MGMT 59000-103 Project

February 20, 2024

1 Data Pre-processing for Sales Data

[1]:
```
import pandas as pd
import numpy as np

# Load the csv files
sales_train_df = pd.read_csv('sales_train_validation.csv')
calendar_df = pd.read_csv('calendar.csv')


# Filter for only Texas stores: TX_1, TX_2, TX_3
texas_stores = ['TX_1', 'TX_2', 'TX_3']
sales_train_df = sales_train_df[sales_train_df['store_id'].isin(texas_stores)]


# Define the day columns for the last 400 days from d_1913
start_day = 1913 - 399  # Calculate the starting day (d_1913 - 399 days)
day_columns = ['d_' + str(day) for day in range(start_day, 1914)]  # List of␣
↪day columns from d_1514 to d_1913

# Select only the necessary columns: id columns + last 400 days
id_columns = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id']
selected_columns = id_columns + day_columns
sales_train_df= sales_train_df[selected_columns]


# Remove "_validation" from the 'id' column
sales_train_df['id'] = sales_train_df['id'].str.replace('_validation', '')

# Remove specified columns from the sales_train_df
sales_train_df_modified = sales_train_df.drop(['item_id', 'dept_id', 'cat_id',␣
↪'store_id', 'state_id'], axis=1)

# Transpose the modified sales_train_df
# To preserve the 'id' column, set it as index before transposing
sales_train_df_transposed = sales_train_df_modified.set_index('id').transpose().
↪reset_index()
```

```python
# Rename columns for merging
sales_train_df_transposed.rename(columns={'index': 'd'}, inplace=True)

print(sales_train_df_transposed.head())

sales_train_df_transposed.to_csv('sales_transpose.csv', index=False)
```

```
id       d  HOBBIES_1_001_TX_1  HOBBIES_1_002_TX_1  HOBBIES_1_003_TX_1  \
0   d_1514                   0                   0                   0
1   d_1515                   0                   0                   0
2   d_1516                   3                   0                   0
3   d_1517                   0                   0                   0
4   d_1518                   0                   0                   0

id  HOBBIES_1_004_TX_1  HOBBIES_1_005_TX_1  HOBBIES_1_006_TX_1  \
0                    2                   0                   2
1                    3                   0                   0
2                    0                   0                   0
3                    0                   0                   0
4                    3                   3                   1

id  HOBBIES_1_007_TX_1  HOBBIES_1_008_TX_1  HOBBIES_1_009_TX_1  …  \
0                    0                   2                   0  …
1                    0                   0                   0  …
2                    0                   8                   0  …
3                    0                  10                   0  …
4                    1                   0                   1  …

id  FOODS_3_818_TX_3  FOODS_3_819_TX_3  FOODS_3_820_TX_3  FOODS_3_821_TX_3  \
0                  3                 1                 1                 0
1                  0                 1                 0                 0
2                  2                 1                 1                 0
3                  3                 0                 0                 0
4                  0                 1                 1                 1

id  FOODS_3_822_TX_3  FOODS_3_823_TX_3  FOODS_3_824_TX_3  FOODS_3_825_TX_3  \
0                  0                 0                 0                 0
1                  3                 1                 0                 2
2                  0                 0                 0                 0
3                  0                 0                 0                 0
4                  0                 0                 0                 0

id  FOODS_3_826_TX_3  FOODS_3_827_TX_3
0                  2                 7
1                  1                 0
2                  1                 2
```

```
3                     0                     0
4                     0                     0

[5 rows x 9148 columns]
```

2 Feature Engineering for Sales Data

[2]:
```
# Function to add moving averages and lagged sales for a given column
'''
def add_features(df, column_name):
    # Apply a 1-day lag
    lagged_series = df[column_name].shift(1)

    # Directly calculate the 7-day and 14-day rolling averages from the 1-day␣
 ↪lagged series
    # without storing the lagged series as a separate column in the dataframe
    df[column_name + '_1d_lag_7d_MA'] = lagged_series.rolling(window=7).mean()
    df[column_name + '_1d_lag_14d_MA'] = lagged_series.rolling(window=14).mean()

# Iterate over each product column to add the modified features
for column in sales_train_df_transposed.columns:
    if column != 'd':  # Assuming 'd' is the column you want to exclude
        add_features(sales_train_df_transposed, column)

'''


def add_features_efficiently(df, column_names):
    # Create a temporary DataFrame to store all new features
    new_features = pd.DataFrame(index=df.index)

    for column_name in column_names:
        if column_name != 'd':  # Assuming 'd' is the column you want to exclude
            # Apply a 1-day lag
            lagged_series = df[column_name].shift(1)


            # Calculate the 7-day and 14-day rolling averages from the 1-day␣
 ↪lagged series
            new_features[column_name + '_1d_lag_7d_MA'] = lagged_series.
 ↪rolling(window=7).mean()
            new_features[column_name + '_1d_lag_14d_MA'] = lagged_series.
 ↪rolling(window=14).mean()

    # Concatenate the new features with the original DataFrame
    return pd.concat([df, new_features], axis=1)
```

```
column_names = sales_train_df_transposed.columns
sales_train_df_transposed = add_features_efficiently(sales_train_df_transposed,␣
 ↪column_names)


# Verify the results
print(sales_train_df_transposed.head())
```

```
/tmp/ipykernel_122652/2723853685.py:31: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe =
frame.copy()`
  new_features[column_name + '_1d_lag_7d_MA'] =
lagged_series.rolling(window=7).mean()
/tmp/ipykernel_122652/2723853685.py:32: PerformanceWarning: DataFrame is highly
fragmented.  This is usually the result of calling `frame.insert` many times,
which has poor performance.  Consider joining all columns at once using
pd.concat(axis=1) instead.  To get a de-fragmented frame, use `newframe =
frame.copy()`
  new_features[column_name + '_1d_lag_14d_MA'] =
lagged_series.rolling(window=14).mean()
```

	d	HOBBIES_1_001_TX_1	HOBBIES_1_002_TX_1	HOBBIES_1_003_TX_1	\
0	d_1514	0	0	0	
1	d_1515	0	0	0	
2	d_1516	3	0	0	
3	d_1517	0	0	0	
4	d_1518	0	0	0	

	HOBBIES_1_004_TX_1	HOBBIES_1_005_TX_1	HOBBIES_1_006_TX_1	\
0	2	0	2	
1	3	0	0	
2	0	0	0	
3	0	0	0	
4	3	3	1	

	HOBBIES_1_007_TX_1	HOBBIES_1_008_TX_1	HOBBIES_1_009_TX_1	…	\
0	0	2	0	…	
1	0	0	0	…	
2	0	8	0	…	
3	0	10	0	…	
4	1	0	1	…	

	FOODS_3_823_TX_3_1d_lag_7d_MA	FOODS_3_823_TX_3_1d_lag_14d_MA	\
0	NaN	NaN	

```
1                                NaN                                NaN
2                                NaN                                NaN
3                                NaN                                NaN
4                                NaN                                NaN

   FOODS_3_824_TX_3_1d_lag_7d_MA  FOODS_3_824_TX_3_1d_lag_14d_MA  \
0                            NaN                            NaN
1                            NaN                            NaN
2                            NaN                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

   FOODS_3_825_TX_3_1d_lag_7d_MA  FOODS_3_825_TX_3_1d_lag_14d_MA  \
0                            NaN                            NaN
1                            NaN                            NaN
2                            NaN                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

   FOODS_3_826_TX_3_1d_lag_7d_MA  FOODS_3_826_TX_3_1d_lag_14d_MA  \
0                            NaN                            NaN
1                            NaN                            NaN
2                            NaN                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

   FOODS_3_827_TX_3_1d_lag_7d_MA  FOODS_3_827_TX_3_1d_lag_14d_MA
0                            NaN                            NaN
1                            NaN                            NaN
2                            NaN                            NaN
3                            NaN                            NaN
4                            NaN                            NaN

[5 rows x 27442 columns]
```

[3]:
```python
# Remove first 35 rows to filter data for past 365 days

sales_train_df_transposed = sales_train_df_transposed.iloc[35:]

# Reset the index after dropping the rows, if you want a continuous index␣
 ↪starting from 0
sales_train_df_transposed.reset_index(drop=True, inplace=True)

# Display the first few rows of the modified DataFrame to verify
print(sales_train_df_transposed.head())

# sales_train_df_transposed.to_csv('sales_t.csv', index=False)
```

```
        d  HOBBIES_1_001_TX_1  HOBBIES_1_002_TX_1  HOBBIES_1_003_TX_1  \
0  d_1549                   1                   0                   0
1  d_1550                   0                   1                   0
2  d_1551                   0                   0                   0
3  d_1552                   3                   0                   0
4  d_1553                   0                   0                   0

   HOBBIES_1_004_TX_1  HOBBIES_1_005_TX_1  HOBBIES_1_006_TX_1  \
0                   4                   2                   3
1                   0                   0                   0
2                   4                   0                   2
3                   4                   0                   0
4                   0                   2                   0

   HOBBIES_1_007_TX_1  HOBBIES_1_008_TX_1  HOBBIES_1_009_TX_1  …  \
0                   1                   6                   0  …
1                   0                   0                   1  …
2                   0                   7                   3  …
3                   0                   1                   0  …
4                   0                   0                   1  …

   FOODS_3_823_TX_3_1d_lag_7d_MA  FOODS_3_823_TX_3_1d_lag_14d_MA  \
0                       0.142857                        0.500000
1                       0.142857                        0.428571
2                       0.142857                        0.357143
3                       0.142857                        0.285714
4                       0.285714                        0.357143

   FOODS_3_824_TX_3_1d_lag_7d_MA  FOODS_3_824_TX_3_1d_lag_14d_MA  \
0                            0.0                             0.0
1                            0.0                             0.0
2                            0.0                             0.0
3                            0.0                             0.0
4                            0.0                             0.0

   FOODS_3_825_TX_3_1d_lag_7d_MA  FOODS_3_825_TX_3_1d_lag_14d_MA  \
0                       0.857143                        0.928571
1                       0.857143                        1.000000
2                       0.857143                        0.857143
3                       1.000000                        0.785714
4                       0.857143                        0.785714

   FOODS_3_826_TX_3_1d_lag_7d_MA  FOODS_3_826_TX_3_1d_lag_14d_MA  \
0                       2.142857                        1.571429
1                       2.000000                        1.571429
2                       2.000000                        1.857143
3                       1.571429                        1.857143
4                       1.428571                        1.857143
```

```
   FOODS_3_827_TX_3_1d_lag_7d_MA  FOODS_3_827_TX_3_1d_lag_14d_MA
0                       1.857143                        1.928571
1                       1.714286                        1.357143
2                       1.142857                        1.357143
3                       1.000000                        1.285714
4                       1.000000                        1.214286

[5 rows x 27442 columns]
```

3 Calendar Data Encoding

[4]:
```
import pandas as pd
from sklearn.preprocessing import LabelEncoder

# Define the transform function
def transform(calendar):
    nan_features = ['event_name_1', 'event_type_1', 'event_name_2',␣
 ↪'event_type_2']
    for feature in nan_features:
        calendar[feature].fillna('unknown', inplace=True)

    cat = ['event_name_1', 'event_type_1', 'event_name_2', 'event_type_2',␣
 ↪'snap_CA', 'snap_TX', 'snap_WI']
    for feature in cat:
        encoder = LabelEncoder()
        calendar[feature] = encoder.fit_transform(calendar[feature])

    return calendar

# Load the dataset


# Apply the transform function to the dataset
transformed_data = transform(calendar_df)

# Display the first few rows of the reordered DataFrame
print(calendar_df.head())
```

```
/apps/cent7/jupyterhub/lib/python3.9/site-packages/scipy/__init__.py:146:
UserWarning: A NumPy version >=1.16.5 and <1.23.0 is required for this version
of SciPy (detected version 1.26.4
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
         date  wm_yr_wk    weekday  wday  month  year    d  event_name_1  \
0  2011-01-29     11101   Saturday     1      1  2011  d_1            30
1  2011-01-30     11101     Sunday     2      1  2011  d_2            30
```

```
2  2011-01-31     11101     Monday     3      1  2011  d_3            30
3  2011-02-01     11101    Tuesday     4      2  2011  d_4            30
4  2011-02-02     11101  Wednesday     5      2  2011  d_5            30

   event_type_1  event_name_2  event_type_2  snap_CA  snap_TX  snap_WI
0             4             4             2        0        0        0
1             4             4             2        0        0        0
2             4             4             2        0        0        0
3             4             4             2        1        1        0
4             4             4             2        1        0        1
```

4 Merging Sales and Calendar Data

[5]:
```
# Merge the transposed sales data with the calendar data
merged_df = pd.merge(sales_train_df_transposed, calendar_df, on='d', how='left')

# Step 1: Identify the calendar columns (assuming these are all columns from␣
 ↪'date' onwards in the merged_df)
calendar_columns = ['date', 'wm_yr_wk', 'weekday', 'wday', 'month', 'year',␣
 ↪'event_name_1', 'event_type_1', 'event_name_2', 'event_type_2', 'snap_CA',␣
 ↪'snap_TX', 'snap_WI']

# Step 2: Construct the new column order
new_column_order = ['d'] + calendar_columns + [col for col in merged_df.columns␣
 ↪if col not in calendar_columns and col != 'd']

# Step 3: Reorder the columns of merged_df
merged_df = merged_df[new_column_order]

# Drop non essential columns
columns_to_remove = ['snap_CA', 'snap_WI', 'weekday', 'wday', 'month',␣
 ↪'year','date']

# Remove the specified columns
merged_df = merged_df.drop(columns=columns_to_remove)

# Display the first few rows of the modified DataFrame to verify
print(merged_df.head())
```

```
        d  wm_yr_wk  event_name_1  event_type_1  event_name_2  event_type_2  \
0  d_1549     11513            30             4             4             2
1  d_1550     11513            30             4             4             2
2  d_1551     11513            30             4             4             2
3  d_1552     11513            30             4             4             2
4  d_1553     11513            30             4             4             2

   snap_TX  HOBBIES_1_001_TX_1  HOBBIES_1_002_TX_1  HOBBIES_1_003_TX_1  …  \
```

```
0        0                    1                    0                    0  …
1        0                    0                    1                    0  …
2        0                    0                    0                    0  …
3        0                    3                    0                    0  …
4        0                    0                    0                    0  …

   FOODS_3_823_TX_3_1d_lag_7d_MA  FOODS_3_823_TX_3_1d_lag_14d_MA  \
0                       0.142857                        0.500000
1                       0.142857                        0.428571
2                       0.142857                        0.357143
3                       0.142857                        0.285714
4                       0.285714                        0.357143

   FOODS_3_824_TX_3_1d_lag_7d_MA  FOODS_3_824_TX_3_1d_lag_14d_MA  \
0                            0.0                             0.0
1                            0.0                             0.0
2                            0.0                             0.0
3                            0.0                             0.0
4                            0.0                             0.0

   FOODS_3_825_TX_3_1d_lag_7d_MA  FOODS_3_825_TX_3_1d_lag_14d_MA  \
0                       0.857143                        0.928571
1                       0.857143                        1.000000
2                       0.857143                        0.857143
3                       1.000000                        0.785714
4                       0.857143                        0.785714

   FOODS_3_826_TX_3_1d_lag_7d_MA  FOODS_3_826_TX_3_1d_lag_14d_MA  \
0                       2.142857                        1.571429
1                       2.000000                        1.571429
2                       2.000000                        1.857143
3                       1.571429                        1.857143
4                       1.428571                        1.857143

   FOODS_3_827_TX_3_1d_lag_7d_MA  FOODS_3_827_TX_3_1d_lag_14d_MA
0                       1.857143                        1.928571
1                       1.714286                        1.357143
2                       1.142857                        1.357143
3                       1.000000                        1.285714
4                       1.000000                        1.214286

[5 rows x 27448 columns]
```

[6]:
```
dimensions = sales_train_df.shape
print(dimensions)
```

```
(9147, 406)
```

5 Data Preprocessing of Sell Prices Data

[7]:
```
import pandas as pd

# Specify the path to your CSV file
csv_file_path = 'sell_prices.csv'

# Load the CSV file into a DataFrame
df = pd.read_csv(csv_file_path)

# Get the dimensions of the DataFrame
dimensions = df.shape

# Print the dimensions
print(f'Number of Rows: {dimensions[0]}')
print(f'Number of Columns: {dimensions[1]}')
```

```
Number of Rows: 6841121
Number of Columns: 4
```

[8]:
```
import pandas as pd

# Load the datasets
sell_prices_path = 'sell_prices.csv'
calendar_data_path = 'calendar.csv'

sell_prices_data = pd.read_csv(sell_prices_path)
calendar_data = pd.read_csv(calendar_data_path)

# Define the list of Texas store IDs
texas_stores = ['TX_1', 'TX_2', 'TX_3']

# Filter the sell_prices_data for only the specified Texas stores
sell_prices_data = sell_prices_data[sell_prices_data['store_id'].
 ↪isin(texas_stores)]

# Display the first few rows of the filtered DataFrame to verify
print(sell_prices_data.head())

# Step 1: Create a new "id" column in sell_prices_data by concatenating␣
 ↪"item_id" and "store_id"
sell_prices_data['id'] = 'price_'+sell_prices_data['item_id'] + '_' +␣
 ↪sell_prices_data['store_id']

# Step 2: Remove 'store_id', 'item_id', and 'wm_yr_wk' columns
sell_prices_data.drop(['store_id', 'item_id'], axis=1, inplace=True)
```

```
# Step 3: Bring the 'id' column to the front
# Reorder the columns so 'id' is first
columns = ['id'] + [col for col in sell_prices_data.columns if col != 'id']
sell_prices_data = sell_prices_data[columns]

print(sell_prices_data.head())

# Get the dimensions of the DataFrame
dimensions = sell_prices_data.shape

# Print the dimensions
print(f'Number of Rows: {dimensions[0]}')
print(f'Number of Columns: {dimensions[1]}')
```

```
        store_id        item_id  wm_yr_wk  sell_price
2708822     TX_1  HOBBIES_1_001     11325        9.58
2708823     TX_1  HOBBIES_1_001     11326        8.26
2708824     TX_1  HOBBIES_1_001     11327        8.26
2708825     TX_1  HOBBIES_1_001     11328        8.26
2708826     TX_1  HOBBIES_1_001     11329        8.26
                              id  wm_yr_wk  sell_price
2708822  price_HOBBIES_1_001_TX_1     11325        9.58
2708823  price_HOBBIES_1_001_TX_1     11326        8.26
2708824  price_HOBBIES_1_001_TX_1     11327        8.26
2708825  price_HOBBIES_1_001_TX_1     11328        8.26
2708826  price_HOBBIES_1_001_TX_1     11329        8.26
Number of Rows: 2092122
Number of Columns: 3
```

[9]:
```
# Pivot the DataFrame to make 'd' values columns, 'id' as the index, and␣
 ↪'sell_price' as the values
pivot_df = sell_prices_data.pivot(index='id', columns='wm_yr_wk',␣
 ↪values='sell_price')

# Display the first few rows of the pivoted DataFrame
print(pivot_df.head())

# Get the dimensions of the DataFrame
dimensions = pivot_df.shape

# Print the dimensions
print(f'Number of Rows: {dimensions[0]}')
print(f'Number of Columns: {dimensions[1]}')
```

```
wm_yr_wk                11101  11102  11103  11104  11105  11106  11107  \
id
price_FOODS_1_001_TX_1   2.00   2.00   2.00   2.00   2.00   2.00   2.00
```

```
price_FOODS_1_001_TX_2   2.00   2.00   2.00   2.00   2.00   2.00   2.00
price_FOODS_1_001_TX_3    NaN   2.00   2.00   2.00   2.00   2.00   2.00
price_FOODS_1_002_TX_1    NaN   7.88   7.88   7.88   7.88   7.88   7.88
price_FOODS_1_002_TX_2   7.88   7.88   7.88   7.88   7.88   7.88   7.88

wm_yr_wk                11108  11109  11110  …  11612  11613  11614  11615  \
id                                            …
price_FOODS_1_001_TX_1   2.00   2.00   2.00  …   2.24   2.24   2.24   2.24
price_FOODS_1_001_TX_2   2.00   2.00   2.00  …   2.24   2.24   2.24   2.24
price_FOODS_1_001_TX_3   2.00   2.00   2.00  …   2.24   2.24   2.24   2.24
price_FOODS_1_002_TX_1   7.88   7.88   7.88  …   9.48   9.48   9.48   9.48
price_FOODS_1_002_TX_2   7.88   7.88   7.88  …   9.48   9.48   9.48   9.48

wm_yr_wk                11616  11617  11618  11619  11620  11621
id
price_FOODS_1_001_TX_1   2.24   2.24   2.24   2.24   2.24   2.24
price_FOODS_1_001_TX_2   2.24   2.24   2.24   2.24   2.24   2.24
price_FOODS_1_001_TX_3   2.24   2.24   2.24   2.24   2.24   2.24
price_FOODS_1_002_TX_1   9.48   9.48   9.48   9.48   9.48   9.48
price_FOODS_1_002_TX_2   9.48   9.48   9.48   9.48   9.48   9.48

[5 rows x 282 columns]
Number of Rows: 9147
Number of Columns: 282
```

[10]:
```python
# Transpose the pivoted DataFrame
transposed_df = pivot_df.T

# Clear the name of the index to remove 'id' label from the top left corner
transposed_df.index.name = None

# Reset the index to turn the index into a regular column, then rename
transposed_df_reset = transposed_df.reset_index()
transposed_df_reset.rename(columns={'index': 'wm_yr_wk'}, inplace=True)


print(transposed_df_reset.head())


# Get the dimensions of the DataFrame
dimensions = transposed_df_reset.shape

# Print the dimensions
print(f'Number of Rows: {dimensions[0]}')
print(f'Number of Columns: {dimensions[1]}')
```

```
id  wm_yr_wk  price_FOODS_1_001_TX_1  price_FOODS_1_001_TX_2  \
0      11101                     2.0                     2.0
```

```
1      11102                    2.0                    2.0
2      11103                    2.0                    2.0
3      11104                    2.0                    2.0
4      11105                    2.0                    2.0

id  price_FOODS_1_001_TX_3  price_FOODS_1_002_TX_1  price_FOODS_1_002_TX_2  \
0                      NaN                     NaN                    7.88
1                      2.0                    7.88                    7.88
2                      2.0                    7.88                    7.88
3                      2.0                    7.88                    7.88
4                      2.0                    7.88                    7.88

id  price_FOODS_1_002_TX_3  price_FOODS_1_003_TX_1  price_FOODS_1_003_TX_2  \
0                    7.88                    2.88                    2.88
1                    7.88                    2.88                    2.88
2                    7.88                    2.88                    2.88
3                    7.88                    2.88                    2.88
4                    7.88                    2.88                    2.88

id  price_FOODS_1_003_TX_3  …  price_HOUSEHOLD_2_513_TX_3  \
0                    2.88  …                         NaN
1                    2.88  …                         NaN
2                    2.88  …                         NaN
3                    2.88  …                         NaN
4                    2.88  …                         NaN

id  price_HOUSEHOLD_2_514_TX_1  price_HOUSEHOLD_2_514_TX_2  \
0                          NaN                       18.97
1                        18.97                       18.97
2                        18.97                       18.97
3                        18.97                       18.97
4                        18.97                       18.97

id  price_HOUSEHOLD_2_514_TX_3  price_HOUSEHOLD_2_515_TX_1  \
0                        18.97                         NaN
1                        18.97                         NaN
2                        18.97                         NaN
3                        18.97                         NaN
4                        18.97                         NaN

id  price_HOUSEHOLD_2_515_TX_2  price_HOUSEHOLD_2_515_TX_3  \
0                          NaN                         NaN
1                          NaN                         NaN
2                          NaN                         NaN
3                          NaN                         NaN
4                          NaN                         NaN

id  price_HOUSEHOLD_2_516_TX_1  price_HOUSEHOLD_2_516_TX_2  \
```

```
0                           NaN                           5.94
1                           NaN                           5.94
2                          5.94                           5.94
3                          5.94                           5.94
4                          5.94                           5.94

id  price_HOUSEHOLD_2_516_TX_3
0                          5.94
1                          5.94
2                          5.94
3                          5.94
4                          5.94

[5 rows x 9148 columns]
Number of Rows: 282
Number of Columns: 9148
```

6 Merging Sell price data and Sales Data

[11]:
```python
import pandas as pd

# Assuming merged_df and transposed_df_reset are already defined and ready

# Step 1: Identify the position of 'snap_TX' in merged_data to know where to␣
 ↪insert new columns
snap_WI_position = merged_df.columns.get_loc("snap_TX") + 1

# Step 2: Merge the DataFrames on 'wm_yr_wk'
# Note: This is a simplification. You might need a left, right, or outer join␣
 ↪depending on your data context.
merged_final = pd.merge(merged_df, transposed_df_reset, on="wm_yr_wk",␣
 ↪how="left")

'''
# Step 3: Reorder columns to ensure new columns are after 'snap_WI', if␣
 ↪necessary
# This step is somewhat complex because it requires dynamically adjusting the␣
 ↪column order based on merge results

# Get a list of all column names
columns = list(merged_final.columns)

# Identify the columns that came from transposed_df_reset (assuming they're not␣
 ↪in merged_data originally)
new_columns = [col for col in transposed_df_reset.columns if col not in␣
 ↪merged_df.columns and col != 'wm_yr_wk']
```

```
# Reorganize columns: keep everything up to 'snap_WI' in place, insert␣
 ↪new_columns, then append the rest
organized_columns = columns[:snap_WI_position] + new_columns +␣
 ↪columns[snap_WI_position:len(columns)-len(new_columns)]

# Reassign column order
merged_final = merged_final[organized_columns]

'''

# Verify the merge and column order
print(merged_final.head())

merged_final.to_csv('merged_final.csv', index=False)
```

```
        d  wm_yr_wk  event_name_1  event_type_1  event_name_2  event_type_2  \
0  d_1549     11513            30             4             4             2
1  d_1550     11513            30             4             4             2
2  d_1551     11513            30             4             4             2
3  d_1552     11513            30             4             4             2
4  d_1553     11513            30             4             4             2

   snap_TX  HOBBIES_1_001_TX_1  HOBBIES_1_002_TX_1  HOBBIES_1_003_TX_1  …  \
0        0                   1                   0                   0  …
1        0                   0                   1                   0  …
2        0                   0                   0                   0  …
3        0                   3                   0                   0  …
4        0                   0                   0                   0  …

   price_HOUSEHOLD_2_513_TX_3  price_HOUSEHOLD_2_514_TX_1  \
0                        2.78                       19.54
1                        2.78                       19.54
2                        2.78                       19.54
3                        2.78                       19.54
4                        2.78                       19.54

   price_HOUSEHOLD_2_514_TX_2  price_HOUSEHOLD_2_514_TX_3  \
0                       19.54                       19.54
1                       19.54                       19.54
2                       19.54                       19.54
3                       19.54                       19.54
4                       19.54                       19.54

   price_HOUSEHOLD_2_515_TX_1  price_HOUSEHOLD_2_515_TX_2  \
0                        1.97                        1.97
1                        1.97                        1.97
```

```
2                         1.97                         1.97
3                         1.97                         1.97
4                         1.97                         1.97

   price_HOUSEHOLD_2_515_TX_3  price_HOUSEHOLD_2_516_TX_1  \
0                        1.97                        5.94
1                        1.97                        5.94
2                        1.97                        5.94
3                        1.97                        5.94
4                        1.97                        5.94

   price_HOUSEHOLD_2_516_TX_2  price_HOUSEHOLD_2_516_TX_3
0                        5.94                        5.94
1                        5.94                        5.94
2                        5.94                        5.94
3                        5.94                        5.94
4                        5.94                        5.94

[5 rows x 36595 columns]
```

7 Downcasting of Data

[12]:
```
import numpy as np
```

[13]:
```
def downcasting(df):
    float_cols = [c for c in df if df[c].dtype == "float64"]
    int_cols = [c for c in df if df[c].dtype in ["int64", "int32"]]
    df[float_cols] = df[float_cols].astype(np.float32)
    df[int_cols] = df[int_cols].astype(np.int16)
    return df

merged_final=downcasting(merged_final)


# Verify the merge and column order
print(merged_final.head())

# Get the dimensions of the DataFrame
dimensions = merged_final.shape

# Print the dimensions
print(f'Number of Rows: {dimensions[0]}')
print(f'Number of Columns: {dimensions[1]}')
```

```
        d  wm_yr_wk  event_name_1  event_type_1  event_name_2  event_type_2  \
0  d_1549     11513            30             4             4             2
1  d_1550     11513            30             4             4             2
```

```
2  d_1551     11513            30             4             4             2
3  d_1552     11513            30             4             4             2
4  d_1553     11513            30             4             4             2

   snap_TX  HOBBIES_1_001_TX_1  HOBBIES_1_002_TX_1  HOBBIES_1_003_TX_1  …  \
0        0                   1                   0                   0  …
1        0                   0                   1                   0  …
2        0                   0                   0                   0  …
3        0                   3                   0                   0  …
4        0                   0                   0                   0  …

   price_HOUSEHOLD_2_513_TX_3  price_HOUSEHOLD_2_514_TX_1  \
0                        2.78                   19.540001
1                        2.78                   19.540001
2                        2.78                   19.540001
3                        2.78                   19.540001
4                        2.78                   19.540001

   price_HOUSEHOLD_2_514_TX_2  price_HOUSEHOLD_2_514_TX_3  \
0                   19.540001                   19.540001
1                   19.540001                   19.540001
2                   19.540001                   19.540001
3                   19.540001                   19.540001
4                   19.540001                   19.540001

   price_HOUSEHOLD_2_515_TX_1  price_HOUSEHOLD_2_515_TX_2  \
0                        1.97                        1.97
1                        1.97                        1.97
2                        1.97                        1.97
3                        1.97                        1.97
4                        1.97                        1.97

   price_HOUSEHOLD_2_515_TX_3  price_HOUSEHOLD_2_516_TX_1  \
0                        1.97                        5.94
1                        1.97                        5.94
2                        1.97                        5.94
3                        1.97                        5.94
4                        1.97                        5.94

   price_HOUSEHOLD_2_516_TX_2  price_HOUSEHOLD_2_516_TX_3
0                        5.94                        5.94
1                        5.94                        5.94
2                        5.94                        5.94
3                        5.94                        5.94
4                        5.94                        5.94

[5 rows x 36595 columns]
Number of Rows: 365
```

```
Number of Columns: 36595
```

8 Saving data as a csv

[14]:
```
merged_final = merged_final.drop(columns=['wm_yr_wk'])
merged_final.to_csv('merged_data_final.csv', index=False)
```

[ ]:
```

```

Part 2 Modeling

February 20, 2024

[1]:
```
pip install numpy pandas scikit-learn tensorflow
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: numpy in /home/sborah/.local/lib/python3.9/site-
packages (1.26.4)
Requirement already satisfied: pandas in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (1.3.4)
Requirement already satisfied: scikit-learn in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (0.24.2)
Requirement already satisfied: tensorflow in
/home/sborah/.local/lib/python3.9/site-packages (2.15.0.post1)
Requirement already satisfied: python-dateutil>=2.7.3 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from pandas) (2021.3)
Requirement already satisfied: joblib>=0.11 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from scikit-learn) (1.1.0)
Requirement already satisfied: scipy>=0.19.1 in
/home/sborah/.local/lib/python3.9/site-packages (from scikit-learn) (1.12.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from scikit-learn) (2.2.0)
Requirement already satisfied: tensorflow-estimator<2.16,>=2.15.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: google-pasta>=0.1.1 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: libclang>=13.0.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (16.0.6)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (0.36.0)
Requirement already satisfied: astunparse>=1.6.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (1.6.3)
Requirement already satisfied: opt-einsum>=2.3.2 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied:
protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3
in /home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (4.25.2)
Requirement already satisfied: setuptools in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from tensorflow) (58.0.4)
```

```
Requirement already satisfied: h5py>=2.9.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from tensorflow) (3.3.0)
Requirement already satisfied: gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (0.5.4)
Requirement already satisfied: packaging in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from tensorflow) (21.0)
Requirement already satisfied: termcolor>=1.1.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (2.4.0)
Requirement already satisfied: typing-extensions>=3.6.6 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (4.9.0)
Requirement already satisfied: tensorboard<2.16,>=2.15 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (2.15.2)
Requirement already satisfied: keras<2.16,>=2.15.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (2.15.0)
Requirement already satisfied: absl-py>=1.0.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (2.1.0)
Requirement already satisfied: ml-dtypes~=0.2.0 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (0.2.0)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (1.60.1)
Requirement already satisfied: flatbuffers>=23.5.26 in
/home/sborah/.local/lib/python3.9/site-packages (from tensorflow) (23.5.26)
Requirement already satisfied: six>=1.12.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from tensorflow) (1.16.0)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from tensorflow) (1.12.1)
Requirement already satisfied: wheel<1.0,>=0.23.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
astunparse>=1.6.0->tensorflow) (0.37.0)
Requirement already satisfied: werkzeug>=1.0.1 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.0.2)
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in
/home/sborah/.local/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (0.7.2)
Requirement already satisfied: requests<3,>=2.21.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.26.0)
Requirement already satisfied: google-auth-oauthlib<2,>=0.5 in
/home/sborah/.local/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (1.2.0)
Requirement already satisfied: markdown>=2.6.8 in
/home/sborah/.local/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (3.5.2)
Requirement already satisfied: google-auth<3,>=1.6.3 in
/home/sborah/.local/lib/python3.9/site-packages (from
tensorboard<2.16,>=2.15->tensorflow) (2.28.0)
Requirement already satisfied: rsa<5,>=3.1.4 in
```

```
/home/sborah/.local/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (4.9)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in
/home/sborah/.local/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (5.3.2)
Requirement already satisfied: pyasn1-modules>=0.2.1 in
/home/sborah/.local/lib/python3.9/site-packages (from google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.3.0)
Requirement already satisfied: requests-oauthlib>=0.7.0 in
/home/sborah/.local/lib/python3.9/site-packages (from google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (1.3.1)
Requirement already satisfied: importlib-metadata>=4.4 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
markdown>=2.6.8->tensorboard<2.16,>=2.15->tensorflow) (4.8.1)
Requirement already satisfied: zipp>=0.5 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from importlib-
metadata>=4.4->markdown>=2.6.8->tensorboard<2.16,>=2.15->tensorflow) (3.6.0)
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in
/home/sborah/.local/lib/python3.9/site-packages (from
pyasn1-modules>=0.2.1->google-
auth<3,>=1.6.3->tensorboard<2.16,>=2.15->tensorflow) (0.5.1)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (1.26.7)
Requirement already satisfied: idna<4,>=2.5 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (3.2)
Requirement already satisfied: charset-normalizer~=2.0.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2.0.4)
Requirement already satisfied: certifi>=2017.4.17 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from
requests<3,>=2.21.0->tensorboard<2.16,>=2.15->tensorflow) (2021.10.8)
Requirement already satisfied: oauthlib>=3.0.0 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from requests-
oauthlib>=0.7.0->google-auth-
oauthlib<2,>=0.5->tensorboard<2.16,>=2.15->tensorflow) (3.1.1)
Requirement already satisfied: pyparsing>=2.0.2 in
/apps/cent7/jupyterhub/lib/python3.9/site-packages (from packaging->tensorflow)
(3.0.4)
Note: you may need to restart the kernel to use updated packages.
```

1 Tensorflow memory growth

[2]:
```
import tensorflow as tf

# Enable TensorFlow's memory growth
gpus = tf.config.experimental.list_physical_devices('GPU')
if gpus:
    try:
        for gpu in gpus:
            tf.config.experimental.set_memory_growth(gpu, True)
    except RuntimeError as e:
        # Memory growth must be set at program startup
        print(e)
```

```
2024-02-20 01:08:58.013330: E
external/local_xla/xla/stream_executor/cuda/cuda_dnn.cc:9261] Unable to register
cuDNN factory: Attempting to register factory for plugin cuDNN when one has
already been registered
2024-02-20 01:08:58.013374: E
external/local_xla/xla/stream_executor/cuda/cuda_fft.cc:607] Unable to register
cuFFT factory: Attempting to register factory for plugin cuFFT when one has
already been registered
2024-02-20 01:08:58.037188: E
external/local_xla/xla/stream_executor/cuda/cuda_blas.cc:1515] Unable to
register cuBLAS factory: Attempting to register factory for plugin cuBLAS when
one has already been registered
2024-02-20 01:08:58.085537: I tensorflow/core/platform/cpu_feature_guard.cc:182]
This TensorFlow binary is optimized to use available CPU instructions in
performance-critical operations.
To enable the following instructions: AVX2 AVX512F FMA, in other operations,
rebuild TensorFlow with the appropriate compiler flags.
2024-02-20 01:09:00.141022: W
tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Could not
find TensorRT
```

[3]:
```
import numpy as np
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from tensorflow.keras.callbacks import EarlyStopping
```

2 Data preprocessing and Normalisation

[4]:
```
# Optionally enable mixed precision
tf.keras.mixed_precision.set_global_policy('mixed_float16')

# The rest of your code for loading data, preprocessing, model definition, and␣
 ↪training goes here


# Load the data
df = pd.read_csv('merged_data_final.csv')

# Assuming the first column is the day and the rest are features
features = df.iloc[:, 1:].values

# Assuming 'features' is your DataFrame
features_df = pd.DataFrame(features)

# Identify and drop constant columns
constant_columns = features_df.columns[features_df.nunique() <= 1]
features_df.drop(constant_columns, axis=1, inplace=True)

features_df_new = features_df.dropna(axis=1)

# If you want to drop columns that are entirely NaN, you can use the 'how'␣
 ↪parameter
# features_df_cleaned = features_df.dropna(axis=1, how='all')

# Show the shape of the original and cleaned DataFrames as a quick check
print("Original shape:", features_df.shape)
print("Cleaned shape:", features_df_new.shape)

# Scale the features
scaler = MinMaxScaler(feature_range=(0, 1))
features_scaled = scaler.fit_transform(features_df_new)
```

```
INFO:tensorflow:Mixed precision compatibility check (mixed_float16): OK
Your GPUs will likely run quickly with dtype policy mixed_float16 as they all
have compute capability of at least 7.0
Original shape: (365, 29943)
Cleaned shape: (365, 29937)
```

3 LSTM Model

[14]:
```
#model 2

import numpy as np
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Reshape, TimeDistributed


def create_dataset(data, look_back=28, forecast_horizon=28):
    X, Y = [], []
    for i in range(len(data) - look_back - forecast_horizon + 1):
        a = data[i:(i + look_back)]
        X.append(a)
        Y.append(data[(i + look_back):(i + look_back + forecast_horizon)])
    return np.array(X), np.array(Y)


look_back = 45
forecast_horizon = 28
X, Y = create_dataset(features_scaled, look_back, forecast_horizon)

# Split the data into training and testing sets
split = int(len(X) * 0.8)
X_train, X_test = X[:split], X[split:]
Y_train, Y_test = Y[:split], Y[split:]

# No need to reshape X_train and X_test as they are already in the correct shape
# However, you might want to ensure Y_train and Y_test are correctly shaped for␣
 ↪your model, especially if using LSTM layers for output


# Adjusting the model architecture
number_of_features = 29937
forecast_horizon = 28
look_back = 45

model = Sequential([
    LSTM(50, input_shape=(look_back, X_train.shape[2]), return_sequences=True),
    LSTM(50),
    # Use Reshape or TimeDistributed layer to adjust for the output shape (28␣
 ↪days, 29937 features each day)
    # Adding a Dense layer with the number of outputs you need for each time␣
 ↪step, wrapped in a TimeDistributed layer
    # to apply it across each of the 28 time steps
    Dense(forecast_horizon * number_of_features, activation='linear'),
    # Reshape the output to the desired format: [samples, time steps, features]
```

```
    Reshape((forecast_horizon, number_of_features))
])


learning_rate = 0.001

optimizer=tf.keras.optimizers.Adam(learning_rate=learning_rate)


model.compile(optimizer=optimizer, loss='mean_squared_error')

# Note: With such a large output dimension, training this model might require␣
 ↪significant computational resources.
# Ensure that your hardware is capable of handling this complexity.

# Update the model's weights to float16 for mixed precision training
model.summary()  # Check if the model uses mixed precision

# Early stopping to prevent overfitting
early_stopping = EarlyStopping(monitor='val_loss', patience=10, mode='min')

# Reduce batch size to decrease memory consumption
batch_size = 32  # Reduced batch size

# Train the model
model.fit(X_train, Y_train, epochs=100, batch_size=batch_size,␣
 ↪validation_split=0.1, callbacks=[early_stopping], verbose=2)
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 lstm_6 (LSTM)               (None, 45, 50)            5997600

 lstm_7 (LSTM)               (None, 50)                20200

 dense_3 (Dense)             (None, 838236)            42750036

 reshape_3 (Reshape)         (None, 28, 29937)         0

=================================================================
Total params: 48767836 (186.03 MB)
Trainable params: 48767836 (186.03 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/100
7/7 - 5s - loss: 0.1540 - val_loss: 0.1536 - 5s/epoch - 703ms/step
```

```
Epoch 2/100
7/7 - 1s - loss: 0.1351 - val_loss: 0.1234 - 695ms/epoch - 99ms/step
Epoch 3/100
7/7 - 1s - loss: 0.0991 - val_loss: 0.0852 - 686ms/epoch - 98ms/step
Epoch 4/100
7/7 - 1s - loss: 0.0631 - val_loss: 0.0603 - 726ms/epoch - 104ms/step
Epoch 5/100
7/7 - 1s - loss: 0.0451 - val_loss: 0.0559 - 696ms/epoch - 99ms/step
Epoch 6/100
7/7 - 1s - loss: 0.0427 - val_loss: 0.0579 - 716ms/epoch - 102ms/step
Epoch 7/100
7/7 - 1s - loss: 0.0426 - val_loss: 0.0579 - 734ms/epoch - 105ms/step
Epoch 8/100
7/7 - 1s - loss: 0.0414 - val_loss: 0.0570 - 766ms/epoch - 109ms/step
Epoch 9/100
7/7 - 1s - loss: 0.0409 - val_loss: 0.0556 - 682ms/epoch - 97ms/step
Epoch 10/100
7/7 - 1s - loss: 0.0407 - val_loss: 0.0555 - 664ms/epoch - 95ms/step
Epoch 11/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0555 - 673ms/epoch - 96ms/step
Epoch 12/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0556 - 697ms/epoch - 100ms/step
Epoch 13/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0553 - 735ms/epoch - 105ms/step
Epoch 14/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0549 - 680ms/epoch - 97ms/step
Epoch 15/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0551 - 734ms/epoch - 105ms/step
Epoch 16/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0555 - 745ms/epoch - 106ms/step
Epoch 17/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0560 - 676ms/epoch - 97ms/step
Epoch 18/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0555 - 711ms/epoch - 102ms/step
Epoch 19/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0552 - 764ms/epoch - 109ms/step
Epoch 20/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0550 - 774ms/epoch - 111ms/step
Epoch 21/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0542 - 826ms/epoch - 118ms/step
Epoch 22/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0552 - 738ms/epoch - 105ms/step
Epoch 23/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0557 - 695ms/epoch - 99ms/step
Epoch 24/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0558 - 706ms/epoch - 101ms/step
Epoch 25/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0552 - 693ms/epoch - 99ms/step
```

```
Epoch 26/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0552 - 758ms/epoch - 108ms/step
Epoch 27/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0551 - 679ms/epoch - 97ms/step
Epoch 28/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0554 - 750ms/epoch - 107ms/step
Epoch 29/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0556 - 697ms/epoch - 100ms/step
Epoch 30/100
7/7 - 1s - loss: 0.0406 - val_loss: 0.0554 - 682ms/epoch - 97ms/step
Epoch 31/100
7/7 - 1s - loss: 0.0405 - val_loss: 0.0553 - 649ms/epoch - 93ms/step
```

[14]:
```
<keras.src.callbacks.History at 0x2ac6a88073d0>
```

[15]:
```
shape_X_test = X_test.shape
print(f"Shape of X_test: {shape_X_test}")

# Find the shape of Y_test
shape_Y_test = Y_test.shape
print(f"Shape of Y_test: {shape_Y_test}")
```

```
Shape of X_test: (59, 45, 29937)
Shape of Y_test: (59, 28, 29937)
```

4 Model Evaluation

[16]:
```
# Model 2

from sklearn.metrics import mean_squared_error
from math import sqrt

# Predicting the features for the test set
predicted_features = model.predict(X_test)

# Assuming `predicted_features` and `Y_test` are now correctly inverse␣
 ↪transformed and have shapes [samples, 28, 29937]

# Flatten the 3D arrays to 2D arrays
predicted_features_flat = predicted_features.reshape(-1, predicted_features.
 ↪shape[1]*predicted_features.shape[2])
Y_test_flat = Y_test.reshape(-1, Y_test.shape[1]*Y_test.shape[2])

# Now both arrays are 2D: [samples, 28*29937]

# Calculating RMSE
rmse = sqrt(mean_squared_error(Y_test_flat, predicted_features_flat))
```

```
print(f"Test RMSE: {rmse}")

# Optionally, calculate additional metrics like MAE in a similar flattened␣
 ↪manner
```

```
2/2 [==============================] - 1s 57ms/step
Test RMSE: 0.27395926736619974
```

[22]:
```
from sklearn.metrics import mean_squared_error
from math import sqrt
import numpy as np

# Function to inverse transform 3D data
def inverse_transform_3d(scaler, data):
    # Reshape data from 3D to 2D to apply inverse transformation
    data_reshaped = data.reshape(-1, data.shape[2])
    data_inverse = scaler.inverse_transform(data_reshaped)
    # Reshape back to 3D
    data_inverse_3d = data_inverse.reshape(data.shape)
    return data_inverse_3d

# Apply inverse transformation to predictions and actual values
predicted_features_inv = inverse_transform_3d(scaler, predicted_features)
Y_test_inv = inverse_transform_3d(scaler, Y_test)

# Flatten the 3D arrays to 2D for overall RMSE calculation
predicted_features_flat = predicted_features_inv.reshape(-1,␣
 ↪predicted_features_inv.shape[1]*predicted_features_inv.shape[2])
Y_test_flat = Y_test_inv.reshape(-1, Y_test_inv.shape[1]*Y_test_inv.shape[2])

# Calculating RMSE
rmse = sqrt(mean_squared_error(Y_test_flat, predicted_features_flat))
print(f"Test RMSE: {rmse}")
```

```
Test RMSE: 1.5886257376946042
```