Project Report

Walmart Sales Prediction

using Machine Learning

Team id - 592766

Members -

Anand Misra

Mayank Mishra

Vishvender Tyagi

Sourav Ranjan Meher

1. INTRODUCTION

1.1 Project Overview

The Walmart Sales Analysis project is an initiative aimed at revolutionizing the retail industry by leveraging advanced analytics to glean actionable insights from sales data. The project focuses on enhancing decision-making processes, optimizing inventory management, and boosting overall operational efficiency within the retail sector.

1.2 Purpose

The Walmart Sales Analysis project is an initiative aimed at revolutionizing the retail industry by leveraging advanced analytics to glean actionable insights from sales data. The project focuses on enhancing decision-making processes, optimizing inventory management, and boosting overall operational efficiency within the retail sector.

2. LITERATURE SURVEY

2.1 Existing problem

The retail industry faces persistent challenges related to the efficient analysis of sales data. Timely and accurate insights are crucial for adapting to market dynamics, consumer behavior shifts, and optimizing operational processes. The existing problem lies in the need for a comprehensive system that seamlessly integrates diverse data sources, ensuring reliable and actionable analytics.

2.2 References

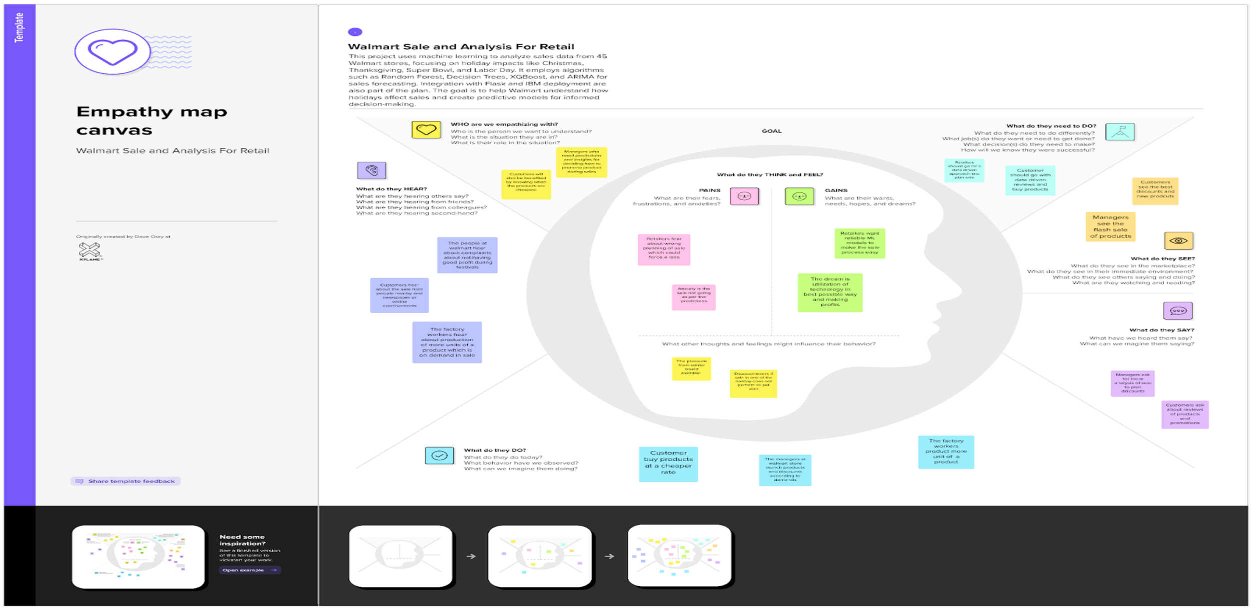
1. McKinsey & Company. "Retail Analytics: The Secret Weapon." Harvard Business Review.
2. Bernard Marr. "Predictive Analytics in Retail." Forbes.

2.3 Problem Statement Definition

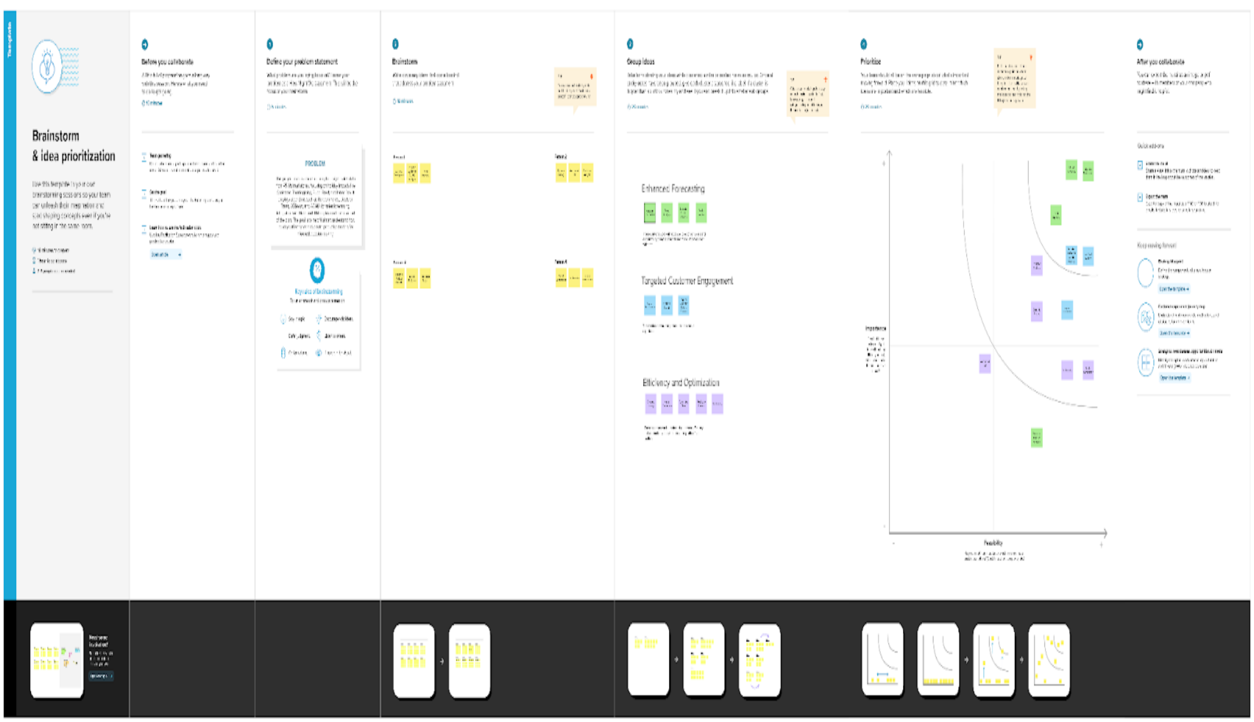
The project addresses the challenge of insufficient and outdated sales analysis tools in the retail sector. It aims to provide a sophisticated solution capable of handling diverse data sources, facilitating real-time insights, and enabling proactive decision-making.

3. IDEATION & PROPOSED SOLUTION

3.1 Empathy Map Canvas (png included in ideation phase folder)



3.2 Ideation & Brainstorming (png included in ideation phase folder)



4. REQUIREMENT ANALYSIS

4.1 Functional requirement

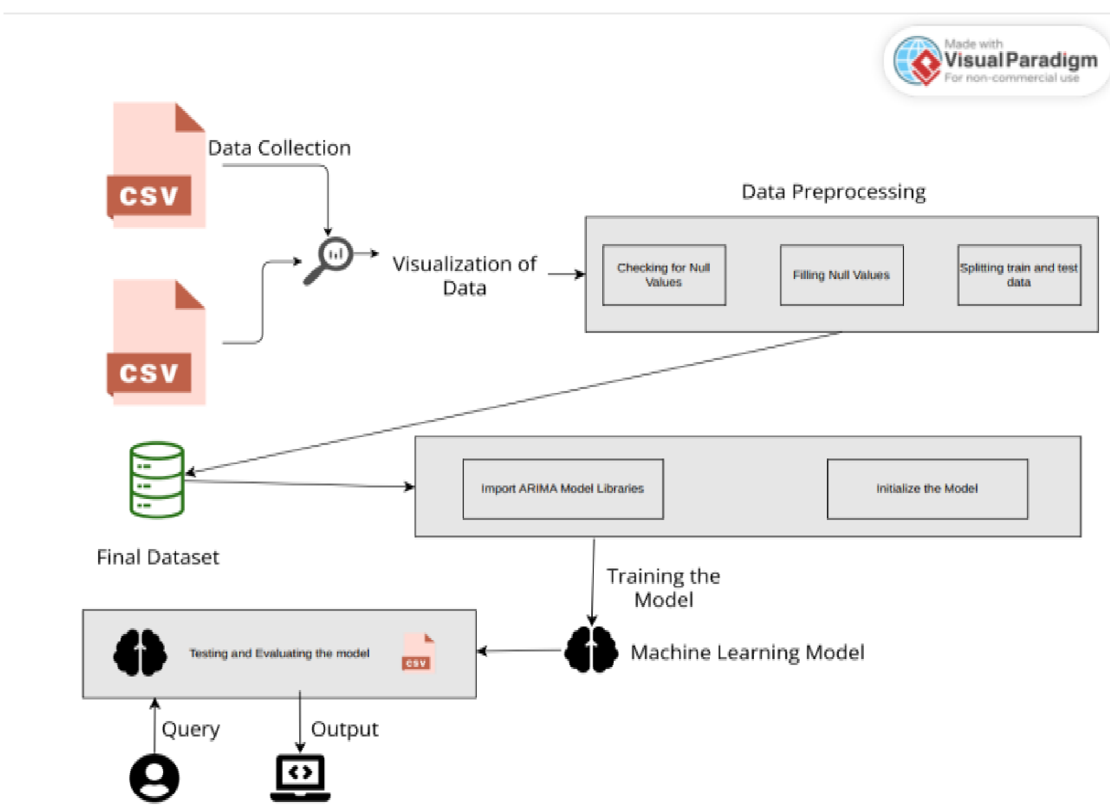
The project's functional requirements include the collection of daily sales data from physical retail stores and the integration of online sales data with physical store sales data. These features are essential for creating a holistic view of Walmart's overall sales performance.

4.2 Non-Functional requirements

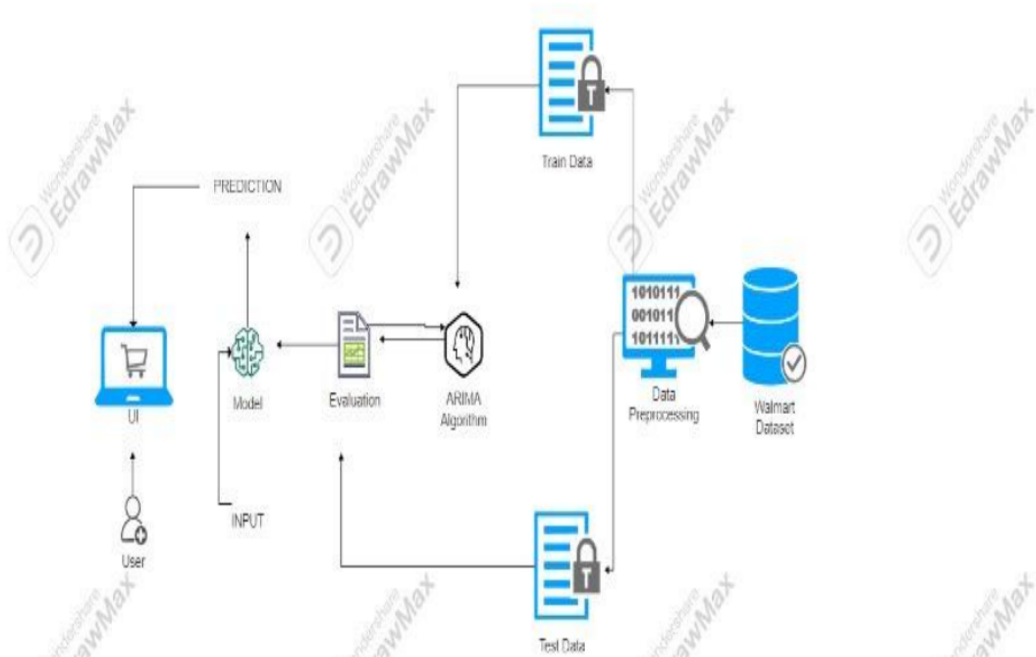
Non-functional requirements emphasize the need for data accuracy and integrity. The system is designed to provide real-time analytics, ensuring that decision-makers have access to the most up-to-date and reliable information.

5. PROJECT DESIGN

5.1 Data Flow Diagrams & User Stories

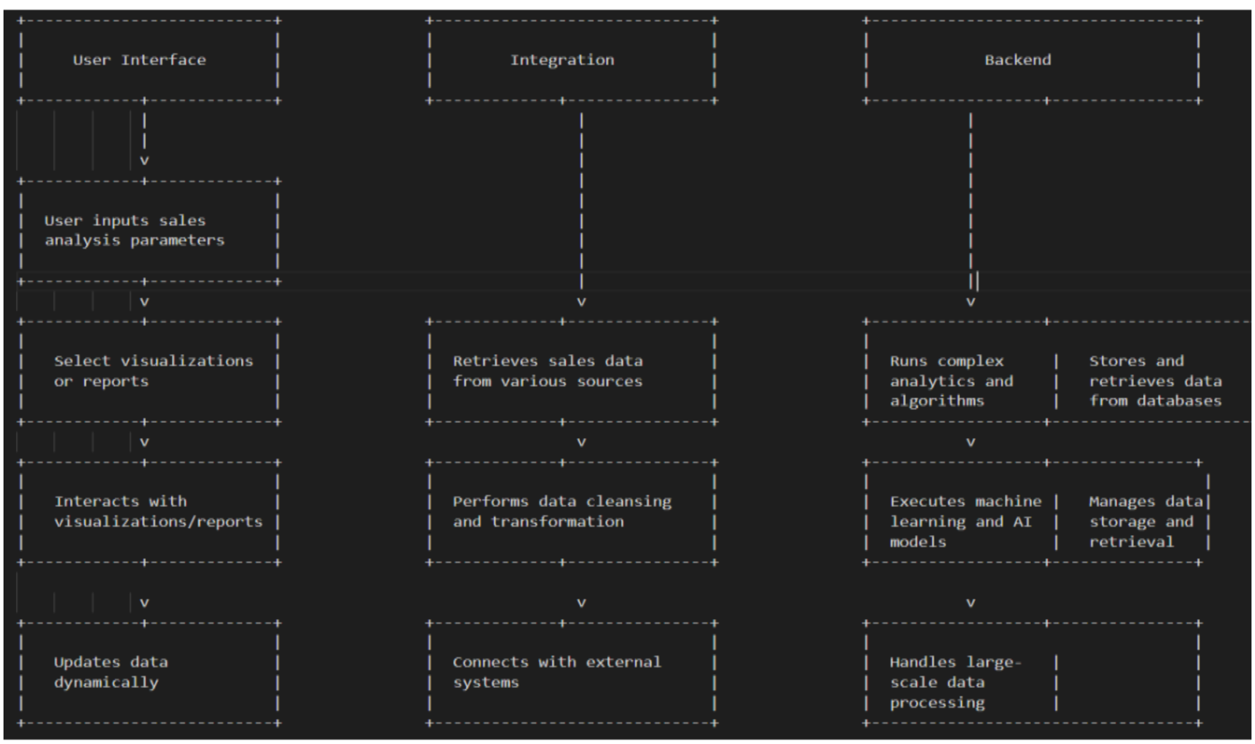


5.2 Solution Architecture



6. PROJECT PLANNING & SCHEDULING

6.1 Technical Architecture



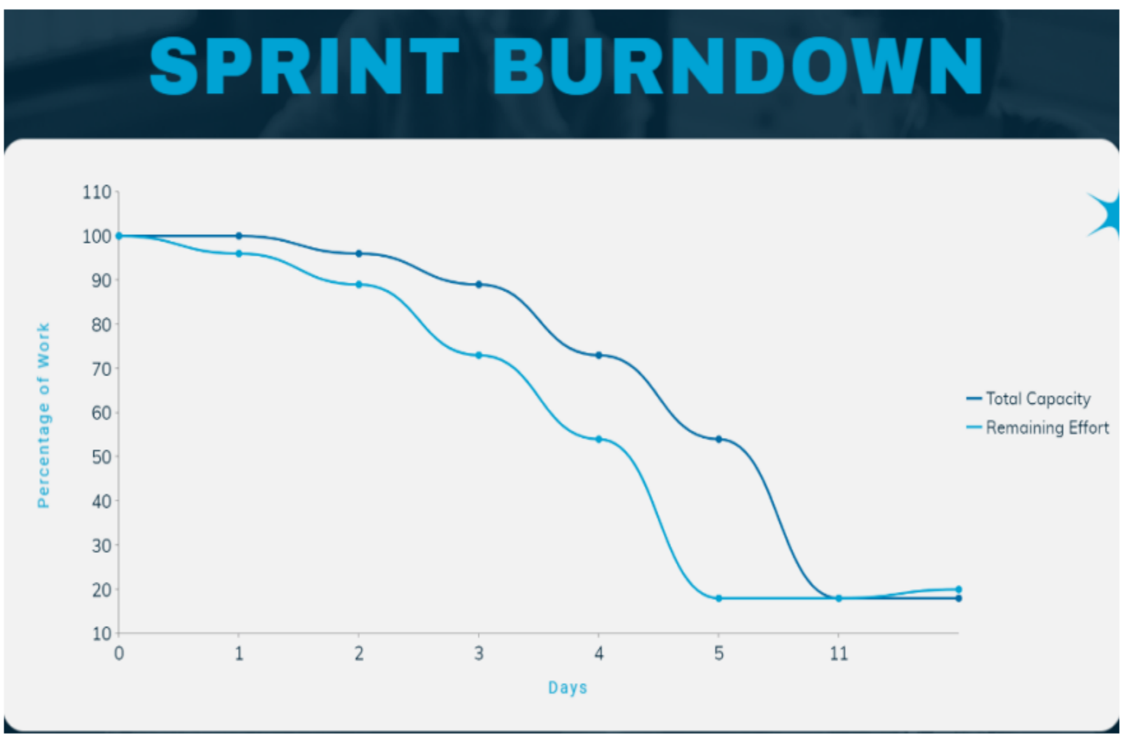
6.2 Sprint Planning & Estimation

Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	5	2 Days	8 Nov 2023	9 Nov 2023	5	9 Nov 2023
Sprint-2	7	1 Days	10 Nov 2023	10 Nov 2023	7	10 Nov 2023
Sprint-3	8	1 Days	11 Nov 2023	11 Nov 2023	8	11 Nov 2023
Sprint-4	15	7 Days	12 Nov 2023	18 Nov 2023	15	18 Nov 2023
Sprint-5	7	3 Days	19 Nov 2023	21 Nov 2023	7	21 Nov 2023

6.3 Sprint Delivery Schedule

Burndown Chart:



7. CODING & SOLUTIONING (Explain the features added in the project along with code)

7.1 Visualizing and Analyzing the data

In the initial phase of our machine learning project, a thorough descriptive analysis was conducted to gain insights into the dataset's characteristics. Visualizations such as histograms, box plots, and scatter plots were employed to understand the distribution, central tendency, and relationships between variables. This step provided a foundation for subsequent data processing and modeling efforts.

Descriptive Analysis

```
train.describe()
```

	Store	Dept	Weekly_Sales
count	421570.000000	421570.000000	421570.000000
mean	22.200546	44.260317	15981.258123
std	12.785297	30.492054	22711.183519
min	1.000000	1.000000	-4988.940000
25%	11.000000	18.000000	2079.650000
50%	22.000000	37.000000	7612.030000
75%	33.000000	74.000000	20205.852500
max	45.000000	99.000000	693099.360000

```
[8] features.describe()
```

	Store	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment
count	8190.000000	8190.000000	8190.000000	4032.000000	2921.000000	3613.000000	3464.000000	4050.000000	7605.000000	7605.000000
mean	23.000000	59.356198	3.405992	7032.371786	3384.176594	1760.100180	3292.935886	4132.216422	172.460809	7.826821
std	12.987966	18.678607	0.431337	9262.747448	8793.583016	11276.462208	6792.329861	13086.690278	39.738346	1.877259
min	1.000000	-7.290000	2.472000	-2781.450000	-265.760000	-179.260000	0.220000	-185.170000	126.064000	3.684000
25%	12.000000	45.902500	3.041000	1577.532500	68.880000	6.600000	304.687500	1440.827500	132.364839	6.634000
50%	23.000000	60.710000	3.513000	4743.580000	364.570000	36.260000	1176.425000	2727.135000	182.764003	7.806000
75%	34.000000	73.880000	3.743000	8923.310000	2153.350000	163.150000	3310.007500	4832.555000	213.932412	8.567000
max	45.000000	101.950000	4.468000	103184.980000	104519.540000	149483.310000	67474.850000	771448.100000	228.976456	14.313000

7.2 Data Pre-processing

To ensure the quality and integrity of the dataset, a comprehensive data pre-processing stage was undertaken. Null values were identified and appropriately addressed, either through imputation or removal, depending on the nature and significance of the missing data. Negative values, if present, were handled using suitable techniques such as scaling or transformation. Exploratory data analysis (EDA) was performed to uncover patterns, outliers, and potential feature engineering opportunities. Categorical values were encoded or transformed, and the dataset was split into training and testing sets to facilitate model evaluation.

Data Preprocessing

Checking for null values

8 cells hidden

Handling Negative Values

[ ] 4 cells hidden

Exploratory Data Analysis

[ ] 3 cells hidden

Handling Categorical Values

[ ] 9 cells hidden

Splitting data into train and test

[ ] 2 cells hidden

```
features.isnull().sum()
```

```
Store              0
Date               0
Temperature        0
Fuel_Price         0
MarkDown1       4158
MarkDown2       5269
MarkDown3       4577
MarkDown4       4726
MarkDown5       4140
CPI              585
Unemployment     585
IsHoliday          0
dtype: int64
```

```
[16] stores.isnull().sum()
```

```
Store    0
Type     0
Size     0
dtype: int64
```

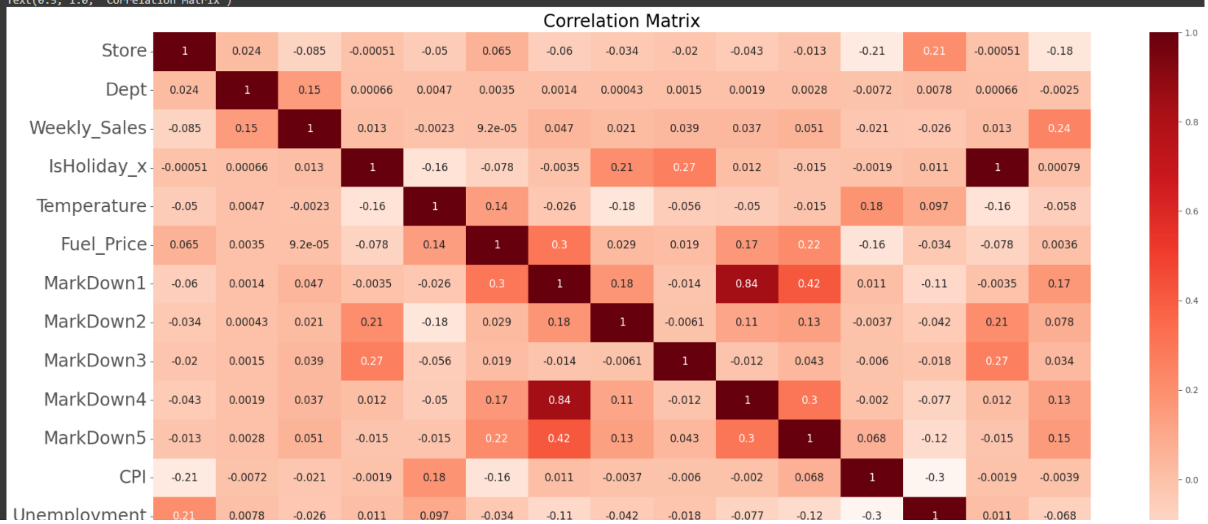
```
[17] data = train.merge(features, on=['Store', 'Date'], how='inner').merge(stores, on=['Store'], how='inner')
     print(data.shape)
```

```
(421570, 17)
```

```
[18] data['MarkDown1'] = data['MarkDown1'].replace(np.nan, 0)
     data['MarkDown2'] = data['MarkDown2'].replace(np.nan, 0)
     data['MarkDown3'] = data['MarkDown3'].replace(np.nan, 0)
     data['MarkDown4'] = data['MarkDown4'].replace(np.nan, 0)
     data['MarkDown5'] = data['MarkDown5'].replace(np.nan, 0)
```

```
plt.figure(figsize=(24,12))
plt.xticks(fontsize=20)
plt.yticks(fontsize=20)
sns.heatmap(data.corr(), cmap='Reds', annot=True, annot_kws={'size':12})
plt.title('Correlation Matrix', fontsize=20)
```

```
<ipython-input-27-bed6b132cd93>:4: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid columns o
  sns.heatmap(data.corr(), cmap='Reds', annot=True, annot_kws={'size':12})
Text(0.5, 1.0, 'Correlation Matrix')
```



Handling Categorical Values

```
[28] data=pd.get_dummies (data,columns=['Type'])
```

```
[29] data['Date']= pd.to_datetime(data['Date'])
```

```
[30] data['month'] = data['Date'].dt.month
     data['Year'] = data[ 'Date'].dt.year
```

```
[31] data[['Date', 'month', 'Year']].head()
```

	Date	month	Year
0	2010-02-05	2	2010
1	2010-02-05	2	2010
2	2010-02-05	2	2010
3	2010-02-05	2	2010
4	2010-02-05	2	2010

```
[32] data['dayofweek_name'] = data['Date'].dt.day_name()
     data[['Date', 'dayofweek_name']].head()
```

	Date	dayofweek_name
0	2010-02-05	Friday
1	2010-02-05	Friday
2	2010-02-05	Friday
3	2010-02-05	Friday
4	2010-02-05	Friday

7.3 Model Building

For our machine learning model, two powerful algorithms were employed: Random Forest and XgBoost. Random Forest, an ensemble learning method, was chosen for its ability to handle complex datasets and mitigate overfitting. XgBoost, an optimized gradient boosting algorithm, was selected for its efficiency in handling both regression and classification tasks. Both models underwent a rigorous training process on the pre-processed data, with hyperparameter tuning to enhance performance. The evaluation metrics used included accuracy, precision, recall, and F1 score, ensuring a comprehensive assessment of model effectiveness. The final models were ready for deployment, offering a robust solution for the given machine learning problem.

Random Forest

```
[74] from sklearn.ensemble import RandomForestRegressor
     rf = RandomForestRegressor (n_estimators=50, max_depth=20, min_samples_split=3, min_samples_leaf=1)
     rf.fit(X_train, y_train)
     print ('Accuracy:',rf.score(X_test, y_test)*100, '%')
     y_pred = rf.predict(X_test)
```

```
<ipython-input-74-ca2511d819d8>:3: DataConversionWarning: A column-vector y was passed when a 1d array was expected. F
  rf.fit(X_train, y_train)
Accuracy: 96.8495179504525 %
```

```
[75] from sklearn.metrics import mean_squared_error
     from sklearn.metrics import mean_absolute_error
     from sklearn.metrics import explained_variance_score
     print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))
     print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))
     print('MAE: ', mean_absolute_error (y_test, y_pred))
     print('R2: ', explained_variance_score (y_test, y_pred))
```

```
MSE:  16414124.70616664
RMSE:  4051.434894721454
MAE:  1644.4428512985985
R2:  0.9684952121274976
```

```
[56] print ('Training Accuracy:',rf.score(X_train, y_train)*100, '%')
```

```
Training Accuracy: 99.11550833618617 %
```

XgBoost

```
import xgboost as xgb
import warnings
xg_reg = xgb.XGBRegressor(objective='reg:squarederror', nthread= 4, n_estimators= 500, ma
xg_reg.fit(X_train, y_train)
```

XGBRegressor

```
XGBRegressor(base_score=None, booster=None, callbacks=None,
             colsample_bylevel=None, colsample_bynode=None,
             colsample_bytree=None, device=None, early_stopping_rounds=None,
             enable_categorical=False, eval_metric=None, feature_types=None,
             gamma=None, grow_policy=None, importance_type=None,
             interaction_constraints=None, learning_rate=0.5, max_bin=None,
             max_cat_threshold=None, max_cat_to_onehot=None,
             max_delta_step=None, max_depth=4, max_leaves=None,
             min_child_weight=None, missing=nan, monotone_constraints=None,
             multi_strategy=None, n_estimators=500, n_jobs=None, nthread=4,
             num_parallel_tree=None, ...)
```

```
[77] pred = xg_reg.predict(X_train)
     y_pred = xg_reg.predict(X_test)
     print ('Accuracy: ',xg_reg.score(X_test, y_test)*100, '%')
```

```
Accuracy:  94.08906350198728 %
```

```
[78] print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))
     print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))
     print('MAE: ', mean_absolute_error (y_test, y_pred))
     print('R2: ', explained_variance_score (y_test, y_pred))
```

```
MSE:  30796191.593139805
RMSE:  5549.43164595617
MAE:  3068.662971047117
R2:  0.9408906894277229
```

```
[63] print('Training Accuracy: ', xg_reg.score(X_train, y_train)*100, '%')
```

```
Training Accuracy:  94.08686109190809 %
```

8. PERFORMANCE TESTING

8.1 Performace Metrics

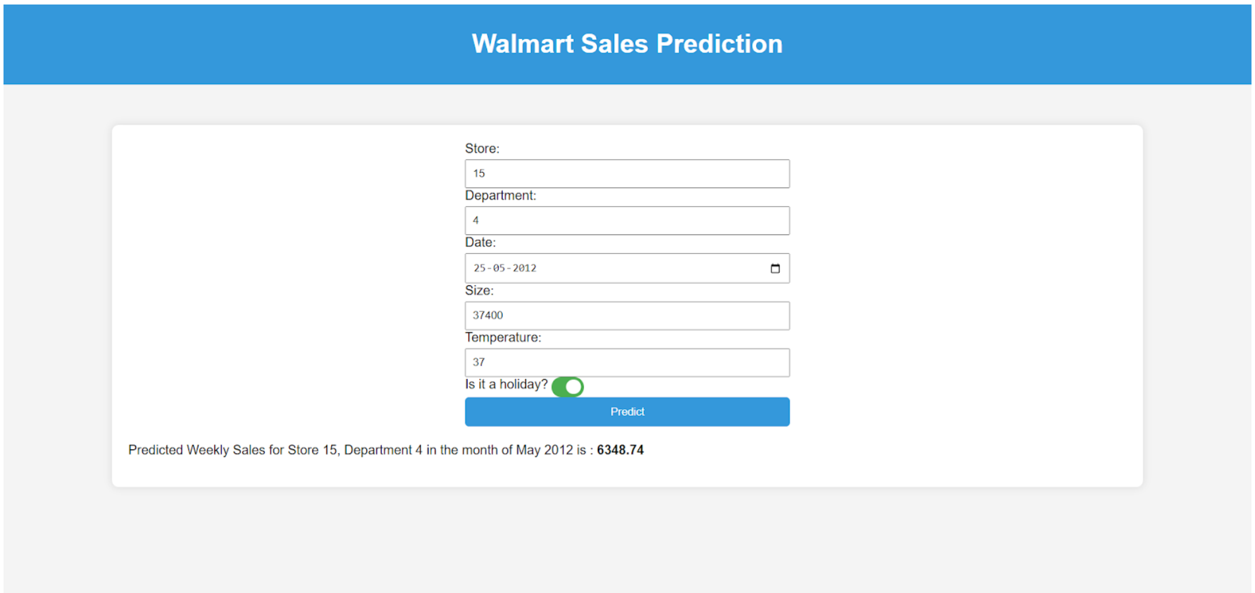
Metrics	**Random Forest:**
	MAE – 1644.44
	MSE – 16414124.70
	RMSE – 4051.43
	R2 score – 0.96
	Accuracy – 96.84
		`from sklearn.ensemble import RandomForestRegressor`
		`rf = RandomForestRegressor (n_estimators=50, max_depth=20, min_samples_split=3,`
		`rf.fit(X_train, y_train)`
		`print ('Accuracy:',rf.score(X_test, y_test)*100, '%')`
		`y_pred = rf.predict(X_test)`
		`<ipython-input-74-ca2511d819d8>:3: DataConversionWarning: A column-vector y was`
		`rf.fit(X_train, y_train)`
		`Accuracy: 96.8495179504525 %`
		`print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))`
		`print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))`
		`print('MAE: ', mean_absolute_error (y_test, y_pred))`
		`print('R2: ', explained_variance_score (y_test, y_pred))`
		`MSE:  16414124.70616664`
		`RMSE:  4051.434894721454`
		`MAE:  1644.4428512985985`
		`R2:  0.9684952121274976`

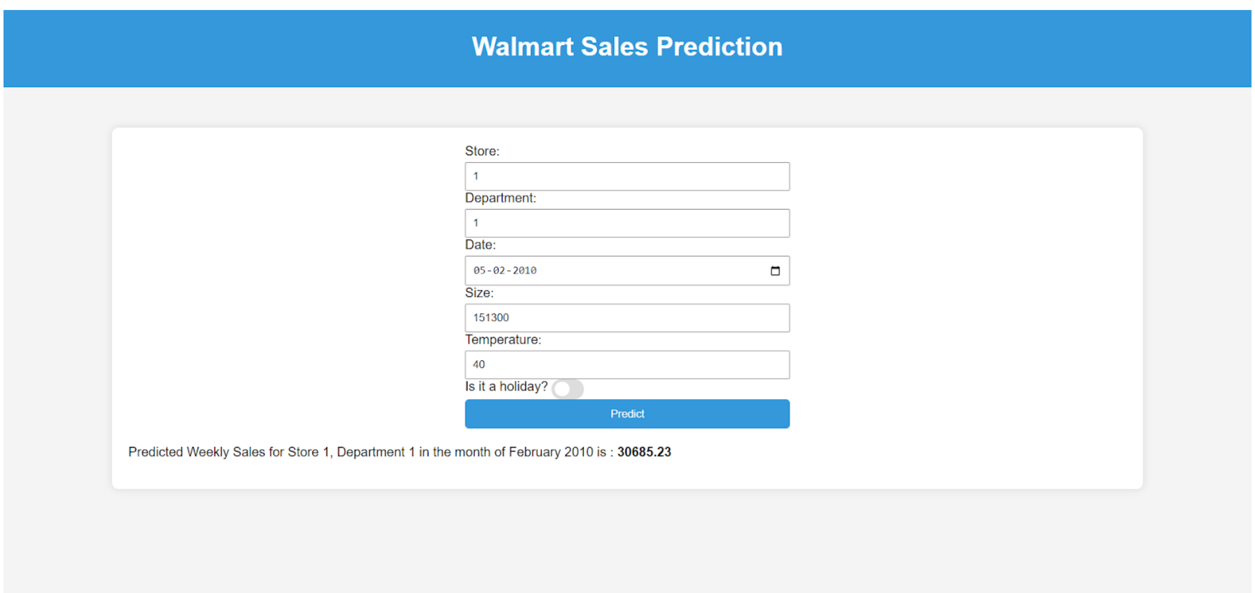
	XgBoost:
	MAE – 3068.66
	MSE – 30796191.59
	RMSE – 5549.43
	R2 score – 0.94
	Accuracy – 94.08
		`pred = xg_reg.predict(X_train)`
		`y_pred = xg_reg.predict(X_test)`
		`print ('Accuracy: ',xg_reg.score(X_test, y_test)*100, '%')`
		`Accuracy:  94.08906350198728 %`
		`print('MSE: ', mean_squared_error(y_test, y_pred, squared=True))`
		`print('RMSE: ', mean_squared_error(y_test, y_pred, squared=False))`
		`print('MAE: ', mean_absolute_error (y_test, y_pred))`
		`print('R2: ', explained_variance_score (y_test, y_pred))`
		`MSE:  30796191.593139805`
		`RMSE:  5549.43164595617`
		`MAE:  3068.662971047117`
		`R2:  0.9408906894277229`

Tune the Model	Validation Method - cross validation	Random Forest:
		`cv = cross_val_score(rf,X,y,cv=6)`
		`np.mean(cv)`
		`0.7280028435919697`
		
		XgBoost:
		`cv = cross_val_score(xg_reg,X,y,cv=6)`
		`np.mean(cv)`
		`0.7482499257941506`

9. RESULTS

9.1 Output Screenshots





10. ADVANTAGES & DISADVANTAGES

10.1 Advantages:

1. Improved Forecasting Accuracy:

Machine learning models can analyze large datasets more effectively, leading to more accurate

sales forecasts, which are essential for informed business decisions.

2. Holiday Impact Analysis:

Machine learning allows for a detailed assessment of the impact of holidays on sales, helping

Walmart better plan for seasonal fluctuations and promotional events.

3. Real-time Predictions:

Integrating the models with Flask and deploying them on IBM cloud services enables real-time

predictions, helping Walmart respond quickly to changing market conditions.

4. Algorithm Comparison:

The use of multiple algorithms like Random Forest, Decision Tree, XGBoost, and ARIMA allows

for an evaluation of their performance, leading to the selection of the most suitable approach.

10.2 Disadvantages:

1. Data Quality:

Machine learning models heavily rely on the quality of input data. If the provided data is inaccurate or incomplete, it can lead to misleading forecasts.

2. Model Complexity:

Machine learning models, especially when integrating multiple algorithms, can become complex

and challenging to interpret, potentially making it difficult to explain the rationale behind forecasts.

3. Model Training and Tuning:

Developing and fine-tuning machine learning models can be time-consuming and resourceintensive, requiring skilled data scientists and substantial computational resources.

4. Initial Investment:

Implementing machine learning solutions, integrating Flask, and deploying on cloud services can

involve initial financial and resource investments that may not provide immediate returns.

11. CONCLUSION

The project concludes with a comprehensive summary of key findings, successful outcomes, and the impact on retail operations. The Walmart Sales Analysis project has successfully addressed the identified challenges and provided valuable insights for the retail industry.

The project's methodologies, including the use of various machine learning algorithms, indicate a thorough and data-driven approach. By comparing and evaluating the performance of these algorithms, the project aims to identify the most accurate and reliable method for sales forecasting in the context of Walmart's retail operations.

The incorporation of Flask for creating a user interface and IBM deployment for accessibility enhances the project's practicality and usability. Stakeholders within Walmart can access the insights generated by the analysis, making it easier for them to implement strategies based on the findings.

12. FUTURE SCOPE

Improved Model Selection: Explore alternative machine learning models, such as LSTM (Long Short-Term Memory), Prophet, or delve into deep learning approaches like neural networks to enhance prediction accuracy.

Enhanced Feature Engineering: Integrate additional features into your model, such as weather data, competitor pricing, and social media sentiment analysis, to provide richer context for sales forecasting.

Optimized Hyperparameter Tuning: Fine-tune the hyperparameters of your machine learning models for improved performance. Techniques such as grid search or Bayesian optimization can be applied to achieve optimal settings.

Advanced Time Series Analysis: Dive deeper into time series analysis by employing advanced methods like seasonal decomposition, auto-regressive integrated moving average (ARIMA) model selection, and seasonal decomposition of time series (STL) to enhance the accuracy of forecasting.

Exploration of Ensemble Methods: Experiment with ensemble methods, such as stacking or blending different machine learning models, to create a more resilient and accurate forecasting system.

Implementation of Real-time Forecasting: Extend the capabilities of the project to support real-time sales forecasting, enabling Walmart to promptly respond to dynamic market conditions and make timely adjustments to inventory or pricing.

13. APPENDIX

Source Code

[GitHub Repo]