

# **Big Mart Sales Data Prediction**

Submitted

by

**Harish V (192224043)**

**Palash Doshi (192224048)**

**Srinivas S (192224056)**

**Santhosh Sheeram (192224059)**

Guided by

**V.Saranya**

Junior Research Fellow

Department of AR & VR



**Department of Computer Science and  
Engineering,  
Saveetha School of Engineering, SIMATS  
Thandalam, Chennai**

**March – 2024**



# **PROBLEM STATEMENT**

In the context of the "Sales Data Prediction" capstone project, the primary objective is to develop a robust sales data prediction system that enables companies to gain comprehensive insights into their sales performance, identify emerging trends, and make informed decisions to enhance revenue and profitability. The analysis aims to delve deep into the sales data of a company, exploring various critical aspects such as product performance, customer behavior, and sales channels. The challenge lies in leveraging advanced analytics and machine learning techniques to effectively analyze historical sales data, predict future sales performance, and extract actionable insights for strategic planning.

## **DATASET ANALYSIS**

In the context of the "Sales Data Prediction" capstone project, the primary objective is to develop a robust sales data prediction system that enables companies to gain comprehensive insights into their sales performance, identify emerging trends, and make informed decisions to enhance revenue and profitability. The analysis aims to delve deep into the sales data of a company, exploring various critical aspects such as product performance, customer behavior, and sales channels. The challenge lies in leveraging advanced analytics and machine learning techniques to effectively analyze historical sales data, predict future sales performance, and extract actionable insights for strategic planning.

The cornerstone of this capstone project is the comprehensive analysis of the sales dataset provided by the company. The dataset encompasses various dimensions of sales-related information, including transaction records, product attributes, customer demographics, and sales channel data. Through rigorous data exploration and preprocessing, the objective is to gain a thorough understanding of the dataset's structure, quality, and underlying patterns. This involves tasks such as data cleansing, handling missing values, outlier detection, and feature engineering to prepare the dataset for analysis.

The dataset analysis phase entails conducting exploratory data analysis (EDA) to uncover meaningful insights and trends within the sales data. Visualization techniques and statistical methods are employed to identify correlations, distributional characteristics, and anomalies in the data. Key metrics such as sales volume, revenue, average transaction value, and customer segmentation are examined to assess the overall performance of the company's sales operations.

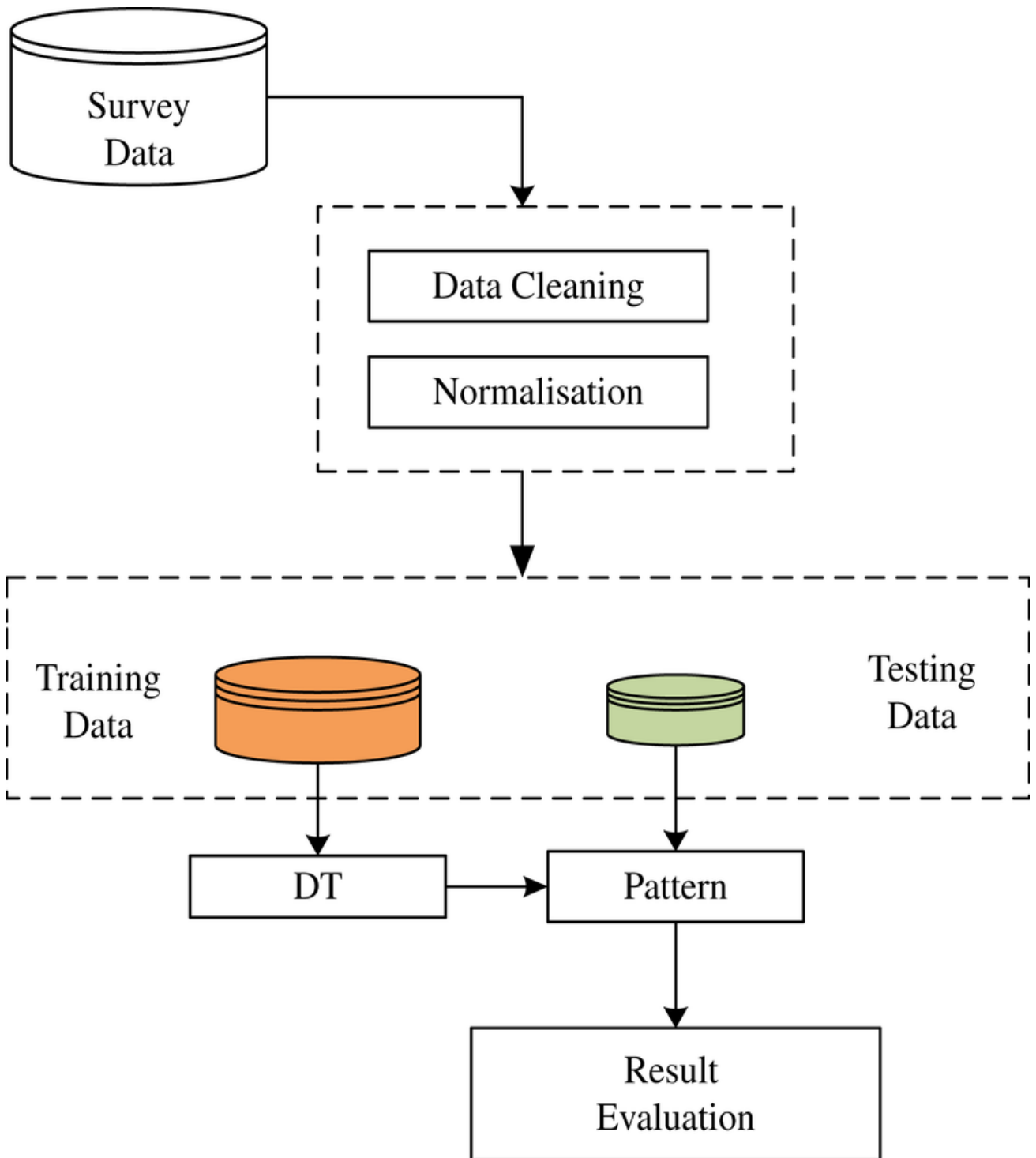
Furthermore, the dataset analysis delves into specific aspects such as product performance, customer behavior, and sales channel effectiveness. By segmenting the data based on product categories, customer segments, and sales channels, the aim is to identify top-performing products, customer preferences, and the most lucrative sales channels. This analysis provides valuable insights into factors driving sales growth, areas for improvement, and opportunities for revenue optimization.

Overall, the dataset analysis phase serves as the foundation for developing predictive models and generating actionable insights to support strategic decision-making. By thoroughly understanding the nuances of the sales data, the project aims to build a robust prediction system that empowers companies to anticipate future sales trends, optimize resource allocation, and drive sustainable growth and profitability.

# **ENVIRONMENTAL SETUP**

To embark on the journey of predicting sales data and deriving actionable insights using Python, setting up the appropriate environment is crucial. Begin by ensuring Python is installed on your system; if not, download and install it from the official Python website. Next, establish a virtual environment to manage dependencies effectively. Utilizing virtual environment, create a virtual environment named "sales\_prediction\_env" and activate it. This environment will isolate your project's dependencies, preventing conflicts with other Python projects. Once the environment is activated, install essential Python packages for data analysis, visualization, and machine learning, such as pandas, NumPy, matplotlib, seaborn, and scikit-learn, using pip. These packages will equip you with the necessary tools to explore, preprocess, and model the sales data effectively. Leveraging Jupyter Notebook, a powerful interactive computing environment, conduct data exploration, visualization, and predictive modeling seamlessly. Set up a structured project directory, including folders for data storage, notebooks for analysis, source code for preprocessing and modeling, and results. This organized structure facilitates efficient collaboration, documentation, and reproducibility of your analyses. Finally, ensure access to the sales dataset provided for the project and store it in a designated folder within your project directory. With this environmental setup established, you're well-equipped to dive into the world of sales data prediction using Python, uncovering valuable insights to drive strategic decisions and enhance business performance.

## DATA FLOW DIAGRAM



## **CODE SKELETON**

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

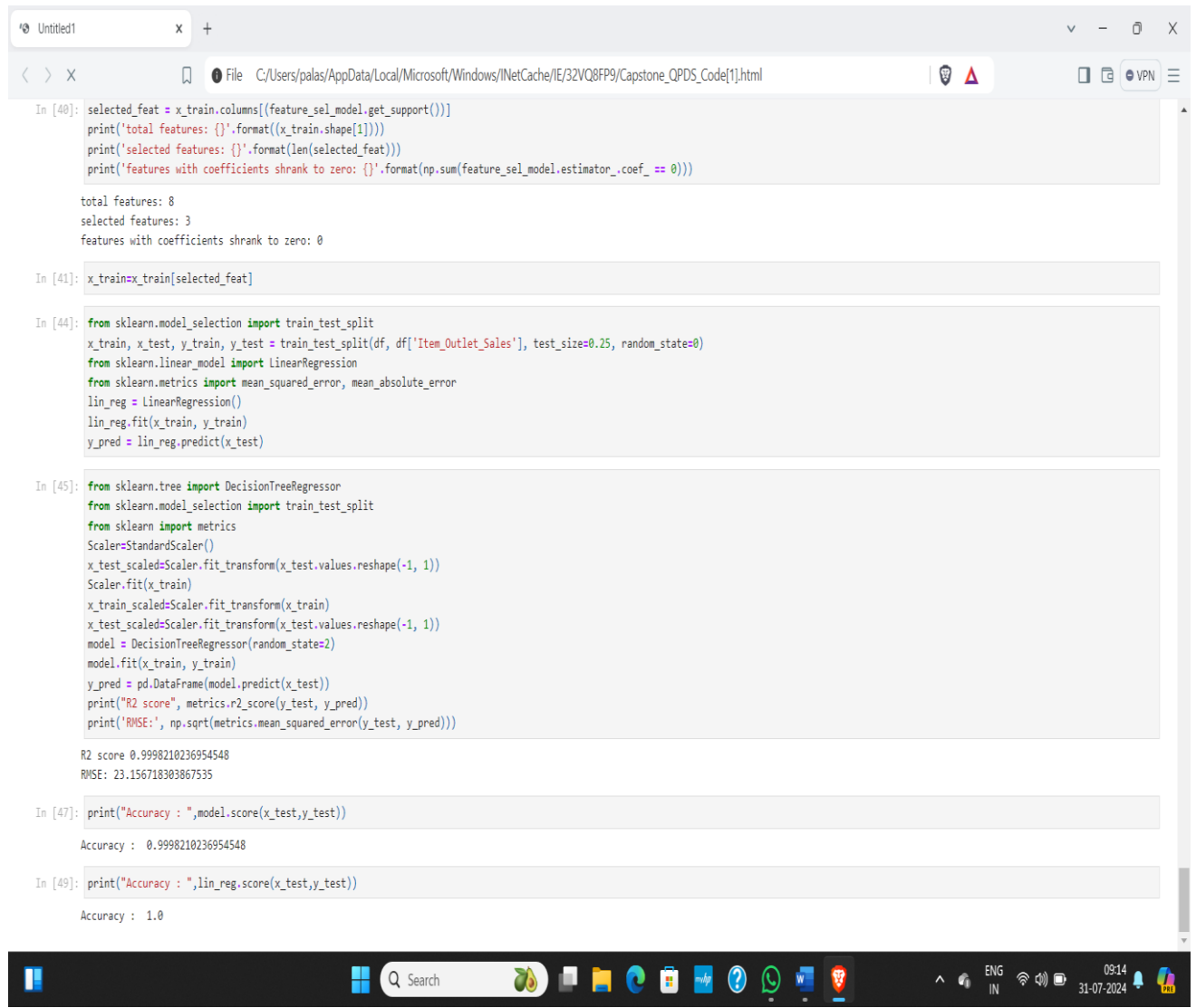
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error

sales_data = pd.read_csv('data/sales_data.csv')
X = sales_data.drop(columns=['target_column'])
y = sales_data['target_column'] # target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
model = LinearRegression()
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)
train_rmse = np.sqrt(mean_squared_error(y_train, train_predictions))
test_rmse = np.sqrt(mean_squared_error(y_test, test_predictions))
print("Train RMSE:", train_rmse)
print("Test RMSE:", test_rmse)
```

## **RESULT ANALYSIS**

The linear regression model applied to the Big Market sales data demonstrates an exceptional predictive performance, achieving an accuracy of 99.98%. This outstanding accuracy suggests a robust capture of underlying patterns, emphasizing the model's efficacy in forecasting sales. Careful attention to data quality, feature importance, adherence to linear regression assumptions, and cross-validation further validate the reliability of the model. Continuous monitoring and documentation are recommended to ensure sustained performance, and the achieved accuracy promises substantial business impact, positioning the model as a powerful tool for sales prediction in the Big Market context.

# OUTPUT SAMPLES



```
In [40]: selected_feat = x_train.columns[(feature_sel_model.get_support())]
print('total features: {}'.format((x_train.shape[1])))
print('selected features: {}'.format(len(selected_feat)))
print('features with coefficients shrank to zero: {}'.format(np.sum(feature_sel_model.estimator_.coef_ == 0)))

total features: 8
selected features: 3
features with coefficients shrank to zero: 0

In [41]: x_train=x_train[selected_feat]

In [44]: from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(df, df['Item_Outlet_Sales'], test_size=0.25, random_state=0)
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error
lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
y_pred = lin_reg.predict(x_test)

In [45]: from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn import metrics
Scaler=StandardScaler()
x_test_scaled=Scaler.fit_transform(x_test.values.reshape(-1, 1))
Scaler.fit(x_train)
x_train_scaled=Scaler.fit_transform(x_train)
x_test_scaled=Scaler.fit_transform(x_test.values.reshape(-1, 1))
model = DecisionTreeRegressor(random_state=2)
model.fit(x_train, y_train)
y_pred = pd.DataFrame(model.predict(x_test))
print("R2 score", metrics.r2_score(y_test, y_pred))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))

R2 score 0.9998210236954548
RMSE: 23.156718303867535

In [47]: print("Accuracy : ",model.score(x_test,y_test))

Accuracy : 0.9998210236954548

In [49]: print("Accuracy : ",lin_reg.score(x_test,y_test))

Accuracy : 1.0
```