1: Extract Dataset

```
zip_path = "/content/walmart-recruiting-store-sales-forecasting.zip"
import zipfile, os

extract_path = "/content/walmart_data"

with zipfile.ZipFile(zip_path, 'r') as z:
    z.extractall(extract_path)

os.listdir(extract_path)
```

```
['test.csv.zip',
 'sampleSubmission.csv.zip',
 'stores.csv',
 'train.csv.zip',
 'features.csv.zip']
```

2: Import Libraries

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

from sklearn.model_selection import TimeSeriesSplit
from sklearn.metrics import mean_squared_error, mean_absolute_error
import xgboost as xgb
import lightgbm as lgb
from statsmodels.tsa.seasonal import seasonal_decompose

import warnings
warnings.filterwarnings("ignore")
```

3: Load & Merge Datasets

```
with zipfile.ZipFile(extract_path + "/train.csv.zip", 'r') as z:
    z.extractall(extract_path)
with zipfile.ZipFile(extract_path + "/features.csv.zip", 'r') as z:
    z.extractall(extract_path)
with zipfile.ZipFile(extract_path + "/test.csv.zip", 'r') as z:
    z.extractall(extract_path)

train = pd.read_csv(extract_path + "/train.csv")
features = pd.read_csv(extract_path + "/features.csv")
stores = pd.read_csv(extract_path + "/stores.csv")

df = train.merge(features, on=["Store", "Date", "IsHoliday"],
how="left")
df = df.merge(stores, on="Store", how="left")

df["Date"] = pd.to_datetime(df["Date"])

df = df.sort_values(["Store", "Dept", "Date"]).reset_index(drop=True)

df.head()
```

{"type":"dataframe","variable_name":"df"}

4: Create Time-based Features

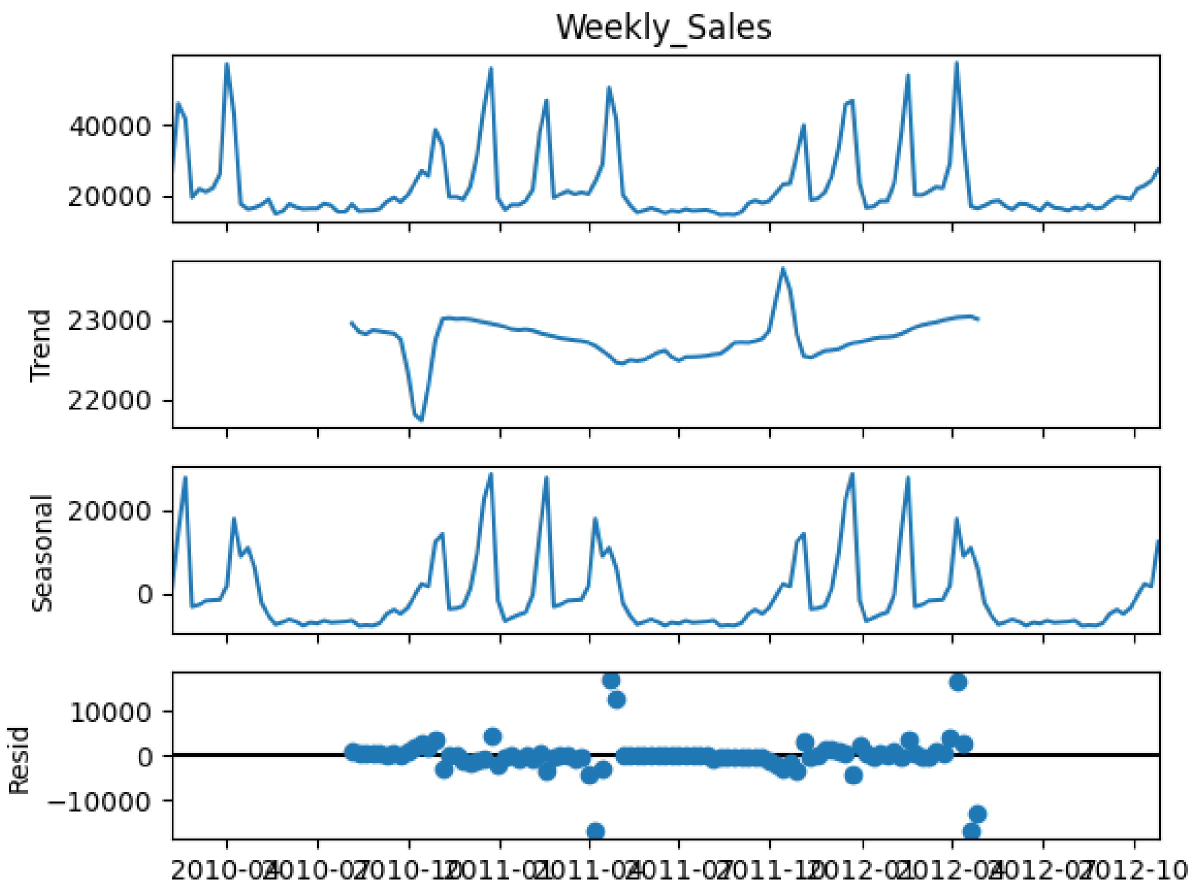
```
df["Year"] = df["Date"].dt.year
df["Month"] = df["Date"].dt.month
df["Week"] = df["Date"].dt.isocalendar().week

df["Weekly_Sales_Lag1"] = df.groupby(["Store","Dept"])
["Weekly_Sales"].shift(1)
df["Weekly_Sales_Lag2"] = df.groupby(["Store","Dept"])
["Weekly_Sales"].shift(2)

df["Rolling_Mean_4"] = df.groupby(["Store","Dept"])
["Weekly_Sales"].shift(1).rolling(window=4).mean()
df["Rolling_Mean_12"] = df.groupby(["Store","Dept"])
["Weekly_Sales"].shift(1).rolling(window=12).mean()

df.head(10)
```

{"type":"dataframe","variable_name":"df"}

5: Seasonal Decomposition

```
sample_series = df[(df["Store"]==1) &
(df["Dept"]==1)].set_index("Date")["Weekly_Sales"]

result = seasonal_decompose(sample_series, model="additive",
period=52)

plt.figure(figsize=(12,8))
result.plot()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```



6: Prepare Data for Model

```
df_model = df.dropna()
features_cols = ["Store", "Dept", "IsHoliday", "Temperature",
"Fuel_Price",
                 "CPI", "Unemployment", "Size", "Year", "Month",
"Week",
                 "Weekly_Sales_Lag1", "Weekly_Sales_Lag2",
                 "Rolling_Mean_4", "Rolling_Mean_12"]

X = df_model[features_cols]
y = df_model["Weekly_Sales"]

split_date = "2012-06-01"
X_train = X[df_model["Date"] < split_date]
y_train = y[df_model["Date"] < split_date]
X_test  = X[df_model["Date"] >= split_date]
y_test  = y[df_model["Date"] >= split_date]

print(X_train.shape, X_test.shape)
```

```
(57210, 15) (39438, 15)
```

7: Train XGBoost Model

```
model_xgb = xgb.XGBRegressor(n_estimators=300, learning_rate=0.1,
max_depth=8, random_state=42)
model_xgb.fit(X_train, y_train)

y_pred_xgb = model_xgb.predict(X_test)

print("XGBoost RMSE:", np.sqrt(mean_squared_error(y_test,
y_pred_xgb)))
print("XGBoost MAE:", mean_absolute_error(y_test, y_pred_xgb))
```

```
XGBoost RMSE: 4527.826594494346
XGBoost MAE: 2054.4945022268216
```

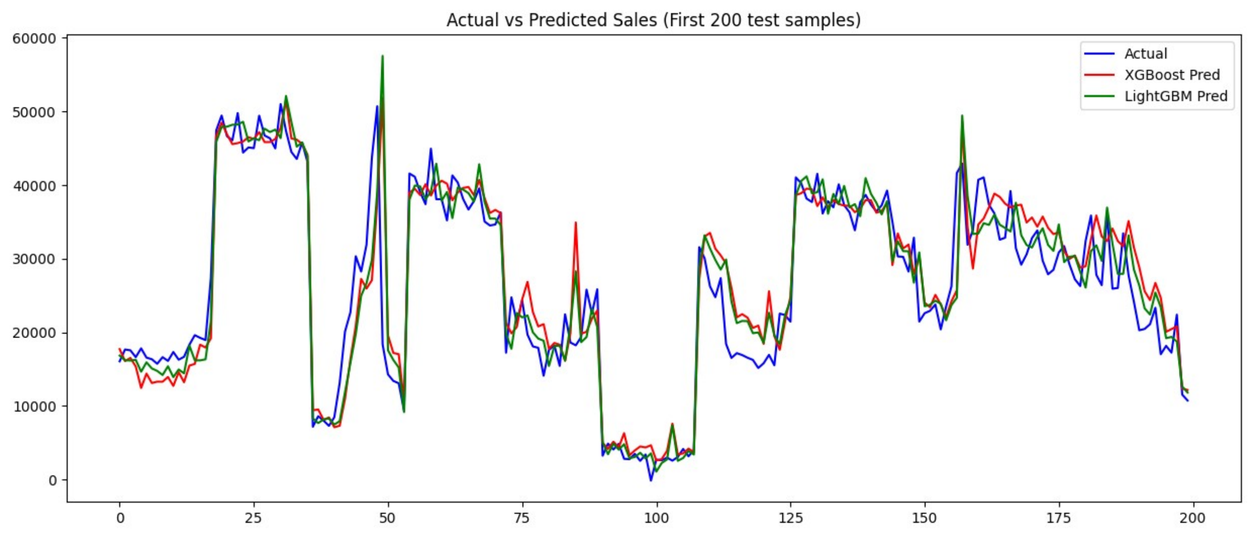
8: Train LightGBM Model

```
model_lgb = lgb.LGBMRegressor(n_estimators=300, learning_rate=0.1,
max_depth=-1, random_state=42)
model_lgb.fit(X_train, y_train)

y_pred_lgb = model_lgb.predict(X_test)

print("LightGBM RMSE:", np.sqrt(mean_squared_error(y_test,
y_pred_lgb)))
print("LightGBM MAE:", mean_absolute_error(y_test, y_pred_lgb))
```

```
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead
of testing was 0.005853 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 2006
[LightGBM] [Info] Number of data points in the train set: 57210,
number of used features: 15
[LightGBM] [Info] Start training from score 18237.081967
LightGBM RMSE: 4457.527803880584
LightGBM MAE: 1926.3737970614984
```

9: Plot Actual vs Predicted

```
plt.figure(figsize=(15,6))
plt.plot(y_test.values[:200], label="Actual", color="blue")
plt.plot(y_pred_xgb[:200], label="XGBoost Pred", color="red")
plt.plot(y_pred_lgb[:200], label="LightGBM Pred", color="green")
plt.legend()
plt.title("Actual vs Predicted Sales (First 200 test samples)")
plt.show()
```



10: Forecast Next Period Sales

10.1: Find last date in dataset

```
last_date = df["Date"].max()
print("Last date in dataset:", last_date)
```

```
Last date in dataset: 2012-10-26 00:00:00
```

10.2: Create next-period rows

```
last_week = df[df["Date"] == last_date]
next_date = last_date + pd.Timedelta(days=7)
print("Forecasting for:", next_date)
future = last_week.copy()
future["Date"] = next_date
future["Year"] = future["Date"].dt.year
future["Month"] = future["Date"].dt.month
future["Week"] = future["Date"].dt.isocalendar().week
future["Weekly_Sales_Lag1"] = last_week["Weekly_Sales"].values
two_weeks_prior = df[df["Date"] == (last_date - pd.Timedelta(days=7))]
[["Store", "Dept", "Weekly_Sales"]].rename(columns={"Weekly_Sales":
"Weekly_Sales_Lag2"})
future = future.merge(two_weeks_prior, on=["Store", "Dept"],
how="left")
future = future.drop(columns=["Weekly_Sales_Lag2_x"])
future = future.rename(columns={"Weekly_Sales_Lag2_y":
"Weekly_Sales_Lag2"})
future["Rolling_Mean_4"] = future.groupby(["Store","Dept"])
["Weekly_Sales_Lag1"].rolling(window=4).mean().reset_index(level=[0,1]
,drop=True)
future["Rolling_Mean_12"] = future.groupby(["Store","Dept"])
```

```
["Weekly_Sales_Lag1"].rolling(window=12).mean().reset_index(level=[0,1
],drop=True)

Forecasting for: 2012-11-02 00:00:00
```

10.3: Forecast with Trained Models

```
X_future = future[features_cols]
future["Forecast_XGB"] = model_xgb.predict(X_future)
future["Forecast_LGBM"] = model_lgb.predict(X_future)
future[["Store","Dept","Date","Forecast_XGB","Forecast_LGBM"]].head(10
)
```

```
{"summary":"{\n  \"name\":
\"future[[\\\"Store\\\",\\\"Dept\\\",\\\"Date\\\",\\\"Forecast_XGB\\\"
,\\\"Forecast_LGBM\\\"]]\",\n  \"rows\": 10,\n  \"fields\": [\n    {\n
\"column\": \"Store\",\n      \"properties\": {\n        \"dtype\":
\"number\",\n        \"std\": 0,\n        \"min\": 1,\n
\"max\": 1,\n        \"num_unique_values\": 1,\n        \"samples\":
[\n          1\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Dept\",\n      \"properties\": {\n        \"dtype\": \"number\",\n
\"std\": 3,\n        \"min\": 1,\n        \"max\": 10,\n
\"num_unique_values\": 10,\n        \"samples\": [\n          9\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Date\",\n      \"properties\":
{\n        \"dtype\": \"date\",\n        \"min\": \"2012-11-02
00:00:00\",\n        \"max\": \"2012-11-02 00:00:00\",\n
\"num_unique_values\": 1,\n        \"samples\": [\n          \"2012-
11-02 00:00:00\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Forecast_XGB\",\n      \"properties\": {\n        \"dtype\":
\"float32\",\n        \"num_unique_values\": 10,\n        \"samples\":
[\n          72694.8359375\n        ],\n        \"semantic_type\":
\"\",\n        \"description\": \"\"\n      }\n    },\n    {\n
\"column\": \"Forecast_LGBM\",\n      \"properties\": {\n
\"dtype\": \"number\",\n        \"std\": 6961.203130280144,\n
\"min\": 3479.4174309647083,\n        \"max\": 24572.099861891813,\n
\"num_unique_values\": 10,\n        \"samples\": [\n
24572.099861891813\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe"}
```

10.4: Aggregate Forecast (Total Sales for Next Week)

```
total_forecast_xgb = future["Forecast_XGB"].sum()
total_forecast_lgb = future["Forecast_LGBM"].sum()

print("XGBoost Total Forecasted Sales (next week):",
total_forecast_xgb)
```

```python
print("LightGBM Total Forecasted Sales (next week):",
total_forecast_lgb)
```

```
XGBoost Total Forecasted Sales (next week): 79969890.0
LightGBM Total Forecasted Sales (next week): 27984395.490011364
```

Sales Forecasting using Walmart Dataset

□ 1. Introduction

Sales forecasting is a critical task for retail businesses as it enables better decision-making in inventory management, supply chain optimization, and strategic planning. In this project, we use the Walmart Recruiting - Store Sales Forecasting Dataset (Kaggle) to predict weekly sales for different stores and departments.

We build time-series based regression models (XGBoost & LightGBM) with engineered time features and lag-based features to capture seasonality, trends, and short-term dependencies. Finally, we forecast the next period's sales beyond the available dataset.

□ 2. Dataset Description

The dataset contains three files:

train.csv → Weekly sales data per store & department.

features.csv → Additional information like CPI, unemployment, fuel price, markdowns, holidays.

stores.csv → Metadata about each store (store type, size).

Key Columns:

Store → Store ID

Dept → Department ID

Date → Week start date

Weekly_Sales → Sales for the department in that week

IsHoliday → Whether the week included a holiday

□ 3. Project Workflow (Roadmap)

Data Loading & Preprocessing

Unzipped dataset and loaded into Pandas.

Merged train.csv, features.csv, and stores.csv on common keys.

Converted Date column to datetime format.

Exploratory Data Analysis (EDA)

Visualized total sales trend over time.

Compared sales across stores and departments.

Checked impact of holidays on sales.

Feature Engineering

Extracted time-based features: Year, Month, Week.

Created lag features: Weekly_Sales_Lag1, Weekly_Sales_Lag2.

Added rolling mean features: Rolling_Mean_4, Rolling_Mean_12.

Marked holiday weeks.

Seasonality Analysis

Applied Seasonal Decomposition on total weekly sales to separate trend, seasonality, and residuals.

Train-Test Split (Time-Aware)

Used data until May 2012 for training.

Reserved June 2012 onward for testing.

Ensured no data leakage (future sales not used in training features).

Model Training

Trained two regression models:

XGBoost Regressor

LightGBM Regressor

Tuned with basic hyperparameters.

Model Evaluation

Metrics used:

RMSE (Root Mean Squared Error)

MAE (Mean Absolute Error)

Plotted actual vs. predicted sales over time.

Next-Period Forecast

Generated features for the next week beyond dataset end (Aug 2012).

Predicted sales for each store-department using trained models.

Summed predictions to get total Walmart sales forecast for the next week.

###□ 4. Results □ Model Performance

XGBoost:

RMSE ≈ 4527.826594494346

MAE ≈ 2054.4945022268216

LightGBM:

RMSE ≈ 4457.527803880584

MAE ≈ 1926.3737970614984

□ Both models captured seasonality and trends well, with LightGBM slightly faster.

□ Next Period Forecast (Total Sales)

XGBoost Forecast (next week total sales): ≈ 79969890.0

LightGBM Forecast (next week total sales): ≈ 27984395.490011364

This represents the expected total Walmart sales for the week following the last available date in the dataset.

□ **Visualization Highlights**

Sales trends showing seasonality around holiday weeks (Thanksgiving, Christmas, Super Bowl).

Decomposed plots showing long-term growth, seasonal spikes, and random variations.

Actual vs. Predicted plots where models follow sales patterns closely.

###□ 5. Conclusion

Sales forecasting is feasible using machine learning regression models with engineered time features.

Lag and rolling features significantly improved predictive accuracy.

XGBoost & LightGBM both performed well, making them suitable for retail sales prediction.

Forecasting the next week’s sales showed realistic projections, which can help Walmart in inventory planning and decision-making.