Date: 28-04-2025  **Modelling Approach: ABB Interview Task** . -Rajiv Dey

1. A Segmented approach based on Outlet\_Type was taken to get to a rank of 261 on the Hackathon Task. 4 separate models were run for each of the Outlet\_Type. As can be seen from the first visual the Distribution of Sales across Grocery Store, & Supermarket Types 1,2 & 3, the Sales numbers are different in varied proportions which gave me the inclination to have separate models for each to train the parameters.
2. EDA for Categorical Variables were across all Outlet Types to see how the variables are placed with respect to Sales. Observations:
   1. Order of Sales across Location Type: Tier3> tier2> Tier1 & Box plots show long tails
   2. Order of Sales across Outlet Size: Medium> Small> large & Box plots show long tails
   3. Order of Sales across Item Type: Dairy, Vegetables, Household, Snacks, Frozen Food and Fruit/Vegetables are the highest Contributors to Sales, whereas canned foods, seafood breads and breakfast make the lowest contributors.
   4. Order of Sales across Fat Type: Low fat makes up highest contributors
3. Feature Engineering:
   1. Fat Content has ‘Low Fat’, ‘LF’ & ‘low fat’ labels. Change them all to ‘Low Fat’
   2. Impute mean weight in NA based on mean weight basis each Item Identifier
   3. Impute Outlet Size basis mode of Outlet Type and Location in NA values
   4. Replace item Visibility of zeros with Median of column
   5. Instead of using year, used Age as Transformed Ordinal Variable to get Age
   6. Use a variable of Item Type combine to get similar items together based on first 4 words.
   7. Visibility Ratio is created by calculating the ratio of the Item’s visibility in particular outlet divide by mean of the Item’s visibility across Outlets.
4. EDA for each Outlet Type Supermarket 2 (Code implementations for the other 3 have not been done):
   1. Weight has a Uniform Distribution
   2. Weight to Output is random
   3. Item Visibility and Sales are right Skewed (Model>Median>mean)
   4. Item Visibility to Sales shows a positive Correlation i.e more Visibility more sales
   5. Item MRP to Sales shows a Strong positive Correlation i.e more Visibility more sales
   6. Item Visibilitility Ratio is left skewed.
   7. All Selling outputs have good Visibility Ration from the graph
5. Modelling Process:
   1. Tests were done using Lasso, Ridge, RandomForest Regressor and XgBoost regressor models as mentioned in the code outputs
   2. However, my initial iteration was done on Entire dataset using One Model (unlike the method chosen for final implementation) using a GridSearch Hyperparameter tuning on Random Forest Regressor (Code shown in Appendix) and prior to that I had done feature selection basis Shapley Value for Feature importance. But RMSE was coming pretty high at 1036. The Rank was coming at 2000+
   3. In the process implemented I kept the feature selection to a Correlation approach, however most had weak positive or negative correlations with the y-variable.
   4. The RMSE numbers significantly improved with the Segment Approach as RMSE for all models were in the range of 600-700. (30-40% improvement on RMSE basis previous approach)

APPENDIX:

1. My code outputs might seem repetitive as I did not have time to restructure it as per code hygiene.
2. I am attaching the grid search used in my failed iteration here

# --- Hyperparameter Tuning (Example with GridSearchCV for RandomForest) ---

from sklearn.model\_selection import GridSearchCV

param\_grid\_rf = {

'n\_estimators': [100, 200, 300],

'max\_depth': [10, 15, 20],

'min\_samples\_split': [2, 5, 10],

'min\_samples\_leaf': [1, 3, 5]

}

grid\_search\_rf = GridSearchCV(estimator=RandomForestRegressor(random\_state=42, n\_jobs=-1),

param\_grid=param\_grid\_rf,

cv=3,

verbose = 1,

scoring='neg\_root\_mean\_squared\_error',

n\_jobs=-1)

grid\_search\_rf.fit(X\_train, y\_train)

print("Best Hyperparameters for Random Forest:", grid\_search\_rf.best\_params\_)

best\_rf\_model = grid\_search\_rf.best\_estimator\_

rf\_pred\_val\_tuned = best\_rf\_model.predict(X\_val)

rmse\_rf\_tuned = mean\_squared\_error(y\_val, rf\_pred\_val\_tuned, squared=False)

print(f'Tuned Random Forest Validation RMSE: {rmse\_rf\_tuned}')

tuned\_rf\_predictions = best\_rf\_model.predict(X\_test)

1. Screenshots of Rank:



