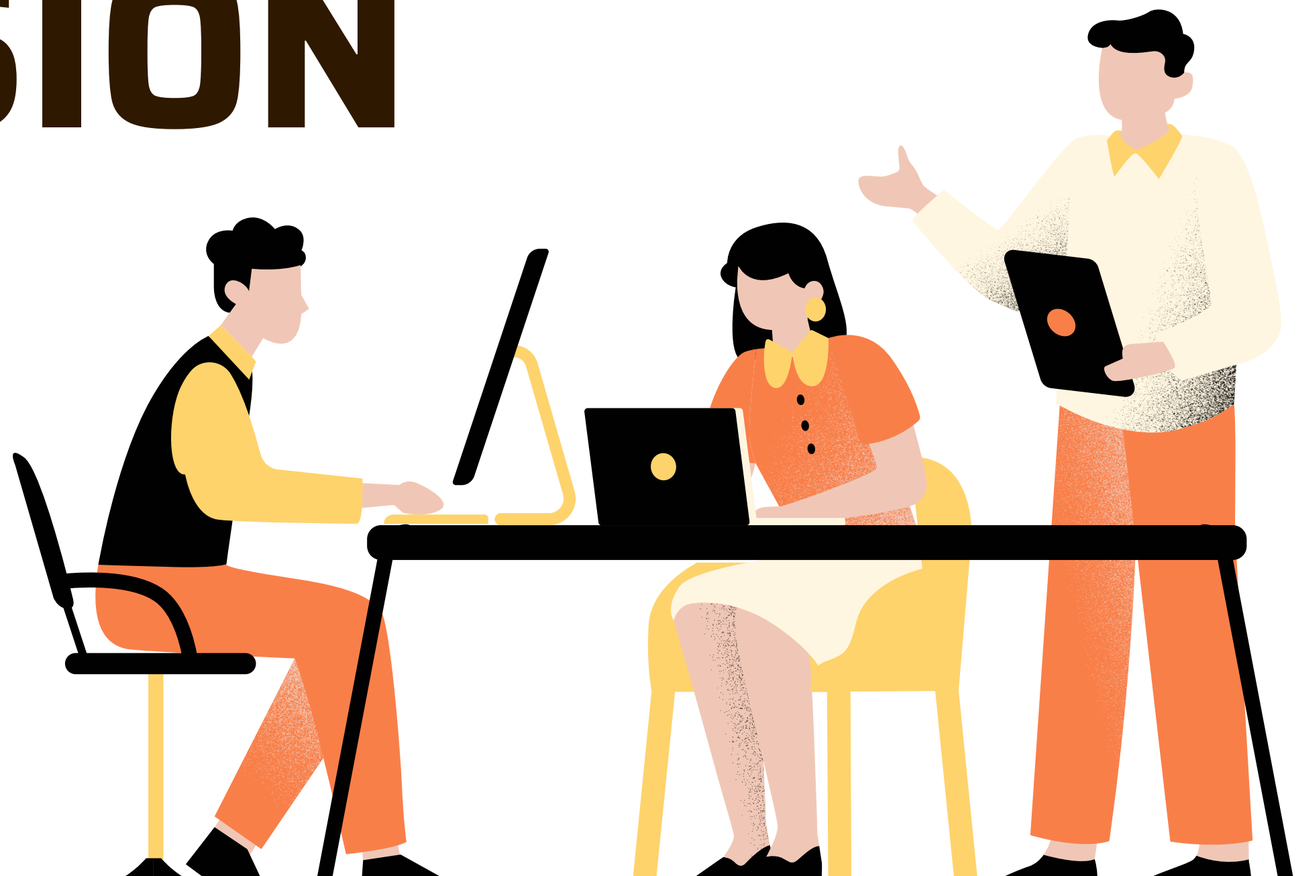
SAUDI DIGITAL ACADEMY
AI Bootcamp

MINI PROJECT (1)

REGRESSION WITH PYTHON

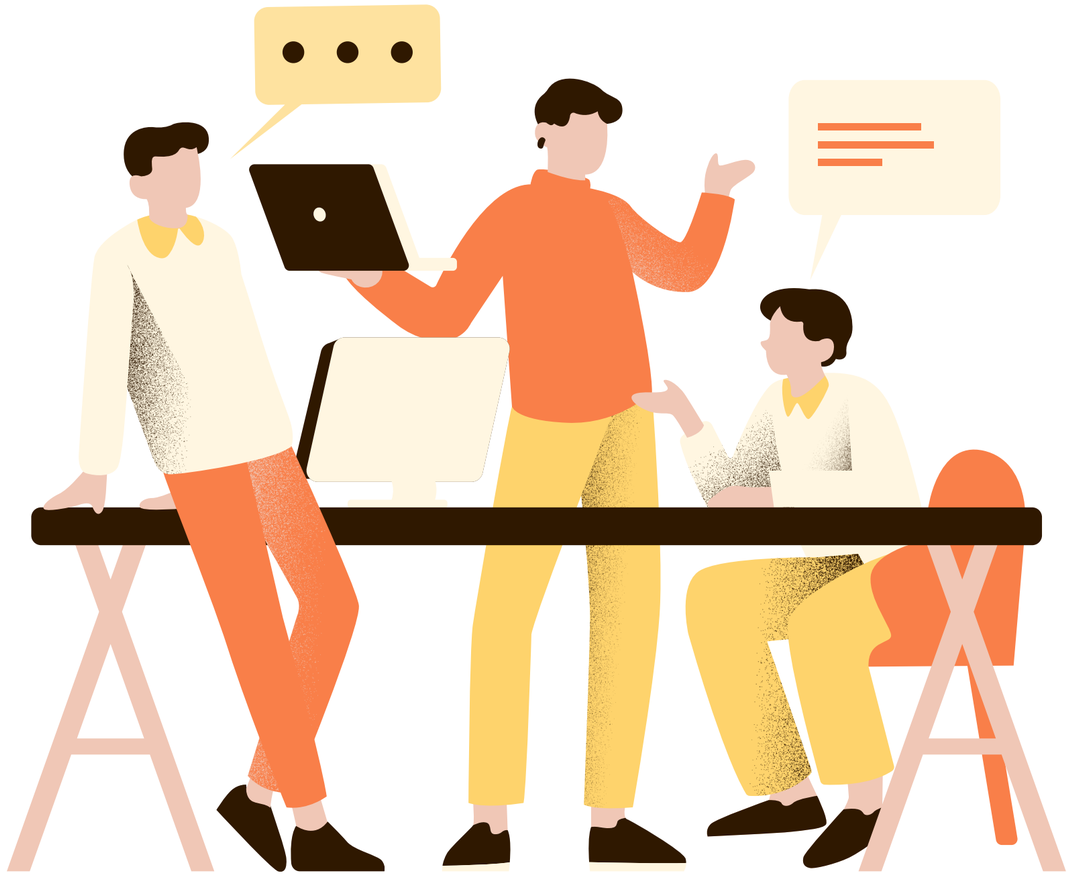


PRESENT BY

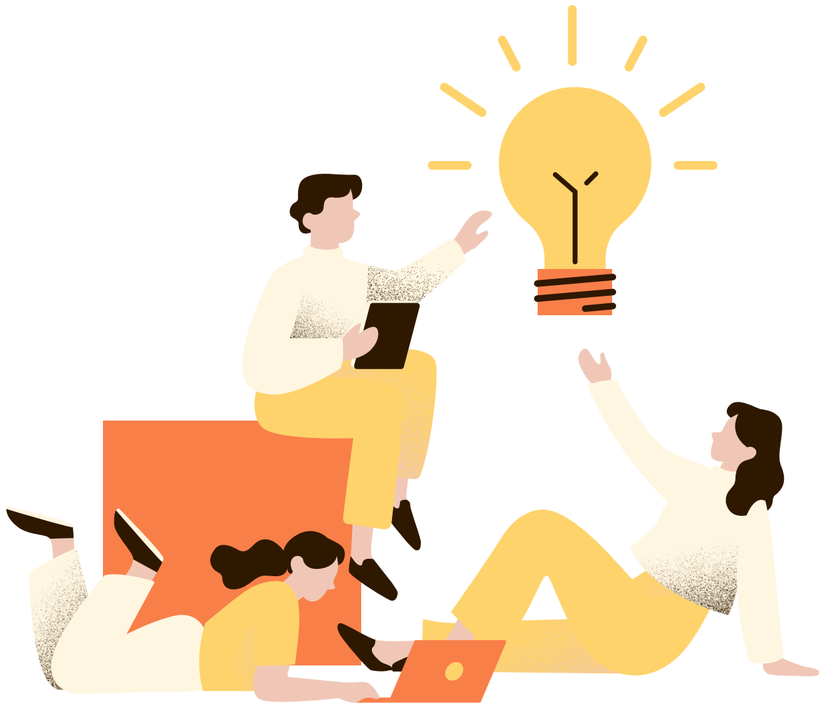
GROUP NUMBER: 5

TEAM MEMBERS

- HANAN ALNBHANI
- RAZAN BAHHARI
- SHATHA HUWAYKIM
- WAFA ALQHTANI



PROJECT SOLUTION STEPS



1. UNDERSTAND DATASET
2. PRE-PROCESS THE DATA
3. SPLIT DATA INTO TRAINING AND TESTING SETS
4. CHOOSE A SUPERVISED LEARNING MODEL
5. EVALUATE THE MODEL
6. MPROVE THE MODEL
7. MAKE PREDICTIONS ON NEW DATA

SOLUTION STEPS

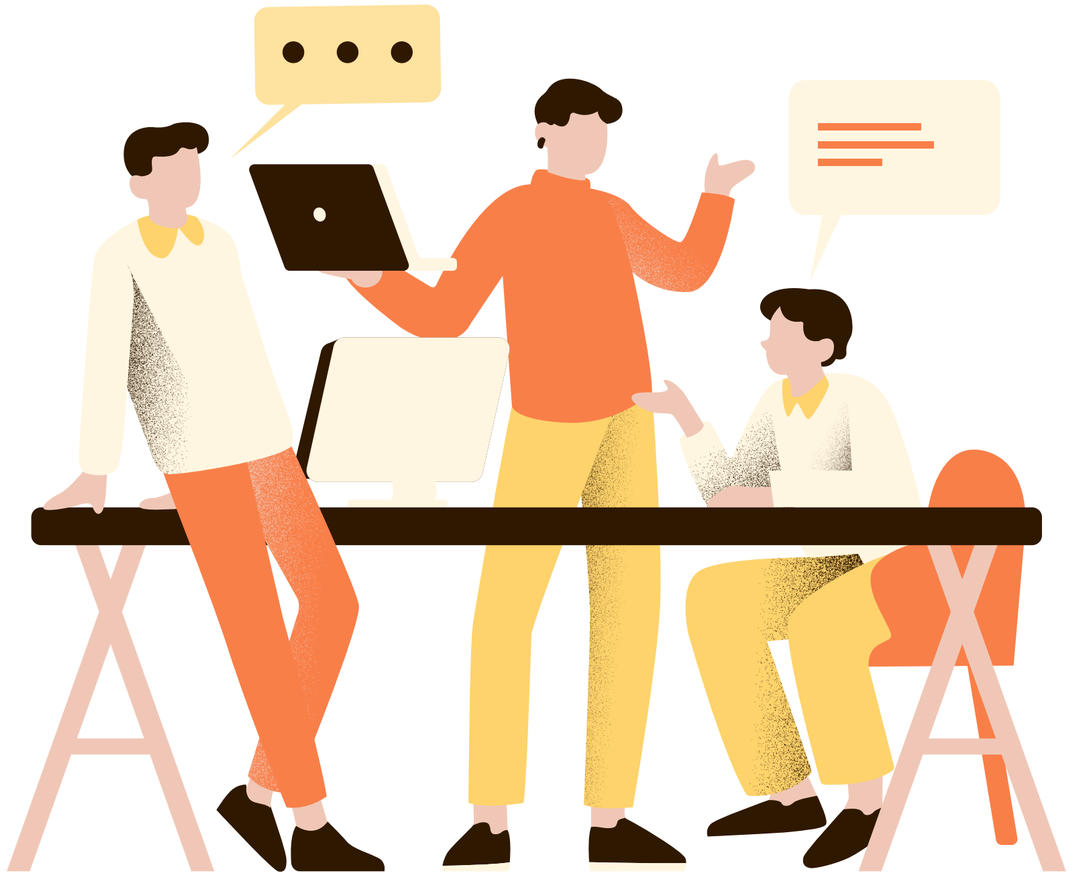
1. UNDERSTAND DATASET

SALES

sales

REAL DATA

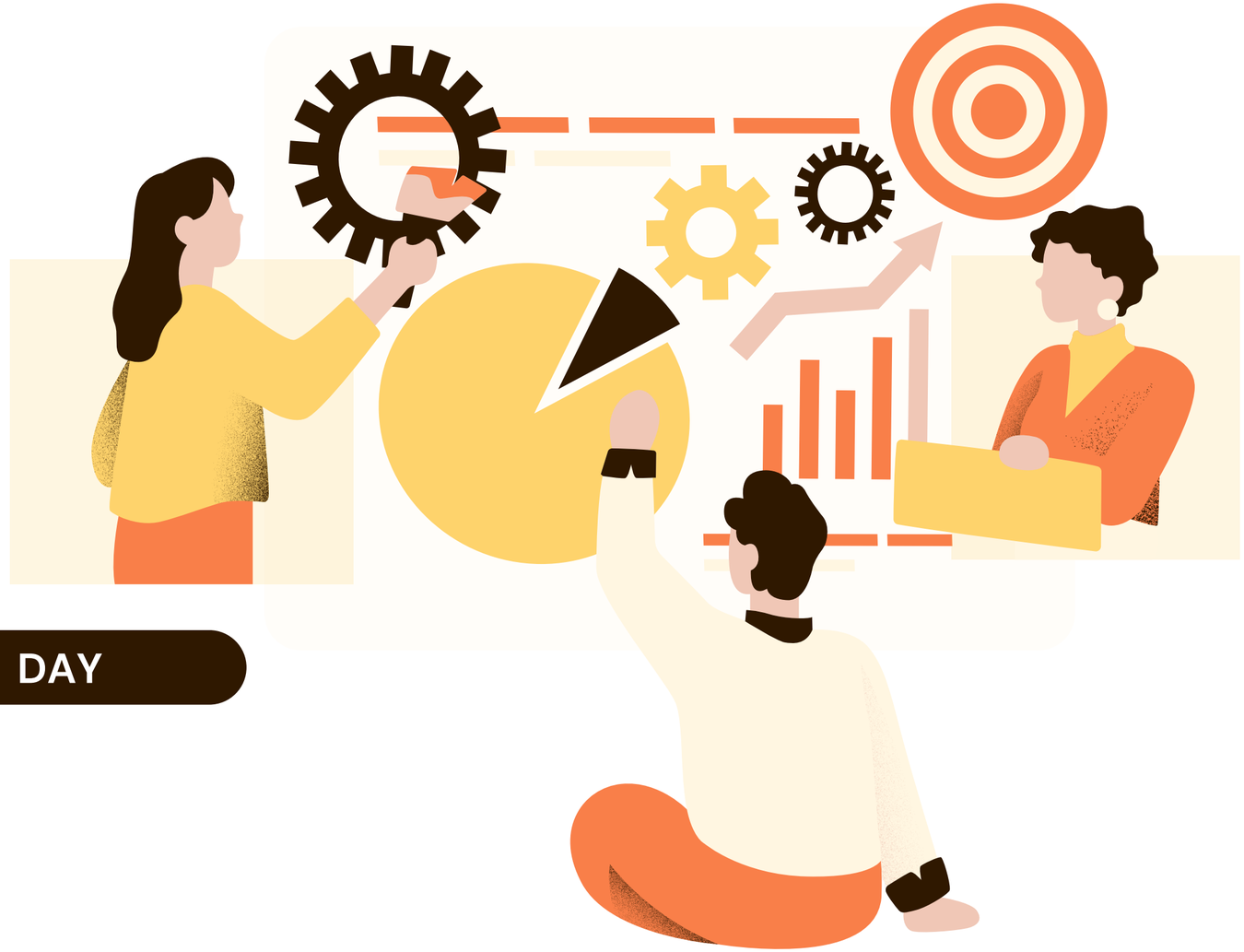
REAL_DATA



SOLUTION STEPS

2. PRE – PROCESS THE DATA

Convert dates into useful features like Year, Month, Day, handle missing values, and normalize/scale numerical features if needed.



YEAR — MONTH — DAY

day	month	year
18	4	2013
11	4	2015
29	8	2013
28	5	2013

SOLUTION STEPS

2. PRE-PROCESS THE DATA

```python
# Convert 'date' column to datetime
sales_p['date'] = pd.to_datetime(sales_p['date'])

# Extract date features
sales_p['day'] = sales_p['date'].dt.day
sales_p['month'] = sales_p['date'].dt.month
sales_p['year'] = sales_p['date'].dt.year
sales_p['DayOfWeek'] = sales_p['date'].dt.isocalendar().week

# Drop original date column
sales_p = sales_p.drop(columns=['date'])
sales_p

# Handle missing values (fill with median)
#sales_p.fillna(inplace=True)
```

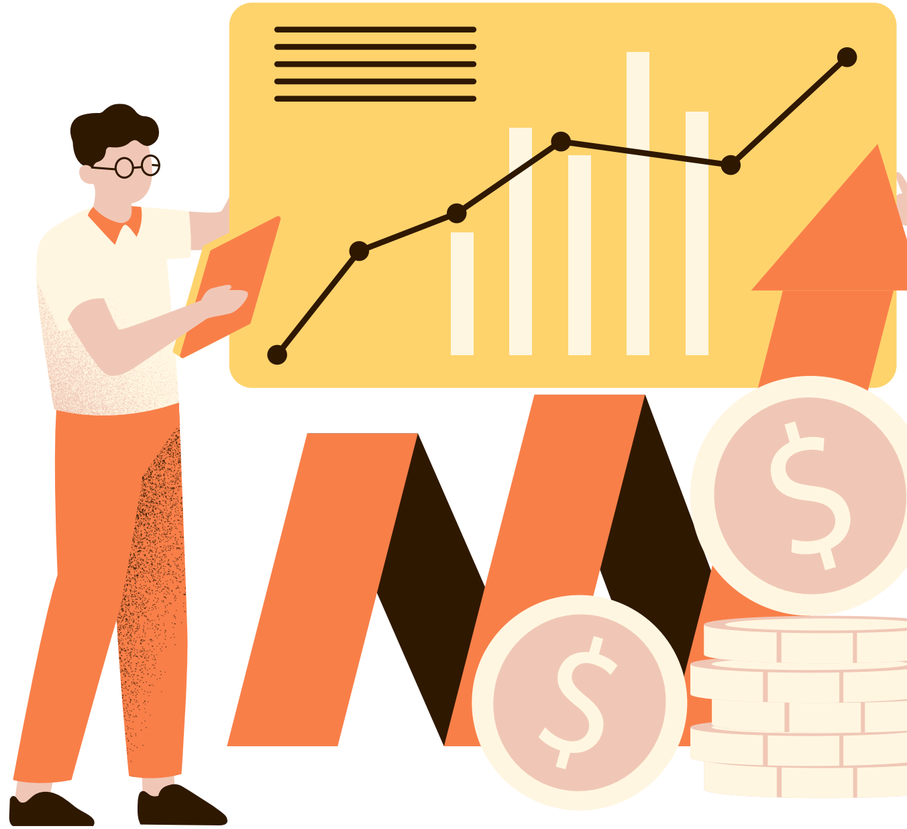
```python
# Check for missing values
print(sales_p.isnull().sum())
```

```
Unnamed: 0            0
store_ID              0
day_of_week           0
date                  0
nb_customers_on_day   0
open                  0
promotion             0
state_holiday         0
school_holiday        0
sales                 0
dtype: int64
```

Python

	Unnamed: 0	store_ID	day_of_week	nb_customers_on_day	open	promotion	state_holiday	school_holiday	sales	day	month	year
0	425390	366	4	517	1	0	0	0	4422	18	4	2013
1	291687	394	6	694	1	0	0	0	8297	11	4	2015
2	411278	807	4	970	1	1	0	0	9729	29	8	2013
3	664714	802	2	473	1	1	0	0	6513	28	5	2013
4	540835	726	4	1068	1	1	0	0	10882	10	10	2013
...	...	...	...	...	...	...	...	...	...	...	...	...
640835	359783	409	6	483	1	0	0	0	4553	26	10	2013
640836	152315	97	1	987	1	1	0	0	12307	14	4	2014
640837	117952	987	1	925	1	0	0	0	6800	7	7	2014
640838	435829	1084	4	725	1	0	0	0	5344	12	6	2014
640839	305711	695	7	0	0	0	0	0	0	3	5	2015

640840 rows × 13 columns



SOLUTION STEPS

3. SPLIT DATA INTO TRAINING AND TESTING SETS

```
# train the model:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression

# Define Features (X) and Target (y)
X = sales_p.drop(columns=['sales'])
y = sales_p['sales']

# Split into train/test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train the model
model = LinearRegression()
model.fit(X_train, y_train)
```

[21]

```
▾ LinearRegression
LinearRegression()
```

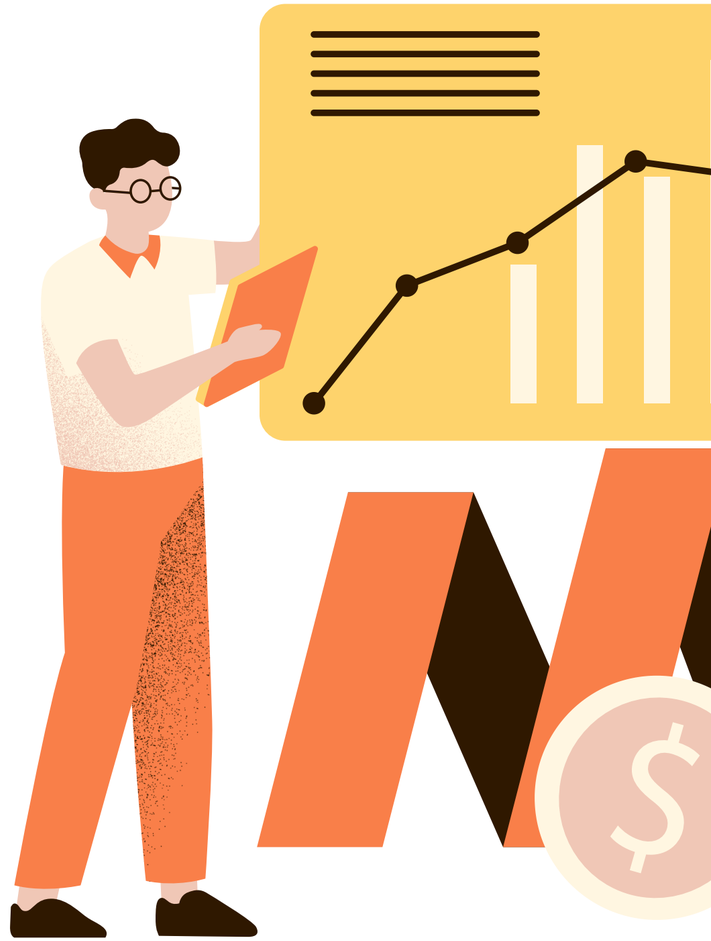
```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Make predictions on the test set
y_pred = model.predict(X_test)

# Calculate evaluation metrics
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print results
print(f'Mean Absolute Error (MAE): {mae:.2f}')
print(f'Mean Squared Error (MSE): {mse:.2f}')
print(f'R² Score (test): {r2:.2f}')

# make predections on the train set:
y_test_pred=model.predict(X_train)
```



SOLUTION STEPS

DATA PREPROCESSING FOR REAL DATA DATASET

```
print(sales_p.dtypes)
```

[19]

```
Unnamed: 0              int64
store_ID                int64
day_of_week             int64
nb_customers_on_day     int64
open                    int64
promotion               int64
state_holiday           int64
school_holiday          int64
sales                   int64
day                     int32
month                   int32
year                    int32
DayOfWeek              UInt32
dtype: object
```

data preprocessing for real data dataset:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()


encoder = LabelEncoder()
real_d["state_holiday"] = encoder.fit_transform(real_d["state_holiday"])
real_d["school_holiday"] = encoder.fit_transform(real_d["school_holiday"])
```

[20]

```
# train the model:
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
```

[21]

SOLUTION STEPS

CHECK MISSING VALUES FOR SALES:

+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···

check missing values for sales:

```
# Check for missing values
print(sales_p.isnull().sum())
```

[12]

```
Unnamed: 0              0
store_ID                0
day_of_week             0
date                    0
nb_customers_on_day     0
open                    0
promotion               0
state_holiday           0
school_holiday          0
sales                   0
dtype: int64
```

extract features from sales dataset:

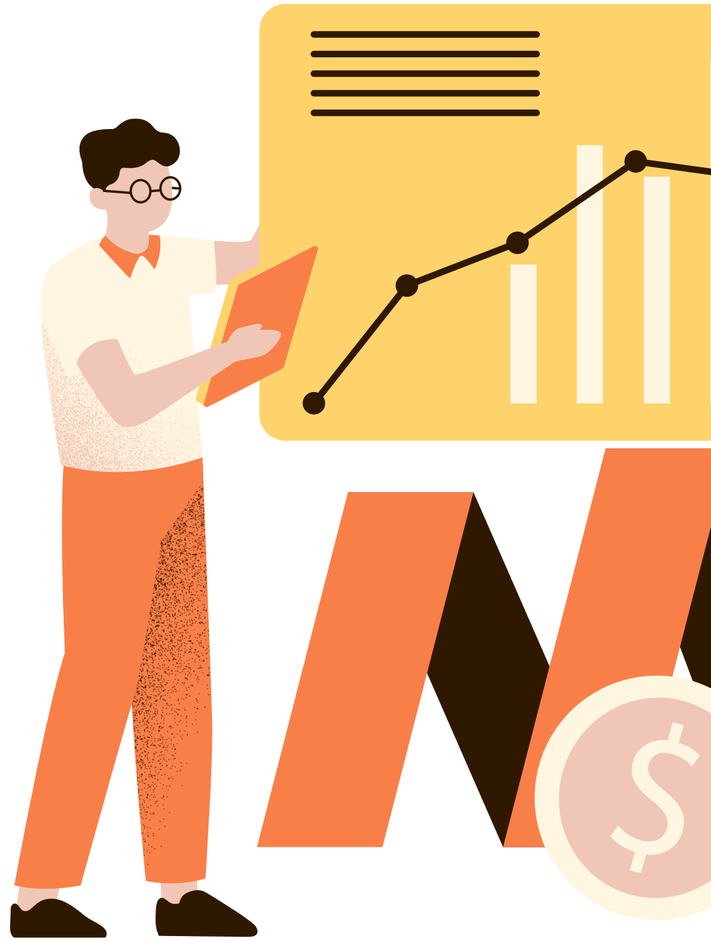
```
# Convert 'date' column to datetime
sales_p['date'] = pd.to_datetime(sales_p['date'])

# Extract date features
sales_p['day'] = sales_p['date'].dt.day
sales_p['month'] = sales_p['date'].dt.month
sales_p['year'] = sales_p['date'].dt.year
sales_p['DayOfWeek'] = sales_p['date'].dt.isocalendar().week

# Drop original date column
sales_p = sales_p.drop(columns=['date'])
sales_p

# Handle missing values (fill with median)
#sales_p.fillna(inplace=True)
```

[13]



SOLUTION STEPS

EXTRACT FEATURES FROM SALES DATASET:

+ Code + Markdown | Run All Restart Clear All Outputs | Jupyter Variables Outline ···

check missing values for sales:

```
# Check for missing values
print(sales_p.isnull().sum())
```

[12]

```
Unnamed: 0              0
store_ID                0
day_of_week             0
date                    0
nb_customers_on_day     0
open                    0
promotion               0
state_holiday           0
school_holiday          0
sales                   0
dtype: int64
```

extract features from sales dataset:

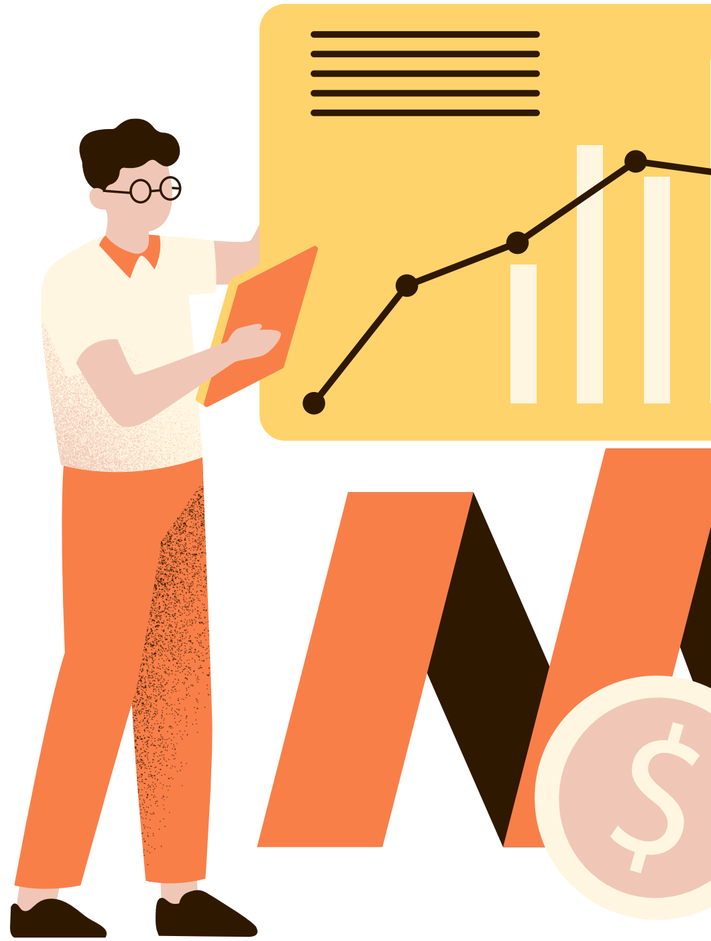
```
# Convert 'date' column to datetime
sales_p['date'] = pd.to_datetime(sales_p['date'])

# Extract date features
sales_p['day'] = sales_p['date'].dt.day
sales_p['month'] = sales_p['date'].dt.month
sales_p['year'] = sales_p['date'].dt.year
sales_p['DayOfWeek'] = sales_p['date'].dt.isocalendar().week

# Drop original date column
sales_p = sales_p.drop(columns=['date'])
sales_p

# Handle missing values (fill with median)
#sales_p.fillna(inplace=True)
```

[13]



SOLUTION STEPS

mini_project1.ipynb > real_d

Code + Markdown | Run All | Restart | Clear All Outputs | Jupyter Variables | Outline ··· Python 3.11.9

```python
# Convert 'date' column to datetime
sales_p['date'] = pd.to_datetime(sales_p['date'])

# Extract date features
sales_p['day'] = sales_p['date'].dt.day
sales_p['month'] = sales_p['date'].dt.month
sales_p['year'] = sales_p['date'].dt.year
sales_p['DayOfWeek'] = sales_p['date'].dt.isocalendar().week

# Drop original date column
sales_p = sales_p.drop(columns=['date'])
sales_p

# Handle missing values (fill with median)
#sales_p.fillna(inplace=True)
```

Python

	Unnamed: 0	store_ID	day_of_week	nb_customers_on_day	open	promotion	state_holiday	school_holiday	sales	day	month	year
0	425390	366	4	517	1	0	0	0	4422	18	4	2013
1	291687	394	6	694	1	0	0	0	8297	11	4	2015
2	411278	807	4	970	1	1	0	0	9729	29	8	2013
3	664714	802	2	473	1	1	0	0	6513	28	5	2013
4	540835	726	4	1068	1	1	0	0	10882	10	10	2013
...	...	...	...	...	...	...	...	...	...	...	...	...
640835	359783	409	6	483	1	0	0	0	4553	26	10	2013
640836	152315	97	1	987	1	1	0	0	12307	14	4	2014
640837	117952	987	1	925	1	0	0	0	6800	7	7	2014
640838	435829	1084	4	725	1	0	0	0	5344	12	6	2014
640839	305711	695	7	0	0	0	0	0	0	3	5	2015

640840 rows × 13 columns

Spaces: 4 Cell 32 of 33

SOLUTION STEPS

TRAIN THE MODEL

```
# Train the model
model_rf = RandomForestRegressor(n_estimators=100, random_state=42)
model_rf.fit(X_train, y_train)
```

[24]

```
RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score

# Make predictions on the test set
y_pred = model_rf.predict(X_test)

# Calculate evaluation metrics
mae_rf = mean_absolute_error(y_test, y_pred)
mse_rf= mean_squared_error(y_test, y_pred)
r2_rf = r2_score(y_test, y_pred)

# Print results
print(f'Mean Absolute Error (MAE): {mae_rf:.2f}')
print(f'Mean Squared Error (MSE): {mse_rf:.2f}')
print(f'R² Score (test): {r2_rf:.2f}')

# Make predictions on the train set
y_test_pred_rf=model.predict(X_train)
r_squared_rf = r2_score(y_train, y_test_pred_rf)

print(f"R-squared score (train): {r_squared_rf}")
```

[25]

```
Mean Absolute Error (MAE): 566.73
Mean Squared Error (MSE): 814642.47
R² Score (test): 0.94
R-squared score (train): 0.8546698450809198
```

SOLUTION STEPS

7. MAKE PREDICTIONS ON NEW DATA

```
# Predict sales
real_d['Predicted_Sales'] = model_rf.predict(real_d)

print("Predictions generated successfully!")
```

```
Predictions generated successfully!
```

```
real_d.to_csv('G5.csv', index=False)
```

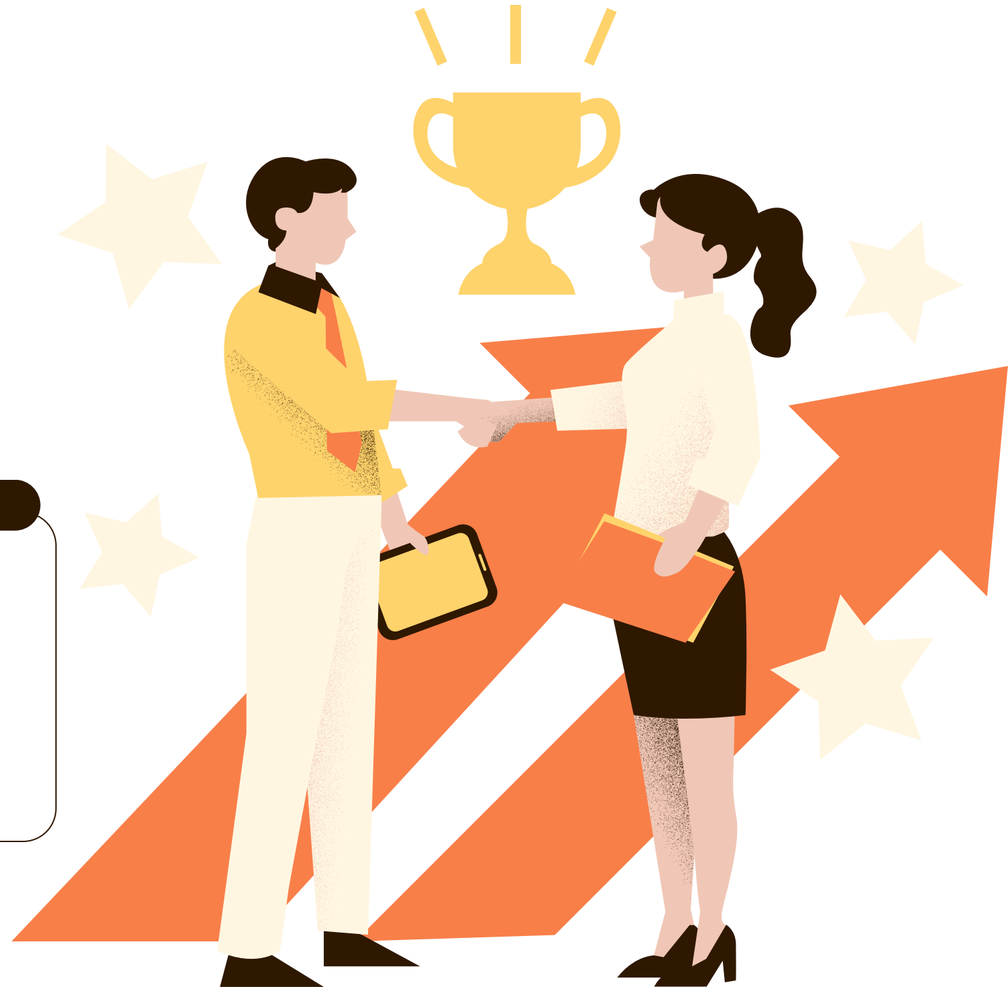
```
real_d
```

	Unnamed: 0	store_ID	day_of_week	nb_customers_on_day	open	promotion	state_holiday	school_holiday	day	month	year
0	0	415	7	0	0	0	0	0	0	0	0
1	0	27	7	0	0	0	0	0	0	0	0
2	0	404	3	657	1	1	0	0	0	0	0
3	0	683	2	862	1	0	0	0	0	0	0
4	0	920	3	591	1	1	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...
71200	0	441	7	0	0	0	0	0	0	0	0
71201	0	377	7	0	0	0	0	0	0	0	0
71202	0	15	3	648	1	0	0	0	0	0	0
71203	0	950	2	626	1	1	0	0	0	0	0
71204	0	932	4	828	1	1	0	0	0	0	0

71205 rows × 13 columns

REAL DATA

REAL_DATA



SAUDI DIGITAL ACADEMY
AI Bootcamp

THANK YOU