BigMart Sales Prediction

Sales Prediction using Machine Learning

Objective:

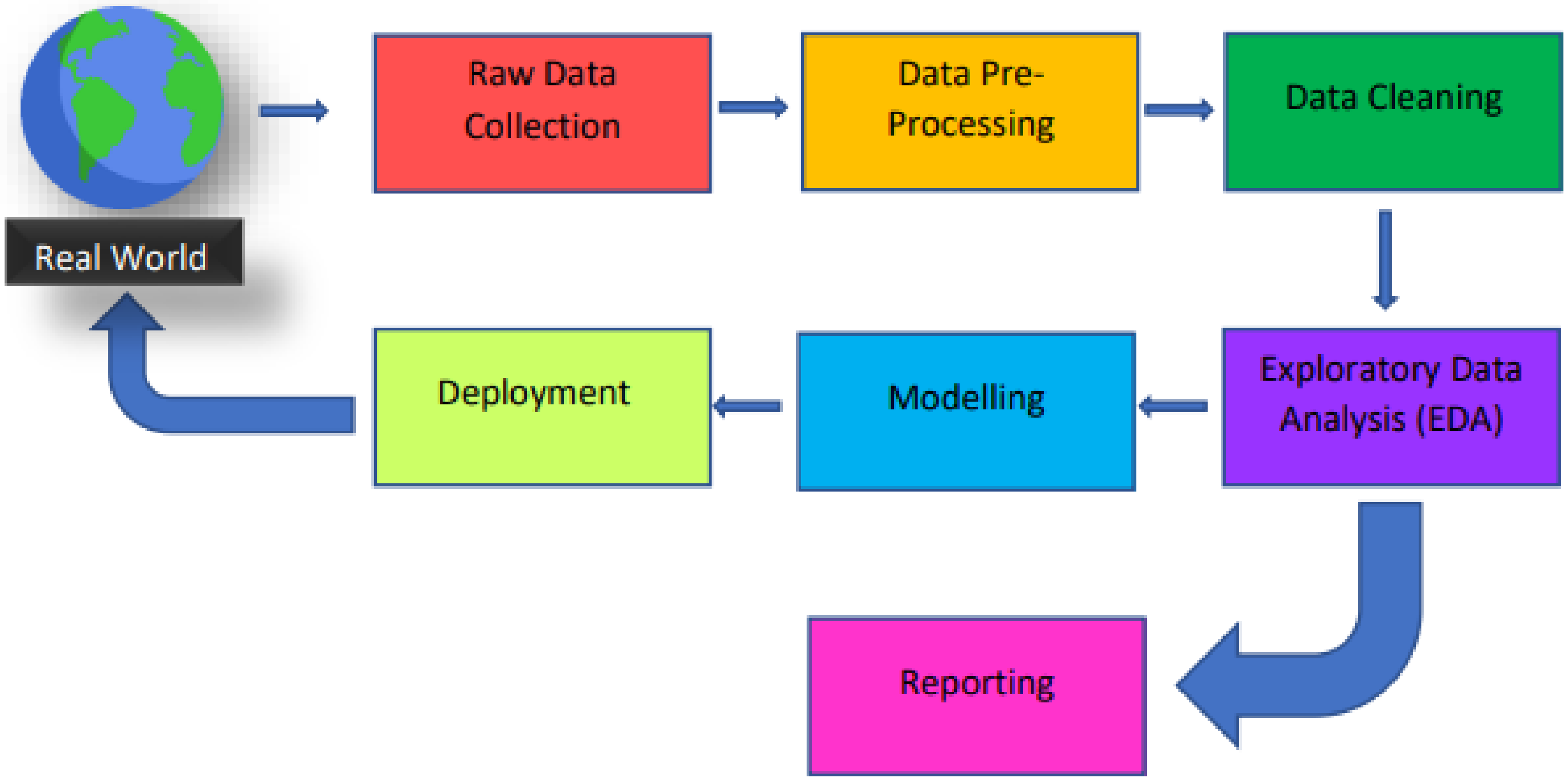
“To find out what role certain properties of an item play and how they affect their sales by understanding Big Mart sales.”

In order to help Big Mart, achieve this goal, a predictive model can be built to find out the sale of every item for every store. Also, the key factors that can increase their sales and what changes could be made to the product or store’s characteristics.

Benefits:

- Detection the features heavily responsible for item sales from particular outlet.
- Gives better insight of customers interest for the item.
- Helps in easy flow for managing resources.
- Manual inspection of what action needed to hike the sale.

Architecture



Data Description

- Item_Identifier: Unique product ID
- Item_Weight: Weight of the product
- Item_Fat_Content: Whether the product is low fat or not
- Item_Visibility: The % of the 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 the 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.

Importing Libraries

```
In [*]: import pandas as pd
        import matplotlib.pyplot as plt
        import pickle
        from pandas_profiling import ProfileReport
        import numpy as np
        from sklearn.preprocessing import LabelEncoder , StandardScaler
        import xgboost as xgb
        from sklearn.model_selection import train_test_split,GridSearchCV
        from sklearn.model_selection import RandomizedSearchCV
        from sklearn.metrics import mean_absolute_error,mean_squared_error,r2_score
        from sklearn. ensemble import GradientBoostingRegressor,RandomForestRegressor
```

Dataset

```
train_df.head()
```

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Location_Type	Outlet_Type	Item_Outlet_Sales
0	FDA15	9.30	Low Fat	0.016047	Dairy	249.8092	OUT049	1999	Medium	Tier 1	Supermarket Type1	3735.1380
1	DRC01	5.92	Regular	0.019278	Soft Drinks	48.2692	OUT018	2009	Medium	Tier 3	Supermarket Type2	443.4228
2	FDN15	17.50	Low Fat	0.016760	Meat	141.6180	OUT049	1999	Medium	Tier 1	Supermarket Type1	2097.2700
3	FDX07	19.20	Regular	0.000000	Fruits and Vegetables	182.0950	OUT010	1998	NaN	Tier 3	Grocery Store	732.3800
4	NCD19	8.93	Low Fat	0.000000	Household	53.8614	OUT013	1987	High	Tier 3	Supermarket Type1	994.7052

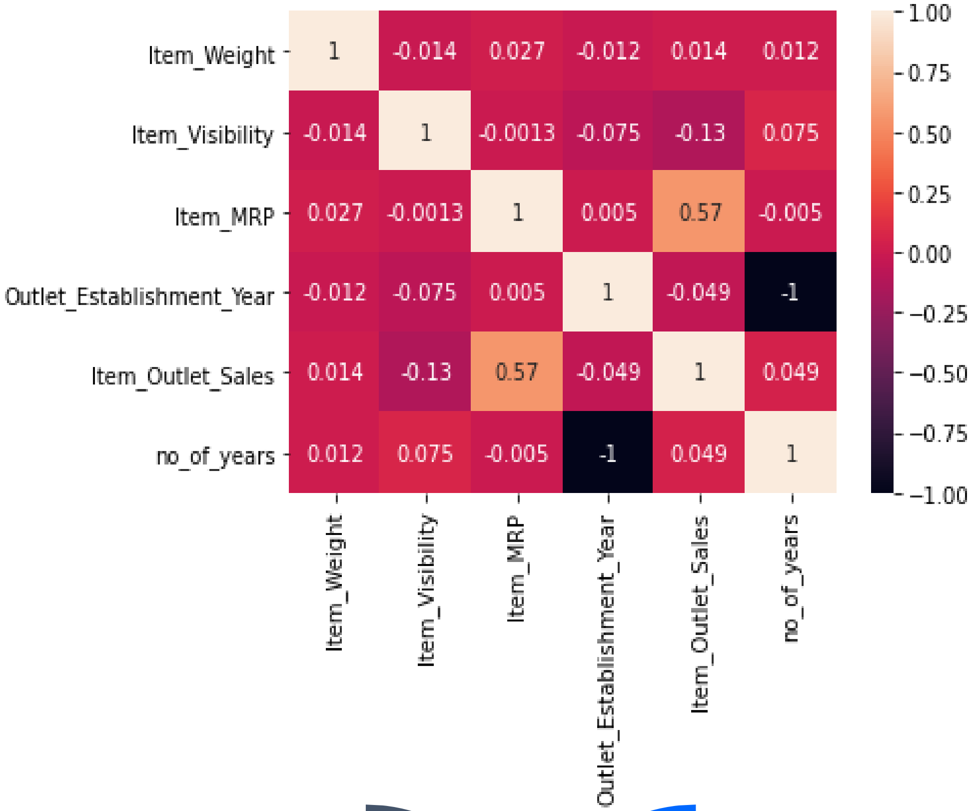
Dataset Info.

The data set consists of various data types from integer to float to object as shown in Fig.

```
In [5]:  1  # datatype of attributes
         2  df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8523 entries, 0 to 8522
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype
---  ------                     --------------  -----
 0   Item_Identifier            8523 non-null   object
 1   Item_Weight                7060 non-null   float64
 2   Item_Fat_Content           8523 non-null   object
 3   Item_Visibility            8523 non-null   float64
 4   Item_Type                  8523 non-null   object
 5   Item_MRP                   8523 non-null   float64
 6   Outlet_Identifier          8523 non-null   object
 7   Outlet_Establishment_Year  8523 non-null   int64
 8   Outlet_Size                6113 non-null   object
 9   Outlet_Location_Type       8523 non-null   object
 10  Outlet_Type                8523 non-null   object
 11  Item_Outlet_Sales          8523 non-null   float64
dtypes: float64(4), int64(1), object(7)
memory usage: 799.2+ KB
```

Correlation

Correlation is used to understand the relation between a target variable and predictors. In this work, Item-Sales is the target variable and its correlation with other variables is observed.



Handling Null Values

Fill Null Values

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
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
```

In [12]:
```
df['Item_Weight'] = df['Item_Weight'].fillna(df['Item_Weight'].mean())
```

In [13]:
```
df['Outlet_Size'].unique()
```

Out[13]:
```
array(['Medium', nan, 'High', 'Small'], dtype=object)
```

In [14]:
```
df = df.fillna({'Outlet_Size':'Medium'})
```

Label Encoding

Label Encoding

In [25]:
```
from sklearn.preprocessing import LabelEncoder
```

In [26]:
```
encode = LabelEncoder()
df["Item_Fat_Content"] = encode.fit_transform(df["Item_Fat_Content"])
df["Item_Type"] = encode.fit_transform(df["Item_Type"])
df["Outlet_Size"] =encode.fit_transform(df["Outlet_Size"])
df["Outlet_Type"] = encode.fit_transform(df["Outlet_Type"])
df["Outlet_Location_Type"] = encode.fit_transform(df["Outlet_Location_Type"])
df["Outlet_Identifier"] = encode.fit_transform(df["Outlet_Identifier"])
```

In [27]:
```
df
```

Out[27]:

	Item_Identifier	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Identifier	Outlet_Establishment_Year	Outlet_Size	Outlet_Locat
0	FDA15	9.300	0	0.016047	4	249.8092	9	1999	1	
1	DRC01	5.920	1	0.019278	14	48.2692	3	2009	1	
2	FDN15	17.500	0	0.016760	10	141.6180	9	1999	1	
3	FDX07	19.200	1	0.000000	6	182.0950	0	1998	1	
4	NCD19	8.930	0	0.000000	9	53.8614	1	1987	0	
...	...	...	...	...	...	...	...	...	...	
8518	FDF22	6.865	0	0.056783	13	214.5218	1	1987	0	

Standard Scaling

```
In [37]: from sklearn.preprocessing import StandardScaler

In [38]: scaler = StandardScaler()

In [39]: x_scaler = scaler.fit_transform(x)

In [44]: x_scaler = pd.DataFrame(data = x_scaler, columns= x.columns)

In [46]: x_scaler
```

Out[46]:

	Item_Weight	Item_Fat_Content	Item_Visibility	Item_Type	Item_MRP	Outlet_Size	Outlet_Type
0	-0.841872	-0.738147	-0.970732	-0.766479	1.747454	-0.284581	-0.252658
1	-1.641706	1.354743	-0.908111	1.608963	-1.489023	-0.284581	1.002972
2	1.098554	-0.738147	-0.956917	0.658786	0.010040	-0.284581	-0.252658
3	1.500838	1.354743	-1.281758	-0.291391	0.660050	-0.284581	-1.508289
4	-0.929428	-0.738147	-1.281758	0.421242	-1.399220	-1.950437	-0.252658
...	...	...	...	...	...	...	...
8518	-1.418084	-0.738147	-0.181193	1.371418	1.180783	-1.950437	-0.252658
8519	-1.059578	1.354743	-0.371154	-1.716656	-0.527301	-0.284581	-0.252658

Separating Dependent and Independent Variable

```
In [28]: x = df.drop(columns = ["Item_Identifier", "Item_Outlet_Sales"])
```

```
In [29]: y = df.Item_Outlet_Sales
```

Train Test Split

Train test split

```
In [52]: x_train, x_test, y_train, y_test = train_test_split(x_scaler, y , test_size=.20, random_state = 40)
```

Model Building

- Linear Regression
- Decision Tree Regressor
- K-Neighbors Regressor
- XGBoost Regressor
- Random Forest Regressor
- Gradient Boosting Regressor

Accuracy Of Model

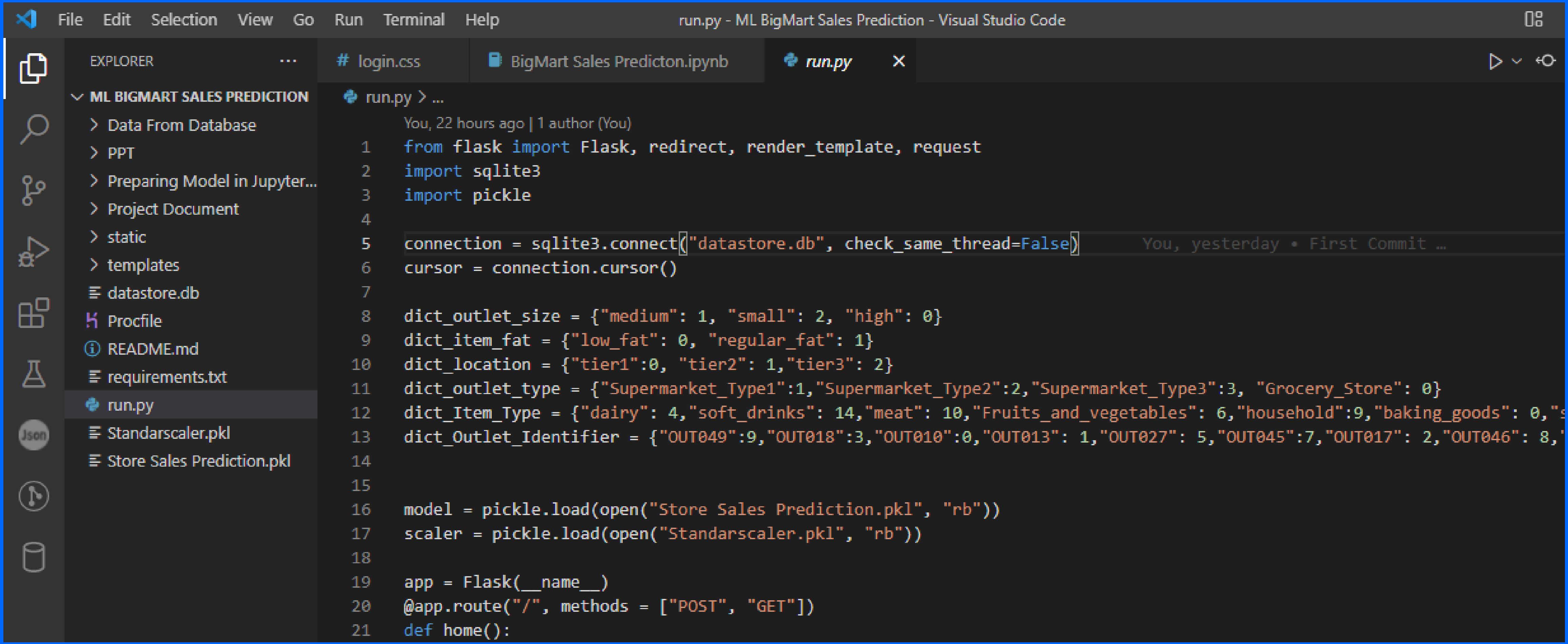
```
In [303]: xgb_model_1 = xgb.XGBRegressor(learning_rate = .01, n_estimators = 400)

In [304]: xgb_model_1.fit(x_train,y_train)

Out[304]: XGBRegressor(base_score=0.5, booster='gbtree', colsample_bylevel=1,
                       colsample_bynode=1, colsample_bytree=1, enable_categorical=False,
                       gamma=0, gpu_id=-1, importance_type=None,
                       interaction_constraints='', learning_rate=0.01, max_delta_step=0,
                       max_depth=6, min_child_weight=1, missing=nan,
                       monotone_constraints='()', n_estimators=400, n_jobs=8,
                       num_parallel_tree=1, predictor='auto', random_state=0, reg_alpha=0,
                       reg_lambda=1, scale_pos_weight=1, subsample=1, tree_method='exact',
                       validate_parameters=1, verbosity=None)

In [305]: xgb_model_1.score(x_train,y_train)

Out[305]: 0.8385486206294918

In [306]: xgb_model_1.score(x_test,y_test)

Out[306]: 0.8332926485737466

In [294]: y_test_pred = xgb_model_1.predict(x_test)
```

Model Saving

```
In [102]: pickle.dump(model, open("BigMart_Prediction.pkl", "wb"))
```

```
In [1]: pickle.dump(scaler, open("Standardscaler.pkl", "wb"))
```

API Using Flask



Deployment:

The Cloud environment was set up and the project was deployed from GitHub into Heroku cloud platform.

App link- https://bigmartprediction147.herokuapp.com/

Thank You

Team Name :- Ex-Holkarian