**Future Sales Prediction**

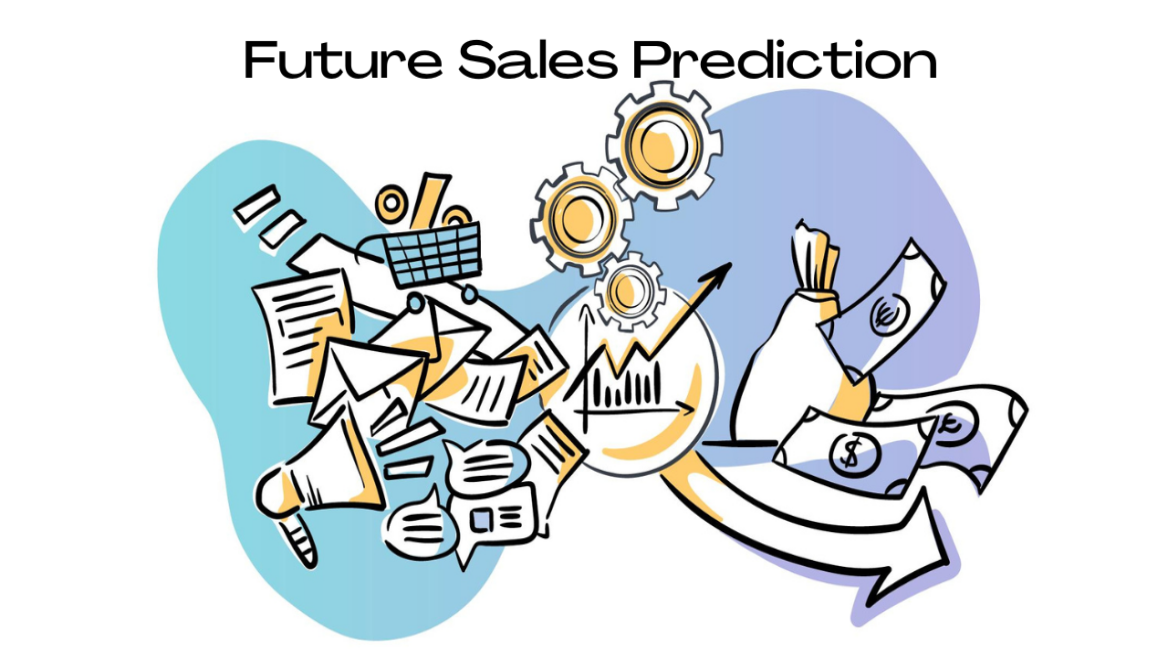
**Problem Statement**

The problem we aim to address is predicting future sales for a retail business. Accurate sales forecasts are critical for inventory management, financial planning, and overall business strategy. By using historical sales data, we want to build a time series forecasting model that can predict sales for future time periods. The project's goal is to develop a model that can help the business make informed decisions about inventory levels, marketing campaigns, and resource allocation.

**Design Thinking Process**

* Understanding the Problem
* **Data Collection**
* **Data Preprocessing**
* **Model Selection**
* **Model Training**
* **Evaluation**
* **Deployment**

1. **Understanding the Problem:** We started by defining the problem and its significance to the business. We discussed the specific challenges related to sales forecasting, such as seasonality, trends, and the impact of external factors**.**
2. **Data Collection:** We gathered historical sales data, which is essential for training and evaluating our forecasting model.
3. **Data Preprocessing:** We cleaned and prepared the dataset by handling missing values, encoding categorical features, and resampling time series data to create consistent intervals.
4. **Model Selection:** We researched and selected an appropriate time series forecasting algorithm based on the characteristics of the data. In this case, we chose to use an ARIMA (Auto Regressive Integrated Moving Average) model.
5. **Model Training:** We split the dataset into training and testing sets. The model was trained on the training data, and we fine-tuned its hyperparameters to achieve the best performance.
6. **Evaluation:** We used evaluation metrics such as Mean Absolute Error (MAE), Mean Squared Error (MSE), and Root Mean Squared Error (RMSE) to assess the model's accuracy. We also visualized the model's predictions against actual sales to get insights.
7. **Deployment:** In this project, we focus on creating a predictive model. However, future work may involve deploying the model into a production environment for real-time forecasting

****

**Phases of Development**

* Data Collection**:** Gather historical sales data from the business database.
* **Data Preprocessing**: Clean and preprocess the data, including handling missing values, encoding categorical features, and resampling.
* **Model Selection**: Choose the appropriate time series forecasting algorithm. In this project, we selected ARIMA.
* **Model Training**: Split the data into training and testing sets, train the model, and fine-tune hyperparameters.
* Evaluation: Assess the model's performance using relevant metrics.

**Dataset Description**

The dataset used for this project contains historical sales data, including information about time periods, product IDs, sales quantities, and possibly other relevant features. The data is structured as a time series, which is crucial for our forecasting task.

**Data Preprocessing**

**Data preprocessing involved the following steps**:

* Handling missing values
* Encoding categorical features
* Resampling the time series data to create consistent time intervals

**Model Training Process**

The chosen model for this project is the ARIMA (Auto Regressive Integrated Moving Average) model.

**The training process involves:**

* Splitting the dataset into training and testing sets.
* Training the ARIMA model on the training data.
* Fine-tuning model hyperparameters, such as order (p, d, q) for ARIMA
* Making predictions on the testing set.

Choice of Time Series Forecasting Algorithm and Evaluation Metrics

We selected the ARIMA model as it is well-suited for time series forecasting tasks and can capture seasonality and trends in the data.

The choice of evaluation metrics includes:

* Mean Absolute Error (MAE)
* Mean Squared Error (MSE)
* Root Mean Squared Error (RMSE)

These metrics provide a quantitative measure of the model's accuracy in predicting future sales.

**import pandas as pd**

**df1=pd.read\_csv('/tmp/train.csv')**

**df2=pd.read\_csv('/tmp/test.csv')**

**df1.head(3)**

|  | **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 |

**df1.isnull().sum()**

**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

**df2.isnull().sum()**

Item\_Identifier 0

Item\_Weight 976

Item\_Fat\_Content 0

Item\_Visibility 0

Item\_Type 0

Item\_MRP 0

Outlet\_Identifier 0

Outlet\_Establishment\_Year 0

Outlet\_Size 1606

Outlet\_Location\_Type 0

Outlet\_Type 0

dtype: int64

**df1['Item\_Type'].value\_counts()**

**import numpy as np**

**import matplotlib.pyplot as plt plt.bar(df1['Item\_Type'].value\_counts().index, df1['Item\_Type'].value\_counts(), width=0.5, bottom=None, align='center', data=df1) plt.title('Item\_Type Distribution')**

**plt.xticks(rotation='vertical')**

**plt.xlabel('Item\_Type')**

**plt.ylabel('Frequency')**

**print('Item\_Type:\n',df1['Item\_Type'].value\_counts())**

Item\_Type:

Fruits and Vegetables 1232

Snack Foods 1200

Household 910

Frozen Foods 856

Dairy 682

Canned 649

Baking Goods 648

Health and Hygiene 520

Soft Drinks 445

Meat 425

Breads 251

Hard Drinks 214

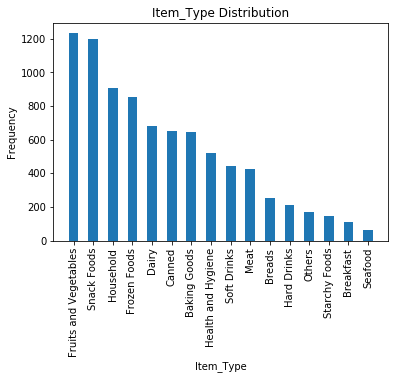
Others 169

Starchy Foods 148

Breakfast 110

Seafood 64

Name: Item\_Type, dtype: int64



**df1['Item\_Type'].value\_counts()**

**import numpy as np**

**import matplotlib.pyplot as plt plt.bar(df1['Item\_Type'].value\_counts().index, df1['Item\_Type'].value\_counts(), width=0.5, bottom=None, align='center', data=df1) plt.title('Item\_Type Distribution')**

**plt.xlabel('Item\_Type')**

**plt.ylabel('Frequency')**

**print('Item\_Type:\n',df1['Item\_Type'].value\_counts())**

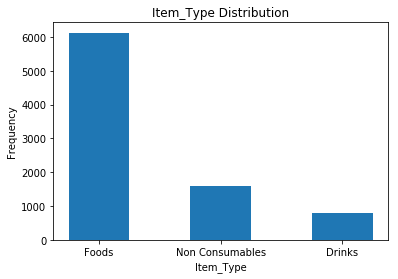
Item\_Type:

Foods 6125

Non Consumables 1599

Drinks 799

Name: Item\_Type, dtype: int64



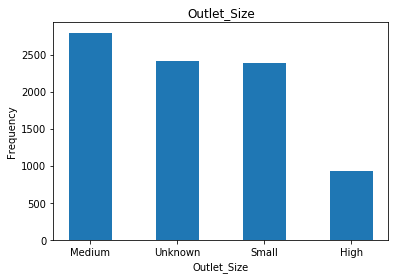
**df1['Outlet\_Size'].value\_counts()**

**import numpy as np**

**import matplotlib.pyplot as plt plt.bar(df1['Outlet\_Size'].value\_counts().index, df1['Outlet\_Size'].value\_counts(), width=0.5, bottom=None, align='center', data=df1) plt.title('Outlet\_Size')**

**plt.xlabel('Outlet\_Size')**

**plt.ylabel('Frequency')**



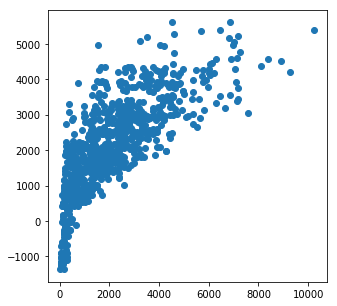
## ****Linear Regression****

**from sklearn.linear\_model**

**import LinearRegression lm=LinearRegression() lm.fit(x\_train,y\_train)**

LinearRegression(copy\_X=True, fit\_intercept=True, n\_jobs=None, normalize=False)

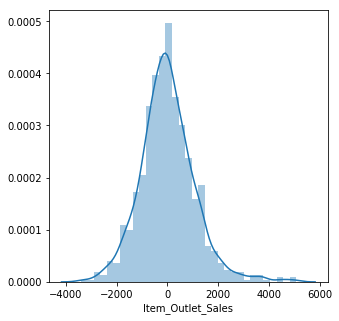
**predictions=lm.predict(x\_val)**



**import seaborn as sns**

**sns.distplot((y\_val-predictions))**

<matplotlib.axes.\_subplots.AxesSubplot at 0x7f71d326f4e0>



## ****Random Forest****

**from sklearn.ensemble**

**import RandomForestRegressor rf=RandomForestRegressor(n\_estimators=400,max\_depth=6, min\_samples\_leaf=76,n\_jobs=4) rf.fit(x\_train,y\_train)**

RandomForestRegressor(bootstrap=True, criterion='mse', max\_depth=6,

max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=76, min\_samples\_split=2,

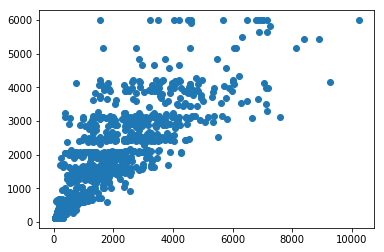
min\_weight\_fraction\_leaf=0.0, n\_estimators=400, n\_jobs=4,

oob\_score=False, random\_state=None, verbose=0,

warm\_start=False)

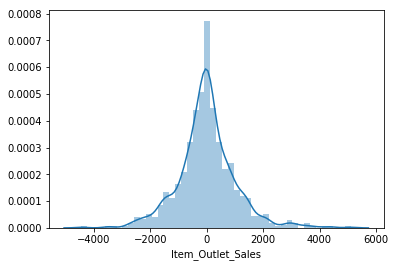
**predictions1=rf.predict(x\_val)**

**plt.scatter(y\_val,predictions1)**



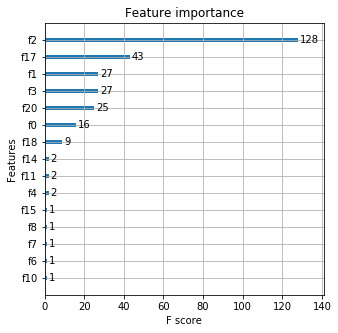
**import seaborn as sns**

**sns.distplot((y\_val-predictions1))**



**xgb.plot\_importance(xg\_reg)**

**plt.rcParams['figure.figsize'] = [5, 5] plt.show()**



SCOPE OF FUTURE USE:

The application can be developed further by providing more accurate results along with ease of usage. Stock prediction can also be done if this is developed accordingly. Also, the predictions can be made not just for advertisements but also based on other factors that influence sales.

Sales forecasting allows companies to efficiently allocate resources for future growth and manage its cash flow. Sales forecasting also helps businesses to estimate their costs and revenue accurately based on which they are able to predict their short-term and long-term performance.



**Conclusion**

**Therefore, we can predict sales based on the amount spent on advertisements. As there might be huge data for input we are using a logical regression algorithm and it is a better algorithm for adapting to changes in the dataset. Also, we displayed GUI for the user's convenience.**