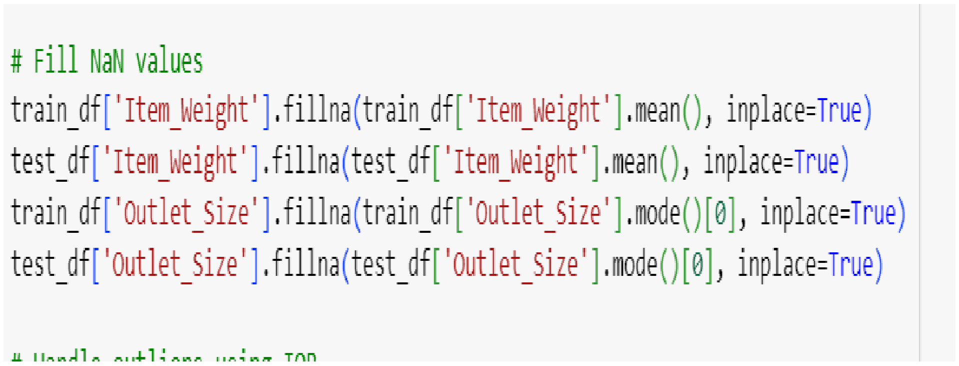
Data Collection and Preprocessing Phase

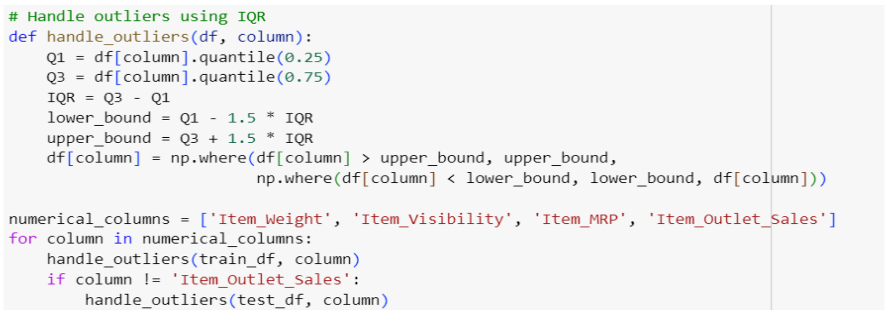
Date	15 July 2024
Team ID	740148
Project Title	Number Oracle:Big Mart Sales Predictive Analysis
Maximum Marks	6 Marks

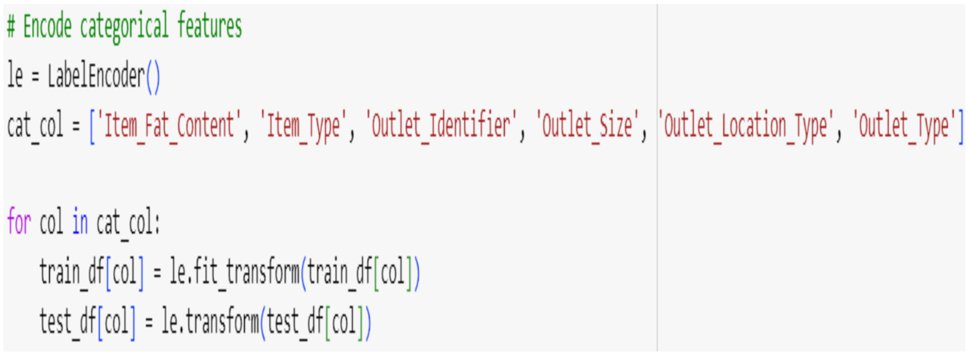
Data Exploration and Preprocessing Template

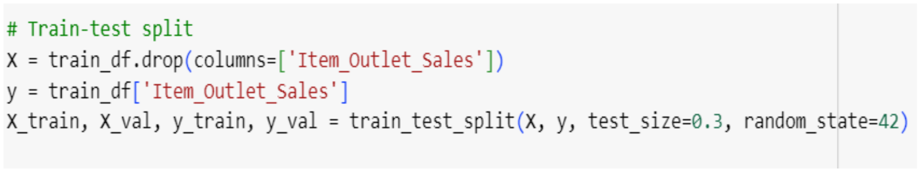
Identifies data sources, assesses quality issues like missing values and duplicates, and implements resolution plans to ensure accurate and reliable analysis.

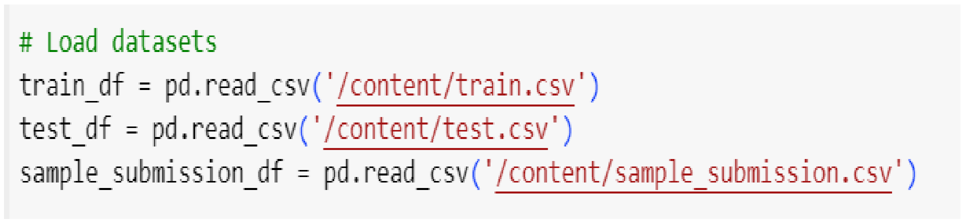
Section	**Description**
Data Overview	Basic statistics, dimensions, and structure of the data.
Univariate Analysis	Exploration of individual variables (mean, median, mode, etc.).
Bivariate Analysis	Relationships between two variables (correlation, scatter plots).
Multivariate Analysis	Patterns and relationships involving multiple variables.
Outliers and Anomalies	Identification and treatment of outliers.
Data Preprocessing Code Screenshots	

Handling Missing Data	
	`# Fill NaN values`
	`train_df['Item_Weight'].fillna(train_df['Item_Weight'].mean(), inplace=True)`
	`test_df['Item_Weight'].fillna(test_df['Item_Weight'].mean(), inplace=True)`
	`train_df['Outlet_Size'].fillna(train_df['Outlet_Size'].mode()[0], inplace=True)`
	`test_df['Outlet_Size'].fillna(test_df['Outlet_Size'].mode()[0], inplace=True)`

Data Transformation	
	`# Handle outliers using IQR`
	`def handle_outliers(df, column):`
	`    Q1 = df[column].quantile(0.25)`
	`    Q3 = df[column].quantile(0.75)`
	`    IQR = Q3 - Q1`
	`    lower_bound = Q1 - 1.5 * IQR`
	`    upper_bound = Q3 + 1.5 * IQR`
	`    df[column] = np.where(df[column] > upper_bound, upper_bound,`
	`                          np.where(df[column] < lower_bound, lower_bound, df[column]))`
	
	`numerical_columns = ['Item_Weight', 'Item_Visibility', 'Item_MRP', 'Item_Outlet_Sales']`
	`for column in numerical_columns:`
	`    handle_outliers(train_df, column)`
	`    if column != 'Item_Outlet_Sales':`
	`        handle_outliers(test_df, column)`

Feature Engineering	
	`# Encode categorical features`
	`le = LabelEncoder()`
	`cat_col = ['Item_Fat_Content', 'Item_Type', 'Outlet_Identifier', 'Outlet_Size', 'Outlet_Location_Type', 'Outlet_Type']`
	
	`for col in cat_col:`
	`    train_df[col] = le.fit_transform(train_df[col])`
	`    test_df[col] = le.transform(test_df[col])`

Save Processed Data	
	`# Train-test split`
	`X = train_df.drop(columns=['Item_Outlet_Sales'])`
	`y = train_df['Item_Outlet_Sales']`
	`X_train, X_val, y_train, y_val = train_test_split(X, y, test_size=0.3, random_state=42)`

Loading Data	
	`# Load datasets`
	`train_df = pd.read_csv('/content/train.csv')`
	`test_df = pd.read_csv('/content/test.csv')`
	`sample_submission_df = pd.read_csv('/content/sample_submission.csv')`