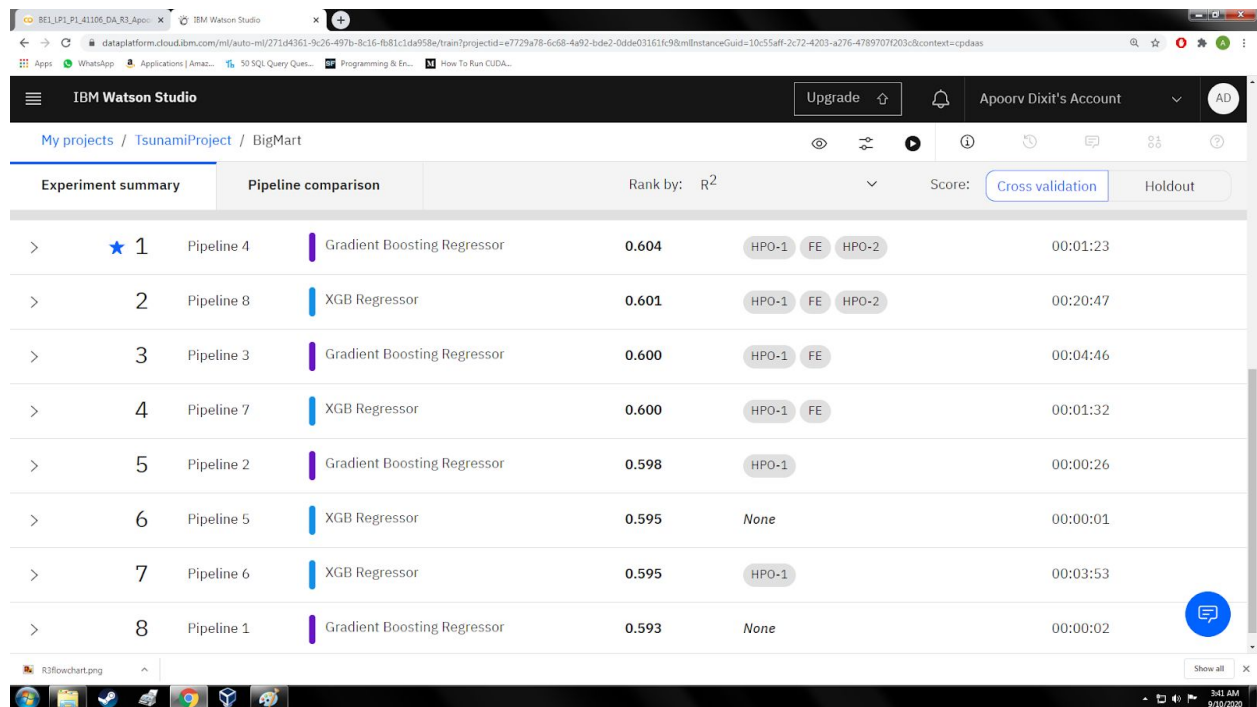
D.	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
0	Extreme Gradient Boosting	788.8962	11768.9612	108.5272	0.8970	0.5568	0.9622	0.0083
1	Gradient Boosting Regressor	755.5219	1175073.3417	1054.9573	0.5951	0.5602	0.5645	1.1598
2	CalBoost Regressor	787.7972	1261659.4898	1122.2421	0.5677	0.5905	0.5840	3.0076
3	Light Gradient Boosting Machine	783.8669	1271670.2613	1126.2969	0.5647	0.5623	0.5723	0.1312
4	Orthogonal Matching Pursuit	838.0810	1287015.3442	1133.3814	0.5598	0.7229	1.0500	0.0108
5	Lasso Regression	839.3370	1289719.4533	1134.5091	0.5590	0.7249	1.0514	0.0608
6	Bayesian Ridge	839.8183	1290773.0605	1134.9566	0.5586	0.7284	1.0491	0.0249
7	Ridge Regression	840.3434	1291197.2407	1135.1476	0.5585	0.7290	1.0544	0.0085
8	ThetaSVM Regressor	839.0231	1291101.0897	1135.0905	0.5585	0.7372	1.0338	7.5853
9	Linear Regression	840.4148	1291267.4332	1135.1787	0.5584	0.7294	1.0549	0.0165
10	Random Sample Consensus	835.8849	1307236.6531	1142.0674	0.5532	0.7174	0.9969	0.9241
11	Huber Regressor	832.9486	1307738.3874	1142.2872	0.5530	0.7096	0.9757	0.2466
12	Lasso Least Angle Regression	841.9526	1307418.7669	1142.2030	0.5530	0.7131	0.9713	0.0104
13	Random Forest	800.1936	1314356.1436	1146.0370	0.5496	0.8626	0.5636	3.2670
14	Extra Trees Regressor	824.8649	1420730.2227	1190.9583	0.5129	0.8687	0.8833	2.6430
15	AdaBoost Regressor	956.9302	1497583.4400	1222.1859	0.4872	0.8070	1.3554	0.4031
16	Elastic Net	916.8870	1588295.2815	1258.7116	0.4573	0.7638	1.0252	0.0117
17	K Neighbors Regressor	1152.0243	2399777.8306	1547.5214	0.1792	0.9328	1.4997	0.0329
18	Decision Tree	1092.1798	2470963.3563	1570.8629	0.1512	0.7470	0.7168	0.0739
19	Support Vector Machine	1306.9096	3030991.9535	1742.0532	-0.0398	1.0633	1.7920	4.0364
20	Passive Aggressive Regressor	1679.0289	6543905.0497	2117.7047	-1.2154	1.3482	2.1415	0.0762
21	Least Absolute Regressor	78627119.3371	140776447975461440.0000	119758972.9549	-42361540055.5350	2.8816	0.8054	2914.0265

```

XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
              colsample_bynode=1, colsample_bytree=1, gamma=0,
              importance_type='gain', learning_rate=0.1, max_delta_step=0,
              max_depth=3, min_child_weight=1, missing=None, n_estimators=100,
              n_jobs=1, nthread=None, objective='reg:linear', random_state=8127,
              reg_alpha=0, reg_lambda=1, scale_pos_weight=1, seed=None,
              silent=None, subsample=1, verbosity=0)

```

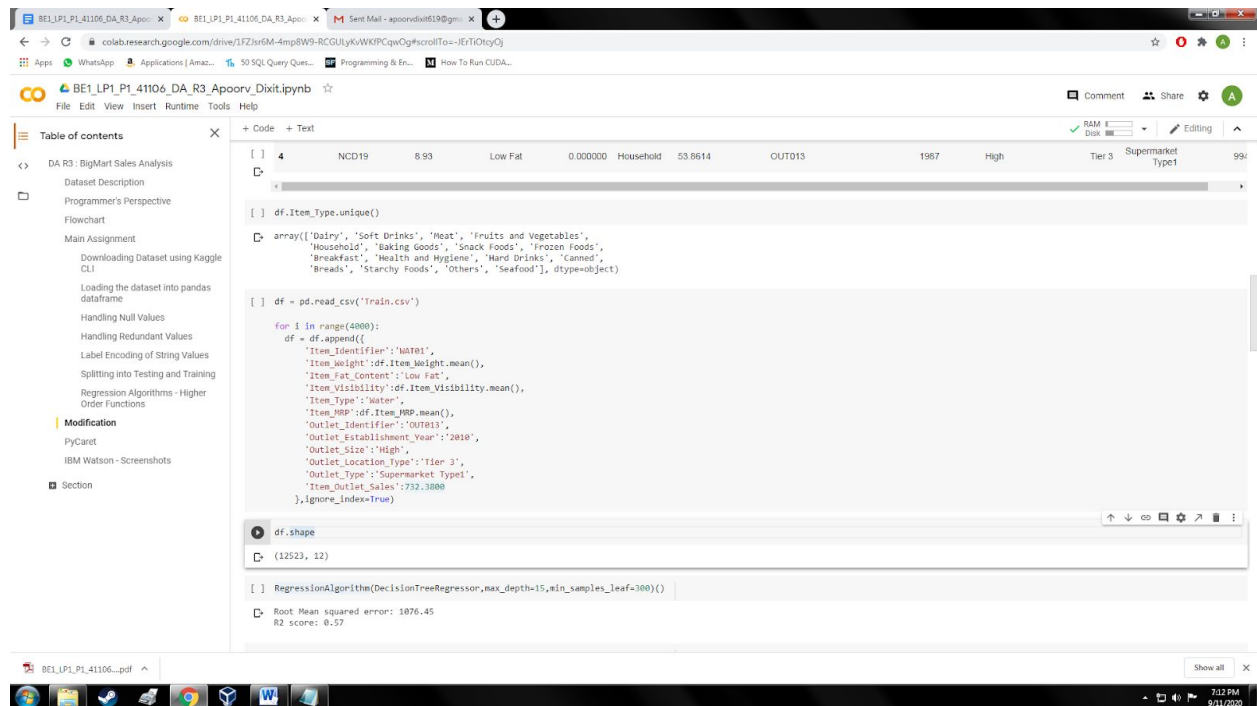
IBM Watson Machine Learning Auto AI Experiment



The screenshot shows the IBM Watson Studio interface. At the top, there's a navigation bar with 'My projects / TsunamiProject / BigMart'. Below this is a table titled 'Experiment summary' with columns for Rank, Pipeline, Model, Score, and Time. The table lists 8 experiments. Experiment 1 is highlighted with a blue star. The 'Rank by' dropdown is set to R^2 . There are buttons for 'Cross validation' and 'Holdout'.

Rank	Pipeline	Model	Score	Time
1	Pipeline 4	Gradient Boosting Regressor	0.604	00:01:23
2	Pipeline 8	XGB Regressor	0.601	00:20:47
3	Pipeline 3	Gradient Boosting Regressor	0.600	00:04:46
4	Pipeline 7	XGB Regressor	0.600	00:01:32
5	Pipeline 2	Gradient Boosting Regressor	0.598	00:00:26
6	Pipeline 5	XGB Regressor	0.595	00:00:01
7	Pipeline 6	XGB Regressor	0.595	00:03:53
8	Pipeline 1	Gradient Boosting Regressor	0.593	00:00:02

Adding Item "Water"



The screenshot shows the IBM Watson Studio code editor. The left sidebar contains a 'Table of contents' with sections like 'DA R3: BigMart Sales Analysis', 'Dataset Description', 'Programmer's Perspective', 'Flowchart', 'Main Assignment', 'Downloading Dataset using Kaggle CLI', 'Loading the dataset into pandas dataframe', 'Handling Null Values', 'Handling Redundant Values', 'Label Encoding of String Values', 'Splitting into Testing and Training', 'Regression Algorithms - Higher Order Functions', 'Modification', 'PyCaret', 'IBM Watson - Screenshots', and 'Section'. The main code area shows a Jupyter Notebook cell with the following code:

```
[ ] 4 NCD19      8.93      Low Fat      0.000000      Household      53.8614      OUT013      1967      High      Tier 3      Supermarket Type1      994
```

```
[ ] df.item_type.unique()
```

```
[ ] array(['Dairy', 'Soft Drinks', 'Meat', 'Fruits and Vegetables', 'Household', 'Baking Goods', 'Snack Foods', 'Frozen Foods', 'Breakfast', 'Health and Hygiene', 'Hard Drinks', 'Canned', 'Breads', 'Starchy Foods', 'Others', 'Seafood'], dtype=object)
```

```
[ ] df = pd.read_csv('train.csv')
```

```
for i in range(4000):
    df = df.append({
        'Item_Identifier': 'WAT01',
        'Item_Weight': df.Item_Weight.mean(),
        'Item_Fat_Content': 'Low Fat',
        'Item_Visibility': df.Item_Visibility.mean(),
        'Item_Type': 'Water',
        'Item_MRP': df.Item_MRP.mean(),
        'Outlet_Identifier': 'OUT013',
        'Outlet_Establishment_Year': 2018,
        'Outlet_Size': 'High',
        'Outlet_Location_Type': 'Tier 3',
        'Outlet_Type': 'Supermarket Type1',
        'Item_Outlet_Sales': 732.3808
    }, ignore_index=True)
```

```
[ ] df.shape
```

```
[ ] (12523, 12)
```

```
[ ] RegressionAlgorithm(DecisionTreeRegressor, max_depth=15, min_samples_leaf=300())
```

```
[ ] Root Mean Squared error: 1076.45
```

```
[ ] R2 score: 0.57
```

Conclusion

I have successfully developed a predictive model for sales of an item at BigMart.