main

December 17, 2025

1 Assignment 1

PRN: 250840125020 (Harsh Chandrakar)
PRN: 250840125052 (Sumit Nagpure)

1.1 1. Load data and basic cleaning

[1]:
```
import pandas as pd
import numpy as np

df = pd.read_csv("train.csv")
df.head()
df.isna().sum()
```

[1]:
```
Item_Identifier                 0
Item_Weight                  1463
Item_Fat_Content                0
Item_Visibility                 0
Item_Type                       0
Item_MRP                        0
Outlet_Identifier               0
Outlet_Establishment_Year       0
Outlet_Size                  2410
Outlet_Location_Type            0
Outlet_Type                     0
Item_Outlet_Sales               0
dtype: int64
```

Typical cleaning for this dataset (same Big Mart problem).[1][2]

[2]:
```
data = df.copy()

# 1) Standardize Item_Fat_Content
data['Item_Fat_Content'] = data['Item_Fat_Content'].replace(
    {'LF': 'Low Fat', 'low fat': 'Low Fat', 'reg': 'Regular'}
)

# 2) Impute Item_Weight
```

```
data['Item_Weight'] = data.groupby('Item_Identifier')['Item_Weight']\
                           .transform(lambda x: x.fillna(x.mean()))
data['Item_Weight'] = data['Item_Weight'].fillna(data['Item_Weight'].mean())

# 3) Impute Outlet_Size by most frequent size per Outlet_Type
data['Outlet_Size'] = data.groupby('Outlet_Type')['Outlet_Size']\
                           .transform(lambda x: x.fillna(x.mode()[0]))
```

1.2 2. Drop composite key and prepare features

[3]:
```
target = 'Item_Outlet_Sales'
drop_cols = ['Item_Identifier', 'Outlet_Identifier']  # composite key

X = data.drop(columns=drop_cols + [target])
y = data[target]

# One-hot encode categoricals
X = pd.get_dummies(X, drop_first=True)
```

1.3 3. Train–test split

[4]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)
```

1.4 4. Build 3 models and evaluate

Use RMSE as the metric (common for this problem).[2][3]

[5]:
```
from sklearn.metrics import mean_squared_error, r2_score
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
import numpy as np
```

1.4.1 Model 1: Linear Regression

[6]:
```
lin = LinearRegression()
lin.fit(X_train, y_train)

pred_lin = lin.predict(X_test)
rmse_lin = mean_squared_error(y_test, pred_lin)
print("Linear Regression RMSE:", rmse_lin)
print("Linear Regression R2:", r2_score(y_test, pred_lin))
```

```
Linear Regression RMSE: 1145541.8463362118
Linear Regression R2: 0.5785303670536149
```

1.4.2 Model 2: Random Forest

[7]:
```
rf = RandomForestRegressor(
    n_estimators=300,
    max_depth=12,
    random_state=42,
    n_jobs=-1
)
rf.fit(X_train, y_train)

pred_rf = rf.predict(X_test)
rmse_rf = mean_squared_error(y_test, pred_rf)
print("Random Forest RMSE:", rmse_rf)
print("Linear Regression R2:", r2_score(y_test, pred_rf))
```

```
Random Forest RMSE: 1117818.4312321786
Linear Regression R2: 0.5887304113604099
```

1.4.3 Model 3: Gradient Boosting

[8]:
```
gb = GradientBoostingRegressor(
    n_estimators=300,
    learning_rate=0.05,
    max_depth=3,
    random_state=42
)
gb.fit(X_train, y_train)

pred_gb = gb.predict(X_test)
rmse_gb = mean_squared_error(y_test, pred_gb)
print("Gradient Boosting RMSE:", rmse_gb)
print("Linear Regression R2:", r2_score(y_test, pred_gb))
```

```
Gradient Boosting RMSE: 1081640.6838566265
Linear Regression R2: 0.6020409874480214
```

1.5 Conclusion

Final Summary In this project, I took the Big Mart sales data and `cleaned` it by filling in gaps and fixing typos. Once the data was ready, I put it through three different models to see which one was the best at predicting future sales.

The `Random Forest` and `Gradient Boosting` models came out on top, beating the basic `Linear Regression model`. This tells us that predicting sales is a bit complicated—factors like the type of item and the size of the store interact in ways that require these more advanced "AI" models to get an accurate result.