

# Walmart Sales Forecasting Using Multivariate Time Series Analysis Techniques

การคาดการณ์ยอดขายของวอลมาร์ตโดยใช้เทคนิคการวิเคราะห์อนุกรมเวลาหลายตัวแปร



พชร นิยมลรัตน์  
รหัสนักศึกษา 65076043



## ความเป็นมาและความสำคัญ



ร้านค้าปลีกยังคงมีบทบาทสำคัญในโลกอีคอมเมิร์ซ การขายปลีกเป็นกระบวนการขายสินค้าอุปโภคบริโภคหรือบริการให้กับลูกค้าเพื่อแสวงหาผลกำไร ทำให้สนใจที่จะทำนายยอดขายของร้านสะดวกซื้อเพื่อที่จะบรรลุเป้าหมายที่กำหนดไว้อาจส่งผลเชิงบวกต่อราคาหุ้นและการรับรู้ของนักลงทุน โดยมีฤดูกาลหรือวันหยุดในแต่ละสัปดาห์ที่มียอดขายสูงหรือต่ำกว่าค่าเฉลี่ย หากบริษัทไม่เตรียมตัวกับฤดูกาลเหล่านี้อาจสูญเสียเงินจำนวนมาก บริษัทจึงมีกลยุทธ์ในการจัดกิจกรรมลดราคา โดยมีการส่งเสริมสินค้าหลายรายการตลอดทั้งปี ซึ่งการลดราคามักจะเกิดขึ้นช่วงวันหยุดสำคัญ 4 ครั้งต่อปี ได้แก่ SuperBowl(ซูเปอร์โบวล์), Labour Day(วันแรงงาน), Thanksgiving(วันขอบคุณพระเจ้า) และ Christmas(คริสต์มาส)

## ความเป็นมาและความสำคัญ



รวมถึงปัจจัยทางเศรษฐกิจอาจส่งผลกระทบต่อการทำงานและความสามารถในการทำกำไรของบริษัท ดังนั้นจึงมีความจำเป็นต้องวิเคราะห์ภาวะทางเศรษฐกิจเพื่อนำไปสู่การคาดการณ์ทิศทางการขยายตัวของเศรษฐกิจโดยมีปัจจัยดังนี้ อัตราการว่างงาน, ดัชนีราคาผู้บริโภค, ผลิตภัณฑ์มวลรวมภายในประเทศ, อัตราการออมส่วนบุคคล และ อัตราดอกเบี้ย ซึ่งจะสะท้อนให้เห็นถึงสภาพเศรษฐกิจและตลาดแรงงานในช่วงระยะเวลานั้น ดังนั้นเราจึงอยากทราบว่าปัจจัยอะไรบ้างที่ส่งผลต่อยอดขาย



## วัตถุประสงค์ของการศึกษา

1. เพื่อวิเคราะห์การคาดการณ์ยอดขายของร้านค้าในอนาคต
2. เพื่อวิเคราะห์ปัจจัยที่ส่งผลต่อการคาดการณ์ยอดขายของร้านค้าในหนึ่งสัปดาห์
3. เพื่อวิเคราะห์ยอดขายที่เพิ่มขึ้นในช่วงสัปดาห์ที่มีวันหยุด
4. เพื่อวิเคราะห์ปัจจัยทางเศรษฐกิจที่ส่งผลต่อการคาดการณ์ยอดขาย





## ขั้นตอนในการดำเนินงาน

### 1.Data Exploration & Data Visualization

- Walmart Sales

### 2.Data Preparation & Preprocessing

- Data cleaning
- Feature Selection
- Train Test split 80:20

### 3.Model

- Vector autoregression(VAR)
- LSTM
- Phophet
- ARIMA
- Exponentialsmoothering(ETS)

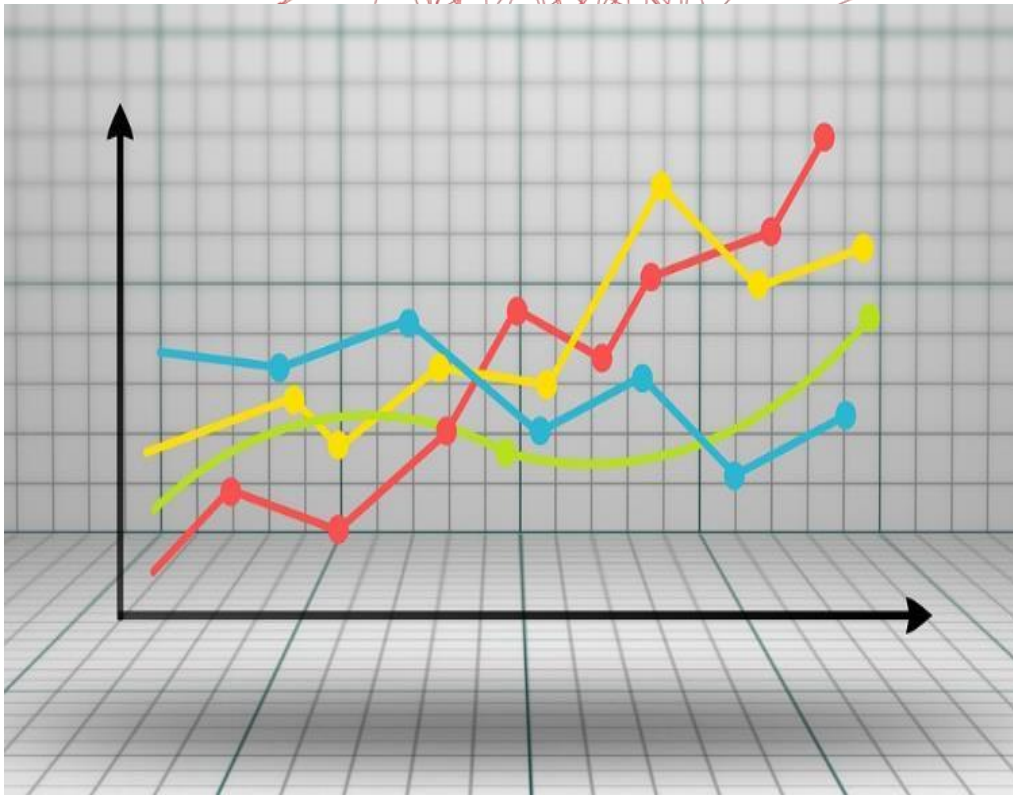
### 4. Use the obtained features to test with other data sets

- Global super store

### 5.Web Application

- Python using Flask API Framework

## Multivariate Time Series (MTS)



เป็นอนุกรมเวลาแบบหลายตัวแปร ซึ่งตัวแปรแต่ละตัวขึ้นอยู่กับตัวแปรอื่นๆ ไม่เพียงแต่ขึ้นอยู่กับค่าในอดีต บางครั้งการทำนายช่วยในการปรับข้อมูลให้เหมาะสมกับโมเดลโดยพิจารณาความเปลี่ยนแปลงตามเวลาในตัวแปรหลายตัว ซึ่งช่วยในการเข้าใจและรับรู้ถึงข้อมูลที่ซับซ้อน ซึ่งใช้สำหรับการคาดการณ์ค่าในอนาคต

# Multivariate Time Series (MTS)

Variable y1	Variable y2	Variable y1	Variable y2
$y1_{t-n}$	$y2_{t-n}$	$y1_{t-n}$	$y2_{t-n}$
...	...	...	...
$y1_{t-2}$	$y2_{t-2}$	$y1_{t-2}$	$y2_{t-2}$
$y1_{t-1}$	$y2_{t-1}$	$y1_{t-1}$	$y2_{t-1}$
$y1_t$	$y2_t$	$y1_t$	$y2_t$

$$\begin{bmatrix} y1(t) \\ y2(t) \\ y3(t) \end{bmatrix} = \begin{bmatrix} b_{10} \\ b_{20} \\ b_{30} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \begin{bmatrix} y1(t-1) \\ y2(t-1) \\ y3(t-1) \end{bmatrix} + \begin{bmatrix} \epsilon_1(t) \\ \epsilon_2(t) \\ \epsilon_3(t) \end{bmatrix}$$

- $a_1$  and  $a_2$  are the constant terms,
- $w_{11}, w_{12}, w_{21},$  and  $w_{22}$  are the coefficients,
- $e_1$  and  $e_2$  are the error terms

VAR

Vector Autoregressive (VAR) เป็นอัลกอริทึมที่ตัวแปรแต่ละตัวจะเป็นฟังก์ชันเชิงเส้นของค่าในอดีตของตัวเองและค่าในอดีตของตัวแปรอื่นๆ ทั้งหมด ใช้วิเคราะห์ความสัมพันธ์ของอนุกรมเวลาและผลกระทบเชิงเคลื่อนที่ของการรบกวนแบบสุ่มต่อตัวแปร



# Multivariate Time Series (MTS)

## LSTM

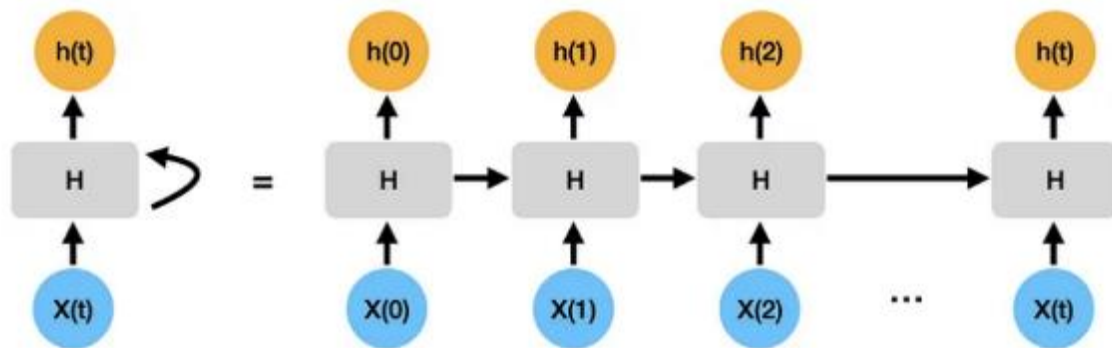


Figure 5: Unrolled RNN with  $X(t)$  as input at time  $t$ ,  $H$  as the hidden layer, and  $h(t)$  as the hidden layer output at time  $t$  [Image by Author]

Long Short-Term Memory (LSTM) ใช้ทำนายชุดข้อมูล  
ที่ประกอบด้วยหลายตัวแปรที่มีความสัมพันธ์กันใน  
ช่วงเวลาที่ต่างกันพร้อมกัน LSTM สามารถจดจำ  
ความสัมพันธ์และลำดับของข้อมูลในช่วงเวลาที่  
ยาวนานได้ดี ทำให้เหมาะสำหรับการจัดการกับข้อมูล  
และการเปลี่ยนแปลงเชิงเวลาที่มีความซับซ้อน





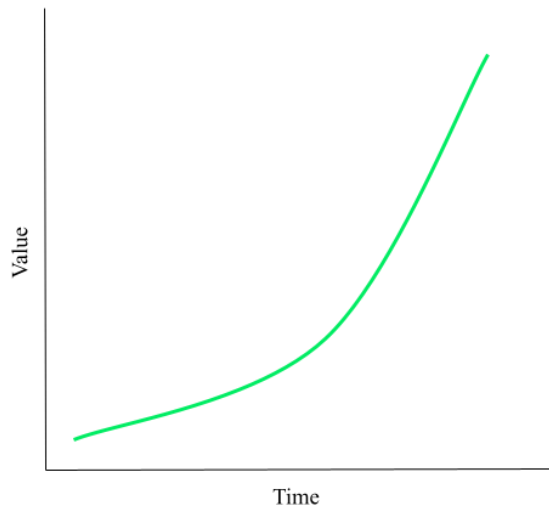
# Multivariate Time Series

## Prophet

เป็นโมเดลพยากรณ์อนุกรมเวลาที่มีส่วนประกอบสำหรับแนวโน้ม ฤดูกาล วันหยุด และการถดถอย ซึ่งสามารถจัดการกับแนวโน้มที่ไม่ใช่เชิงเส้นที่มีหลายช่วงเวลาและข้อมูลที่ขาดหายไปได้

# Univariate Time Series

## Univariate Time Series



เป็นอนุกรมเวลาแบบตัวแปรเดียวซึ่งแต่ละค่าข้อมูลจะถูกบันทึกในช่วงเวลาที่แตกต่างกันหรือในช่วงเวลาเดียวกันโดยไม่มีตัวแปรอื่นที่มีผลต่อตัวแปรหลัก



# Univariate Time Series

## ARIMA

ARIMA Model คือ Auto Regressive Integrated Moving Average พยายามกำจัด “Noise” ออกเพื่อที่จะลด Error ให้ได้มากที่สุดโดยแบบจำลอง ARIMA(p,d,q) มีส่วนประกอบ 3 ส่วนดังนี้

- auto regressive(AR): ทำนายค่าในอนาคตโดยใช้ค่าในอดีตของตัวแปรต้นทางอนุกรมเวลาเรียกว่า(p)
- Integrated(I): รวมข้อมูลเพื่อให้ชุดข้อมูลมีลักษณะ stationary หรือคงที่เรียกว่า(d)
- Moving Average(MA): ค่าเฉลี่ยของความแปรปรวนในอดีตเพื่อทำนายค่าในอนาคตเรียกว่า (q)





# Univariate Time Series

## Exponential Smoothing

ETS เป็นโมเดลทำนายแนวโน้มข้อมูลเวลาเพื่อปรับแต่งและทำนายข้อมูลในอนาคต ทำงานในลักษณะ state space models ซึ่งใช้เก็บข้อมูลทั้งหมดในรูปแบบ state vectors ประกอบด้วย error, trend, และ seasonal component. การทำนายข้อมูลจะเกิดขึ้นจากการอัปเดต state vectors



## การประเมินค่าความคลาดเคลื่อนจากการทำนาย

- 1.ค่าความคลาดเคลื่อนสัมบูรณ์เฉลี่ย(Mean Absolute Error : MAE)
- 2.ค่าความคลาดเคลื่อนกำลังสองเฉลี่ย(Mean Squared Error : MSE)
- 3.ค่ารากที่สองของความคลาดเคลื่อนกำลังสองเฉลี่ย(Root Mean Squared Error : RMSE)
4. ค่าเฉลี่ยของร้อยละของความคลาดเคลื่อนในการทำนาย  
(Mean Absolute Percentage Error :MAPE)

## ตารางข้อมูลในแต่คอลัมน์ที่ใช้วิเคราะห์

ชื่อคอลัมน์	ความหมาย	ชนิดข้อมูล
store	หมายเลขร้านค้า	Integer
Dept	แผนกของร้านค้า	Integer
Date	การระบุสัปดาห์(ทุกวันศุกร์ของสัปดาห์)	Categorical
Weekly_Sales	ยอดขายรายสัปดาห์	Float
IsHoliday_x	สัปดาห์นั้นเป็นวันหยุดพิเศษ	Boolean
Temperature	อุณหภูมิเฉลี่ยในภูมิภาค(°F)	Float
Fuel_Price	ค่าเชื้อเพลิงในภูมิภาค	Float
MarkDown1	การลดราคาส่งเสริมการขาย	Float
MarkDown2	การลดราคาส่งเสริมการขาย	Float
MarkDown3	การลดราคาส่งเสริมการขาย	Float
MarkDown4	การลดราคาส่งเสริมการขาย	Float
MarkDown5	การลดราคาส่งเสริมการขาย	Float
CPI	ดัชนีราคาผู้บริโภค	Float
Unemployment	อัตราการว่างงาน	Float
Type	ประเภทร้านค้า	Categorical
Size	ขนาดร้านค้า	Integer



# ตารางข้อมูลในแต่ละคอลัมน์ที่ใช้วิเคราะห์

## ข้อมูลปัจจัยภาวะทางเศรษฐกิจ

ชื่อคอลัมน์	ความหมาย	ชนิดข้อมูล
Close	ราคาหุ้นปิดของวัน	Float
GDP	ผลิตภัณฑ์มวลรวมภายในประเทศ	Float
PSAVERT	อัตราการออมส่วนบุคคล	Float
DFF	อัตราดอกเบี้ยนโยบายของสหรัฐ	Float

ข้อมูลมาจากเว็บไซต์: <https://www.kaggle.com/datasets/amandam1/walmart-stock-20122016/data>

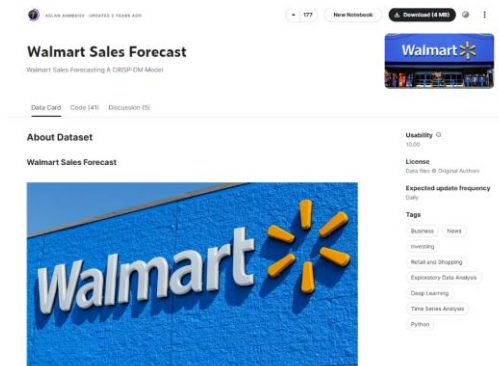
ข้อมูลมาจากเว็บไซต์: <https://www.kaggle.com/datasets/mikoajfish99/us-recession-and-financial-indicators?select=Personal+Saving+Rate.csv>

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Select Data

```
# merging 3 different sets
df = df_train.merge(df_features, on=['Store', 'Date'], how='inner').merge(df_store, on=['Store'], how='inner')
df.head(5)
```

	Store	Dept	Date	Weekly_Sales	IsHoliday	Temperature	Fuel_Price	MarkDown1	MarkDown2	MarkDown3	MarkDown4	MarkDown5	CPI	Unemployment	Type	Size
0	1	1	2010-02-05	24924.50	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	A	151315
1	1	2	2010-02-05	50605.27	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	A	151315
2	1	3	2010-02-05	13740.12	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	A	151315
3	1	4	2010-02-05	39954.04	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	A	151315
4	1	5	2010-02-05	32229.38	False	42.31	2.572	NaN	NaN	NaN	NaN	NaN	211.096358	8.106	A	151315



มีชุดข้อมูล 4 ชุดสำหรับสร้างแบบจำลองการคาดการณ์ยอดขาย โดยจะนำไฟล์ 3 ชุดข้อมูลมารวมกัน ได้แก่ Stores , Train , Features มีข้อมูลทั้งหมด 421570 แถว , 16 คอลัมน์

ข้อมูลมาจากเว็บไซต์ <https://www.kaggle.com/datasets/aslanahmedov/walmart-sales-forecast>

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Clean Data

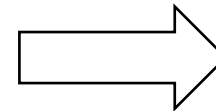
```
store_dept_table = pd.pivot_table(df, index='Store', columns='Dept',  
                                  values='Weekly_Sales', aggfunc=np.mean)  
display(store_dept_table)
```

Dept	1	2	3	4	5	6	7	8	9	10
Store										
1	22513.322937	46102.090420	13150.478042	36964.154476	24257.941119	4801.780140	24566.487413	35718.257622	28062.052238	31033.386364
2	30777.980769	65912.922517	17476.563357	45607.666573	30555.315315	6808.382517	40477.837063	58707.369441	34375.864476	38845.854476
3	7328.621049	16841.775664	5509.300769	8434.186503	11695.366573	2012.411818	10044.341608	8310.254196	9062.007692	10871.944126
4	36979.940070	93639.315385	19012.491678	56603.400140	45668.406783	8241.777692	50728.151399	62949.723776	34437.170979	37269.667413
5	9774.553077	12317.953287	4101.085175	9860.806783	6699.202238	1191.057622	6124.484336	13735.709441	7919.805944	9783.395385
6	23867.553776	50269.437273	16806.638811	34187.366503	34465.307622	7225.566643	34526.870420	47577.719790	48271.060140	47436.477902
7	9542.801259	22603.690769	8633.536923	14950.518601	13860.350490	6329.928811	10925.757063	13970.619371	29722.736084	21136.560280
8	14789.827343	35729.821748	10683.305105	21089.309301	19838.849231	3395.425455	20268.743776	26438.524336	11792.661678	20666.433776
9	11846.558252	24969.477413	7497.356783	17165.947762	19282.746014	2806.416364	13826.694336	21424.470699	13196.569720	12810.480350
10	39925.138951	109795.291469	32086.181469	48579.826364	58373.460280	10556.550769	58964.715664	86739.846643	64436.722517	48108.063497

```
df = df.loc[df['Weekly_Sales'] > 0]
```

```
df.shape # new data shape
```

```
(420212, 16)
```



```
df.isna().sum()
```

Store	0
Dept	0
Date	0
Weekly_Sales	0
IsHoliday	0
Temperature	0
Fuel_Price	0
MarkDown1	270031
MarkDown2	309308
MarkDown3	283561
MarkDown4	285694
MarkDown5	269283
CPI	0
Unemployment	0
Type	0
Size	0
Super_Bowl	0
Labor_Day	0
Thanksgiving	0
Christmas	0
dtype:	int64

```
df = df.fillna(0) # filling null's with 0
```

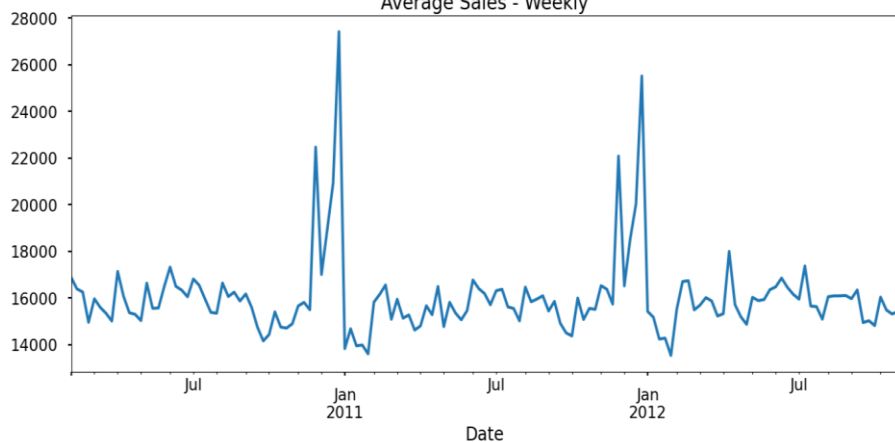
เหลือข้อมูลทั้งหมด 420212 แถว



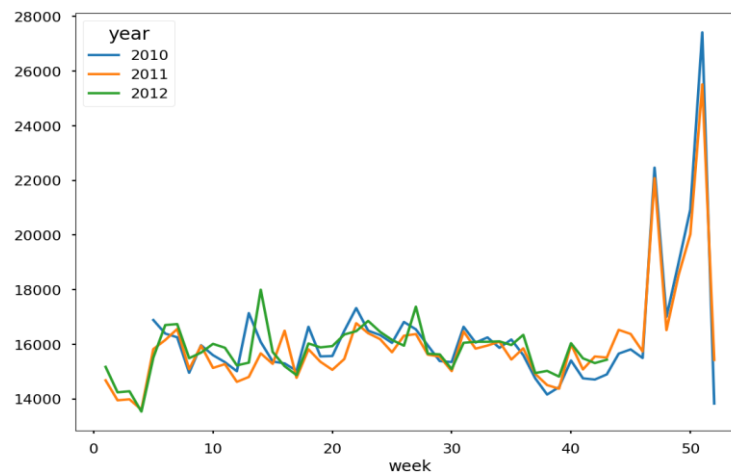
# ขั้นตอนในการวิเคราะห์ข้อมูล

## Explore Data

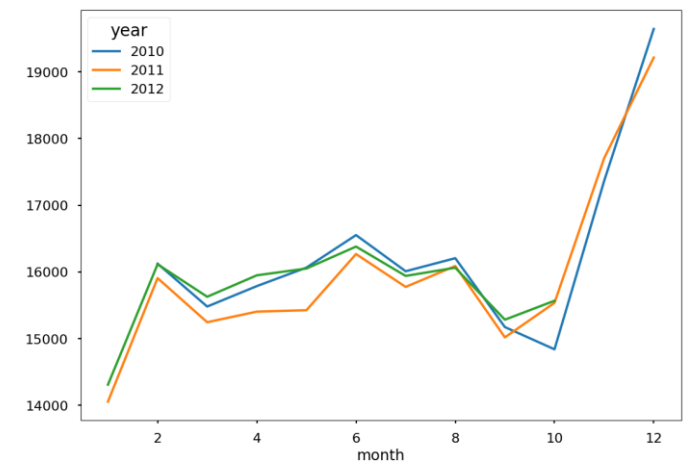
Average Sales - Weekly



กราฟค่าเฉลี่ยยอดขายรายสัปดาห์



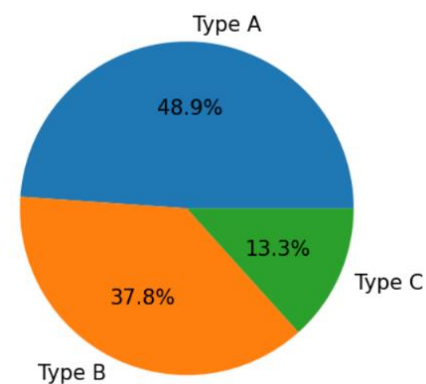
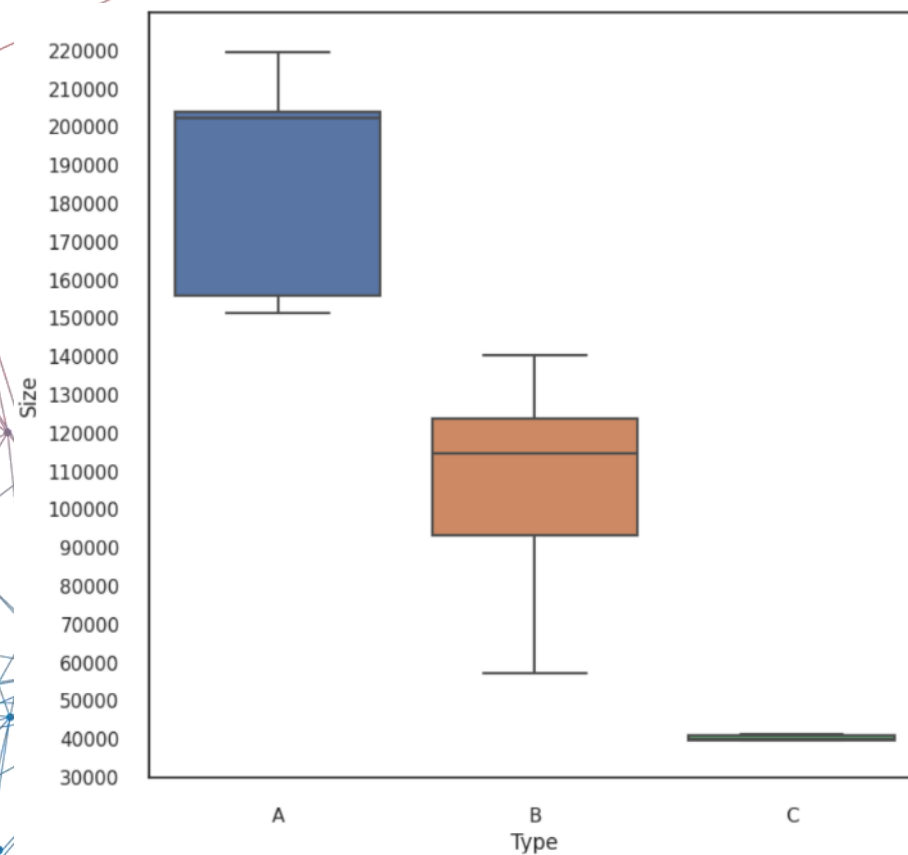
กราฟยอดขายรายสัปดาห์ในแต่ละปี



กราฟยอดขายรายเดือนในแต่ละปี

# ขั้นตอนในการวิเคราะห์ข้อมูล

Explore Data

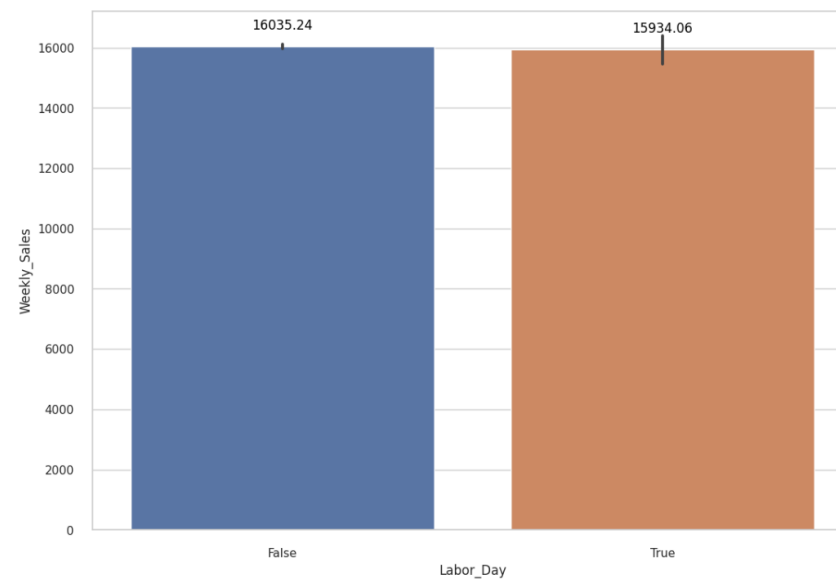
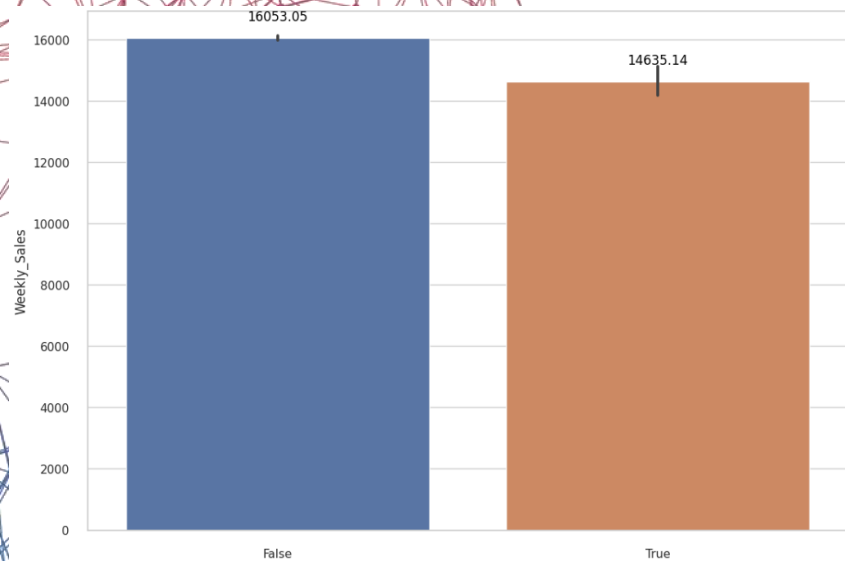


กราฟขนาดของประเภทร้านค้า

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Explore Data

กราฟแสดงยอดขายเฉลี่ยโดยเทียบวันหยุดกับวันปกติ

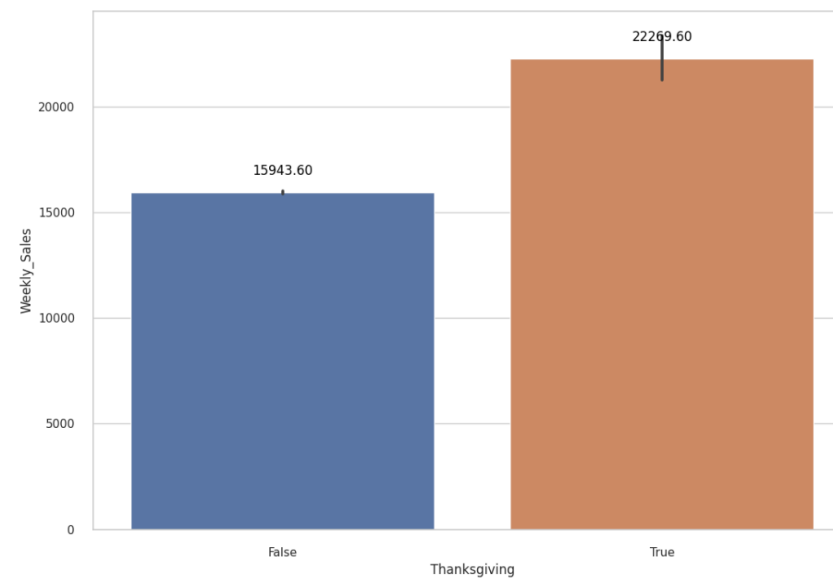




# ขั้นตอนในการวิเคราะห์ข้อมูล

## Explore Data

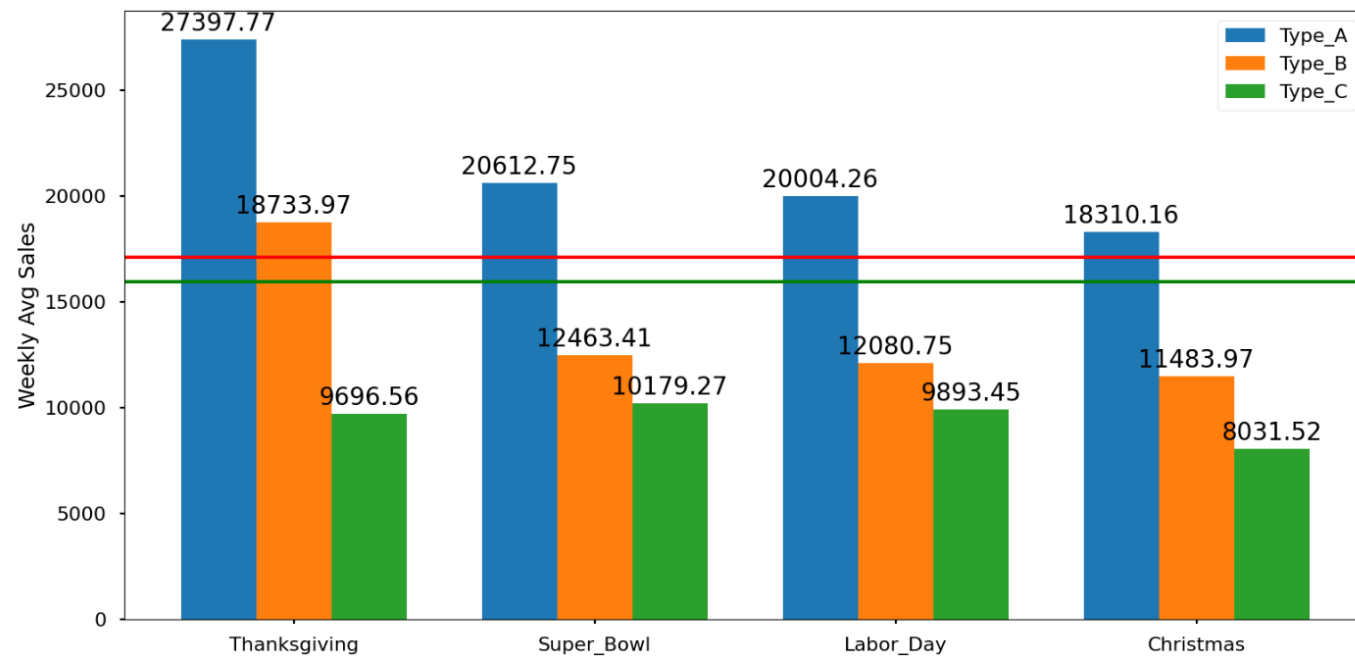
กราฟแสดงยอดขายเฉลี่ยโดยเทียบวันหยุดกับวันปกติ



# ขั้นตอนในการวิเคราะห์ข้อมูล

## Explore Data

กราฟแสดงยอดขายเฉลี่ยโดยเทียบระหว่างขนาดร้านค้าและวันหยุด



- holidays avg
- not-holiday avg

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Data preparation

### 1.Import file Economic dataframe

- Stock price
- GDP
- Personal saving rate
- DFF

### 2.แปลงคอลัมน์ 'Date' ในข้อมูล DFF ให้อยู่ในรูปแบบ YYYY-MM-DD

### 3.แปลงคอลัมน์วันที่ให้เป็นคำว่า 'Date' ในทุก dataframe

### 4.Filter ข้อมูลให้อยู่ในช่วง วันที่ 2010-02-01 ถึง 2012-12-31

### 5. set 'Date' เป็น index เพื่อ resample ข้อมูลให้อยู่ในรูปแบบรายสัปดาห์ โดยใช้ค่าเฉลี่ย

- Data frame Walmarts (Weekly\_Sales , CPI ,Unemployment)
- Stock price
- GDP
- Personal saving rate
- DFF

### 6. Fill ค่า NaN ใน dataframe GDP ให้แต่ละสามเดือนจะมีค่าเท่ากัน เนื่องจากเป็นข้อมูลแบบไตรมาส และ Fill ค่า NaN ใน dataframe Personal saving rate ให้แต่ละเดือนจะมีค่าเท่ากัน

### 7. Merge ข้อมูลทุก Data frame เข้าด้วยกัน



# ขั้นตอนในการวิเคราะห์ข้อมูล

## Data preparation

ตารางข้อมูล Walmart

	Weekly_Sales	Fuel_Price	CPI	Unemployment	Close	GDP	PSAVERT	DFF
Date								
2010-02-07	16836.121997	2.717869	167.398405	8.576731	53.532001	14764.611	5.6	0.134286
2010-02-14	16352.056032	2.696102	167.384138	8.567309	53.080001	14764.611	5.6	0.122857
2010-02-21	16216.658979	2.673666	167.338966	8.576351	53.645001	14764.611	5.6	0.125714
2010-02-28	14899.549688	2.685642	167.691019	8.561375	53.918000	14764.611	5.6	0.122857
2010-03-07	15921.015727	2.731816	167.727351	8.572689	53.850000	14764.611	5.6	0.157143

ข้อมูลอยู่ในช่วงวันที่ 2010-02-07 ถึง 2012-10-28 มีจำนวน 143 แถว , 8 คอลัมน์

# ขั้นตอนในการวิเคราะห์ข้อมูล

VAR

## Flow Chart Step to run VAR

Create Train Test splite



Granger Causality Test



Optimal lag length



Train Model



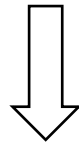
Prediction

# ขั้นตอนในการวิเคราะห์ข้อมูล

VAR

## Granger Causality test

Fuel\_Price\_y is caused by Weekly\_Sales\_x  
Close\_y is caused by Weekly\_Sales\_x  
CPI\_y is caused by Fuel\_Price\_x  
Fuel\_Price\_y is caused by CPI\_x  
Close\_y is caused by CPI\_x  
GDP\_y is caused by CPI\_x  
Fuel\_Price\_y is caused by Close\_x  
CPI\_y is caused by Close\_x  
Fuel\_Price\_y is caused by GDP\_x  
Close\_y is caused by GDP\_x



## Performing VAR directly

VAR Order Selection (\* highlights the minimums)

	AIC	BIC	FPE	HQIC
0	20.94	21.18	1.244e+09	21.04
1	12.28	13.13*	2.159e+05	12.63*
2	12.09*	13.54	1.780e+05*	12.68



# ขั้นตอนในการวิเคราะห์ข้อมูล

## VAR

## Train Model & Forecasting

```
var_model_fit = var_model.fit(maxlags=2, ic='aic')
var_model_fit.summary()
```

### Summary of Regression Results

```
Model: VAR
Method: OLS
Date: Wed, 10, Apr, 2024
Time: 19:03:50

No. of Equations: 5.00000 BIC: 13.5435
Nobs: 112.000 HQIC: 12.7502
Log likelihood: -1423.28 FPE: 201135.
AIC: 12.2086 Det(Omega_mle): 125908.
```

### Results for equation Weekly\_Sales

	coefficient	std. error	t-stat	prob
const	10189.626282	30682.285790	0.332	0.740
L1.Weekly_Sales	0.248020	0.101129	2.453	0.014
L1.Fuel_Price	-3792.828077	4526.812611	-0.838	0.402
L1.CPI	-78.232217	1318.320949	-0.059	0.953
L1.Close	-252.591290	220.688087	-1.145	0.252
L1.GDP	-0.290889	4.146387	-0.070	0.944
L2.Weekly_Sales	0.112404	0.100953	1.113	0.266
L2.Fuel_Price	3532.539783	4614.151522	0.766	0.444
L2.CPI	10.559884	1347.300242	0.008	0.994
L2.Close	348.354528	216.808739	1.607	0.108
L2.GDP	0.761622	4.076607	0.187	0.852

```
#Forecasting
var_pred = var_model_fit.forecast(y=pred_input, steps=len(df_all_test))
df_var_pred = pd.DataFrame(var_pred, index = df_all_test.index, columns = df_all_train.columns + '_1d')
df_var_pred.head()
```

	Weekly_Sales_1d	Fuel_Price_1d	CPI_1d	Close_1d	GDP_1d
Date					
2012-04-15	16943.923433	3.984502	174.996079	60.548617	16218.049686
2012-04-22	16943.711425	4.006383	175.074114	60.299166	16229.625214
2012-04-29	16812.730525	4.030360	175.161313	60.115120	16240.525805
2012-05-06	16740.463996	4.051026	175.252688	59.968702	16251.916554
2012-05-13	16678.700417	4.070252	175.348028	59.860222	16263.689357

# ขั้นตอนในการวิเคราะห์ข้อมูล

LSTM

## Flow Chart Step to run LSTM

Create Train Test splite



Scale Data



Create Model LSTM



Train Model



Prediction

# ขั้นตอนในการวิเคราะห์ข้อมูล

## LSTM

```
# เลือก features และ target variable
features = ['Fuel_Price', 'CPI', 'Close', 'GDP']
target = 'Weekly_Sales'
# ข้อมูลที่ใช้ในการเทรนและทดสอบ
X = df_all[features].values
y = df_all[target].values
# ปรับสเกลข้อมูล
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)
y_scaled = scaler.fit_transform(y.reshape(-1, 1))
# แบ่งข้อมูลเป็นชุดเทรนและชุดทดสอบ
train_size = int(len(X_scaled) * 0.8)
X_train, X_test = X_scaled[:train_size], X_scaled[train_size:]
y_train, y_test = y_scaled[:train_size], y_scaled[train_size:]
# ฟังก์ชันสำหรับสร้างชุดข้อมูลต่อเนื่อง (time series dataset)
def create_dataset(X, y, time_steps):
    Xs, ys = [], []
    for i in range(len(X) - time_steps):
        Xs.append(X[i:(i + time_steps), :])
        ys.append(y[i + time_steps])
    return np.array(Xs), np.array(ys)
# กำหนดจำนวนขั้นตอนเวลา (time steps)
time_steps = 1
# สร้างชุดข้อมูลต่อเนื่อง
X_train, y_train = create_dataset(X_train, y_train, time_steps)
X_test, y_test = create_dataset(X_test, y_test, time_steps)
# สร้างโมเดล LSTM
model = Sequential([
    LSTM(units=100, return_sequences=True, input_shape=(X_train.shape[1], X_train.shape[2])),
    LSTM(units=100, return_sequences=False),
    Dense(units=1)
])
# คอมไพล์โมเดล
model.compile(optimizer='adam', loss='mean_squared_error')
# เทรนโมเดล
model.fit(X_train, y_train, validation_data=(X_test, y_test), epochs=100, batch_size=1, verbose=1)
# ทำนาย
trainPredict = model.predict(X_train)
testPredict = model.predict(X_test)
# แปลงข้อมูลที่สเกลแล้วกลับเป็นค่าเดิม
trainPredict = scaler.inverse_transform(trainPredict)
trainY = scaler.inverse_transform(y_train)
testPredict = scaler.inverse_transform(testPredict)
testY = scaler.inverse_transform(y_test)
```



# ขั้นตอนในการวิเคราะห์ข้อมูล

Prophet

## Flow Chart Step to run Prophet

เปลี่ยนคอลัมน์ Date เป็น ds  
และตัวแปร Target เป็น y



Create Train Test splite



Train Model



Prediction

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Prophet

	ds	y	Fuel_Price	CPI	Close	GDP
0	2010-02-07	16836.121997	2.717869	167.398405	53.532001	14764.611
1	2010-02-14	16352.056032	2.696102	167.384138	53.080001	14764.611
2	2010-02-21	16216.658979	2.673666	167.338966	53.645001	14764.611
3	2010-02-28	14899.549688	2.685642	167.691019	53.918000	14764.611
4	2010-03-07	15921.015727	2.731816	167.727351	53.850000	14764.611

```
# Train the model
model_pro = Prophet()
model_pro.add_regressor('Fuel_Price')
model_pro.add_regressor('CPI')
model_pro.add_regressor('Close')
model_pro.add_regressor('GDP')

# Fit the model with train set
model_pro.fit(train_all)

# Predict on valid set
y_pred_pro = model_pro.predict(x_valid_all)
```

# ขั้นตอนในการวิเคราะห์ข้อมูล

ARIMA

## Flow Chart Step to run ARIMA

Create Train Test split



Stationary Test



Adfuller Test



Create Model ARIMA



Train Model



Prediction

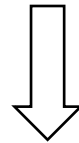


# ขั้นตอนในการวิเคราะห์ข้อมูล

ARIMA

Stationary test

	p-value	Is the series Stationary
Weekly_Sales	0.0	True



Augmented Dickey-Fuller (ADF)

```
result_ad = adfuller(df_all['Weekly_Sales'])  
  
# Extract and print the results  
print('ADF Statistic:', result_ad[0])  
print('p-value:', result_ad[1])  
print('Critical Values:', result_ad[4])
```

```
ADF Statistic: -5.930802744748703  
p-value: 2.3832272706103574e-07  
Critical Values: {'1%': -3.47864788917503, '5%': -2.882721765644168, '10%': -2.578065326612056}
```



# ขั้นตอนในการวิเคราะห์ข้อมูล

## ARIMA

```
model_auto_arima = auto_arima(train_data['Weekly_Sales'], trace=True, start_p=0, start_q=0, start_P=0, start_Q=0,
                               max_p=20, max_q=20, max_P=20, max_Q=20, seasonal=True, maxiter=200,
                               information_criterion='aic', stepwise=False, suppress_warnings=True, D=1, max_D=10,
                               error_action='ignore', approximation=False)
model_auto_arima.fit(train_data['Weekly_Sales'])
```

```
ARIMA(0,0,0)(0,0,0)[1] intercept : AIC=2056.737, Time=0.07 sec
ARIMA(0,0,1)(0,0,0)[1] intercept : AIC=2049.861, Time=0.04 sec
ARIMA(0,0,2)(0,0,0)[1] intercept : AIC=2047.325, Time=0.07 sec
ARIMA(0,0,3)(0,0,0)[1] intercept : AIC=2047.343, Time=0.11 sec
ARIMA(0,0,4)(0,0,0)[1] intercept : AIC=2028.358, Time=0.12 sec
ARIMA(0,0,5)(0,0,0)[1] intercept : AIC=2027.763, Time=0.25 sec
ARIMA(1,0,0)(0,0,0)[1] intercept : AIC=2046.098, Time=0.04 sec
ARIMA(1,0,1)(0,0,0)[1] intercept : AIC=2047.442, Time=0.10 sec
ARIMA(1,0,2)(0,0,0)[1] intercept : AIC=2049.320, Time=0.28 sec
ARIMA(1,0,3)(0,0,0)[1] intercept : AIC=2051.352, Time=0.22 sec
ARIMA(1,0,4)(0,0,0)[1] intercept : AIC=2029.658, Time=0.19 sec
ARIMA(2,0,0)(0,0,0)[1] intercept : AIC=2046.487, Time=0.07 sec
ARIMA(2,0,1)(0,0,0)[1] intercept : AIC=2048.477, Time=0.12 sec
ARIMA(2,0,2)(0,0,0)[1] intercept : AIC=2048.064, Time=0.38 sec
ARIMA(2,0,3)(0,0,0)[1] intercept : AIC=2042.946, Time=0.24 sec
ARIMA(3,0,0)(0,0,0)[1] intercept : AIC=2048.348, Time=0.08 sec
ARIMA(3,0,1)(0,0,0)[1] intercept : AIC=2050.356, Time=0.12 sec
ARIMA(3,0,2)(0,0,0)[1] intercept : AIC=2048.349, Time=0.51 sec
ARIMA(4,0,0)(0,0,0)[1] intercept : AIC=2047.818, Time=0.09 sec
ARIMA(4,0,1)(0,0,0)[1] intercept : AIC=2045.409, Time=0.41 sec
ARIMA(5,0,0)(0,0,0)[1] intercept : AIC=2031.295, Time=0.22 sec
```

Best model: ARIMA(0,0,5)(0,0,0)[1] intercept  
Total fit time: 3.772 seconds

ARIMA  
ARIMA(0,0,5)(0,0,0)[1] intercept

# ขั้นตอนในการวิเคราะห์ข้อมูล

ExponentialSmoothing

Flow Chart Step to run ETS

Create Train Test splite



Create Model ETS



Train Model



Prediction

# ขั้นตอนในการวิเคราะห์ข้อมูล

## ExponentialSmoothing

```
model_Expo = ExponentialSmoothing(train_data['Weekly_Sales'], seasonal_periods=12, seasonal='additive',  
                                   trend='additive', damped=True).fit()  
y_pred_ex = model_Expo.forecast(len(test_data['Weekly_Sales']))
```



# ขั้นตอนในการวิเคราะห์ข้อมูล

## Feature Selection

```
from sklearn.feature_selection import SelectKBest, f_regression

# เลือกคุณลักษณะที่ดีที่สุดโดยใช้ SelectKBest
def select_best_features(X_train, y_train, k=4):
    selector = SelectKBest(score_func=f_regression, k=k)
    X_selected = selector.fit_transform(X_train, y_train)
    selected_features_indices = selector.get_support(indices=True)
    return X_selected, selected_features_indices

# แบ่งข้อมูลเป็นชุด train และ test
train_size = int(len(df_all) * 0.8)
train_data, test_data = df_all[:train_size], df_all[train_size:]

# แบ่งข้อมูล train test โดยมี 1 คอลัมน์เป็น target
X_train = train_data.drop(columns=['Weekly_Sales'])
y_train = pd.DataFrame(train_data['Weekly_Sales'])
X_test = test_data.drop(columns=['Weekly_Sales'])
y_test = pd.DataFrame(test_data['Weekly_Sales'])

# เลือกคุณลักษณะที่ดีที่สุดจากข้อมูล train
X_train_selected, selected_features_indices = select_best_features(X_train, y_train)

# แสดงคุณลักษณะที่ถูกเลือก
selected_features = X_train.columns[selected_features_indices]
print("Selected Features:", selected_features)

Selected Features: Index(['Fuel_Price', 'CPI', 'Close', 'GDP'], dtype='object')
```



# ขั้นตอนในการวิเคราะห์ข้อมูล

## Result Feature Selection

### Walmart

7 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	131.28	102226.37	319.72	15.26
LSTM	118.56	132638.68	364.19	3.54
Prophet	1464.85	2709466.28	1646.04	9.36

6 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP' , 'PSAVERT' , 'DFF'

6 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	140.92	112664.77	335.65	16.74
LSTM	151.53	204634.03	452.36	3.73
Prophet	1295.57	246637.68	1570.55	8.23

5 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP' , 'PSAVERT'

5 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	154.43	125092.98	353.68	6.97
LSTM	98.83	78373.08	279.95	2.30
Prophet	685.20	744161.73	862.64	4.36

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Result Feature Selection

### Walmart

4 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP'

4 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	200.61	211485.40	459.87	6.41
LSTM	120.37	93917.82	306.46	2.20
Prophet	677.57	651797.89	807.34	4.30

3 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close'

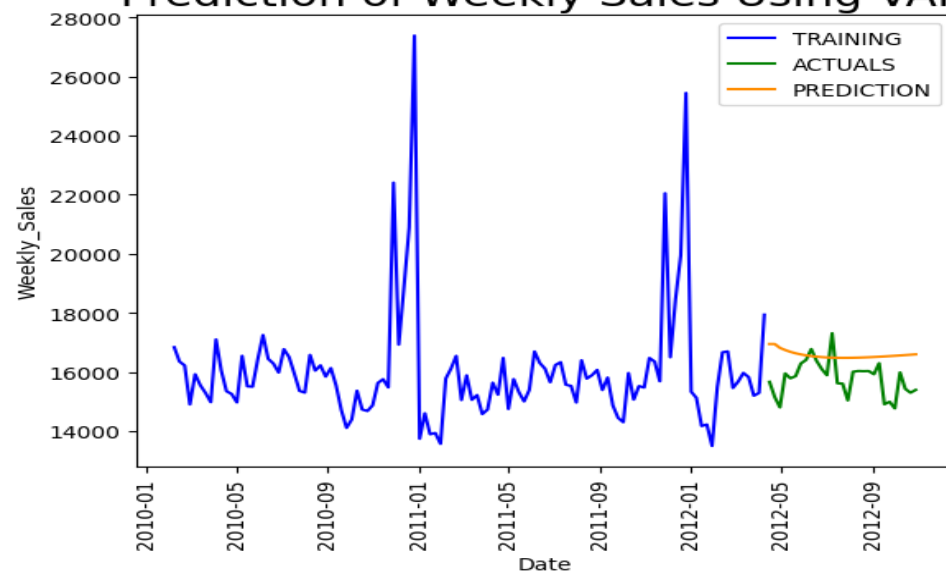
3 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	216.78	253127.06	503.11	7.71
LSTM	135.53	109938.78	331.57	2.45
Prophet	714.22	725462.98	851.74	4.53

	MAE	MSE	RMSE	MAPE
ARIMA	556.02	519113.03	720.49	3.57
ETS	1538.91	383569.5	1958.4	9.94

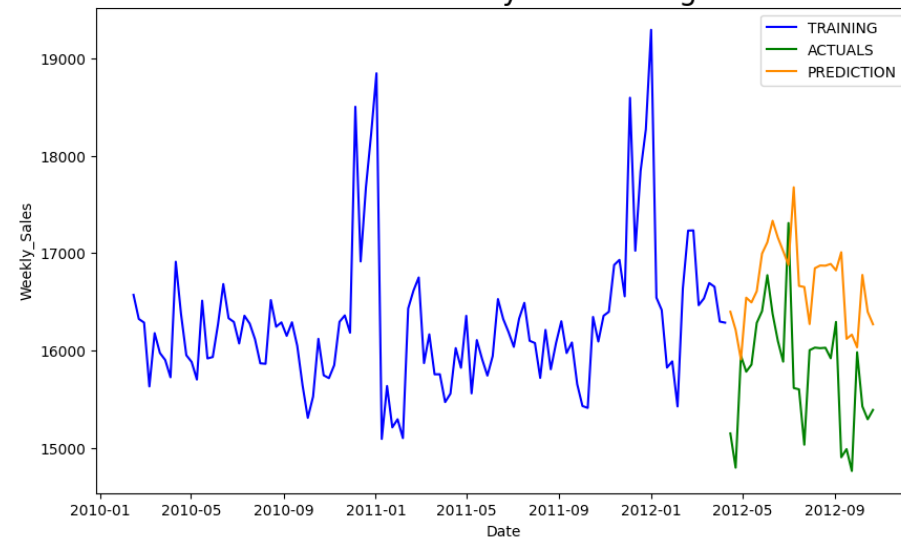
## ผลการทดลองของโมเดล

กราฟแสดงแนวโน้มยอดขายระหว่างค่าที่แท้จริงกับค่าทำนาย

Prediction of Weekly Sales Using VAR



Prediction of Weekly Sales Using LSTM



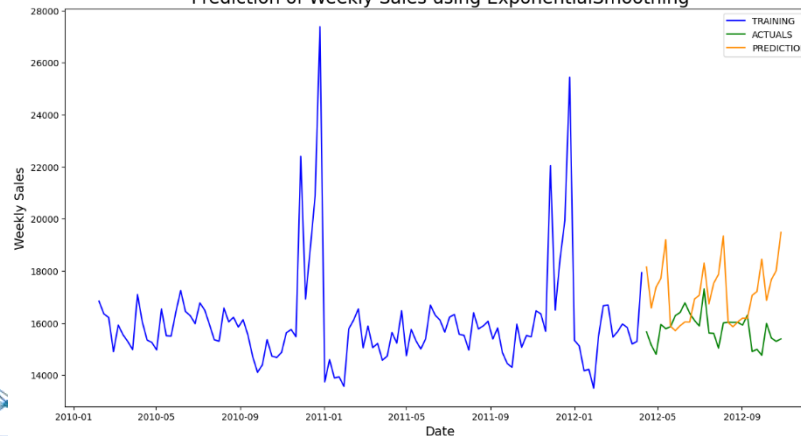
# ผลการทดลองของโมเดล

กราฟแสดงแนวโน้มยอดขายระหว่างค่าที่แท้จริงกับค่าทำนาย

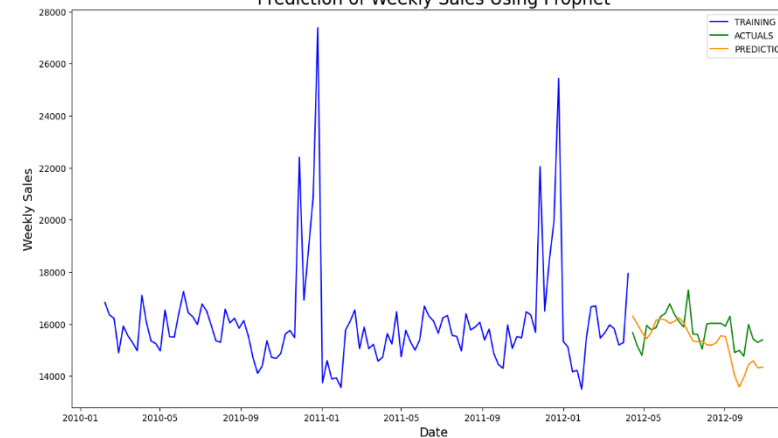
Prediction of Weekly Sales Using Auto-ARIMA



Prediction of Weekly Sales using ExponentialSmoothing



Prediction of Weekly Sales Using Prophet





# ขั้นตอนในการวิเคราะห์ข้อมูล

## Data preparation

### Global super store



GLOBAL SUPERMARKET

#### 1. Import file dataframe

- Global super store
- Stock price
- GDP
- Personal saving rate
- DFF
- CPI
- Unemployment
- Fuel price
- USD

2. แปลงคอลัมน์ 'Date' ในข้อมูล DFF  
ให้อยู่ในรูปแบบ YYYY-MM-DD

3. Merge ข้อมูล Fuel price กับ  
USD เข้าด้วยกันเพื่อแปลงราคา  
น้ำมันจากหน่วย Baht เป็น USD

4. นำ Global Super Store Dataset  
มากรองข้อมูลโดยนำยอดขายเฉพาะ  
Country ที่เป็น 'United States' มา  
ใช้งาน

Global Super Store Dataset : [https://www.kaggle.com/datasets/apoorvaappz/global-super-store-dataset?select=Global\\_Superstore2.csv](https://www.kaggle.com/datasets/apoorvaappz/global-super-store-dataset?select=Global_Superstore2.csv)

Super store Dataset : <https://www.kaggle.com/datasets/vivek468/superstore-dataset-final>

Financial Indicators of US : <https://www.kaggle.com/datasets/mikoajfish99/us-recession-and-financial-indicators?select=Personal+Saving+Rate.csv>

Fuel Price : <https://th.investing.com/currencies/wti-usd>

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Data preparation

### Global super store



GLOBAL SUPERMARKET

5. แปลงคอลัมน์วันที่ให้เป็นคำว่า 'Date' ในทุก dataframe

6. set 'Date' เป็น index เพื่อ resample ข้อมูลให้อยู่ในรูปแบบรายสัปดาห์โดยใช้ค่าเฉลี่ย

- Global super store
- Stock price
- GDP
- Personal saving rate
- DFF
- CPI
- Unemployment
- Fuel price

7. Fill ค่า NaN ใน dataframe GDP ให้แต่ละสามเดือนจะมีค่าเท่ากันเนื่องจากเป็นข้อมูลแบบไตรมาสและ Fill ค่า NaN ใน dataframe Personal saving rate , CPI , Unemployment ให้แต่ละเดือนจะมีค่าเท่ากัน

8. Merge ข้อมูลทุก Data frame เข้าด้วยกัน

Global Super Store Dataset : [https://www.kaggle.com/datasets/apoorvaappz/global-super-store-dataset?select=Global\\_Superstore2.csv](https://www.kaggle.com/datasets/apoorvaappz/global-super-store-dataset?select=Global_Superstore2.csv)

Super store Dataset : <https://www.kaggle.com/datasets/vivek468/superstore-dataset-final>

Financial Indicators of US : <https://www.kaggle.com/datasets/mikoajfish99/us-recession-and-financial-indicators?select=Personal+Saving+Rate.csv>

Fuel Price : <https://th.investing.com/currencies/wti-usd>

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Data preparation

## Global super store



GLOBAL SUPERMARKET

ตารางข้อมูล Global super store

	Sales	Close	GDP	PSAVERT	DFF	CPIAUCSL	UNRATE	Fuel_Price
Date								
2011-01-09	301.143875	54.3560	15351.444	6.9	0.175714	221.187	9.1	2.964
2011-01-16	215.056611	54.4940	15351.444	6.9	0.162857	221.187	9.1	2.984
2011-01-23	127.501481	55.4725	15351.444	6.9	0.174286	221.187	9.1	2.950
2011-01-30	104.924533	56.9800	15351.444	6.9	0.171429	221.187	9.1	2.822
2011-02-06	120.267333	56.0420	15351.444	7.2	0.172857	221.898	9.0	2.938

ข้อมูลอยู่ในช่วงวันที่ 2011-01-09 ถึง 2017-12-31 มีจำนวน 365 แถว , 8 คอลัมน์

# ขั้นตอนในการวิเคราะห์ข้อมูล

## Feature Selection

### Global super store

7 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	50.24	4065.10	63.75	25.10
LSTM	50.26	4054.75	63.67	25.79
Prophet	53.84	4691.52	68.49	28.57

6 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP' , 'PSAVERT' , 'DFF'

6 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	50.35	4089.86	63.95	25.06
LSTM	50.12	4029.52	63.47	25.85
Prophet	54.32	4763.11	69.01	29.52

5 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP' , 'PSAVERT'

5 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	50.25	3972.52	63.02	25.81
LSTM	50.42	4080.49	63.87	25.73
Prophet	54.05	4717.13	68.68	29.30



# ขั้นตอนในการวิเคราะห์ข้อมูล

## Feature Selection

## Global super store

4 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close' , 'GDP'

4 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	50.20	3968.47	62.99	25.92
LSTM	50.10	4147.72	64.40	25.10
Prophet	54.43	4784.83	69.17	29.71

3 ปัจจัย ได้แก่ 'Fuel\_Price' , 'CPI' , 'Close'

3 ปัจจัย	MAE	MSE	RMSE	MAPE
VAR	49.72	3961.99	62.94	26.08
LSTM	50.60	4111.68	64.12	25.76
Prophet	54.55	4789.78	69.20	29.58

	MAE	MSE	RMSE	MAPE
ARIMA	51.90	4233.97	65.05	27.09
ETS	52.70	5231.21	72.32	27.19

# Flask API

```
app = Flask(__name__)
model = load_model('model_with_lstm-input(4)(1).h5')
scaler = pickle.load(open('scaler-input(4).pkl','rb'))
scaler_input = pickle.load(open('X_scaler-input(4).pkl','rb'))

@app.route('/')
def home():
    #return 'Hello World'
    return render_template('home.html')

@app.route('/predict',methods = ['POST'])
def predict():
    # Get input data from request
    int_features = [float(x) for x in request.form.values()] # Extract input features from request

    # Scale the input features
    scaled_features = scaler_input.transform([int_features])
    final_features = np.array(scaled_features).reshape(1, 1, len(int_features)) # Reshape input data to match model's input shape

    # Make prediction using the loaded model
    prediction_scaled = model.predict(final_features)

    # Inverse transform the prediction to get the actual predicted value
    predicted_value = scaler.inverse_transform(prediction_scaled)[0,0]

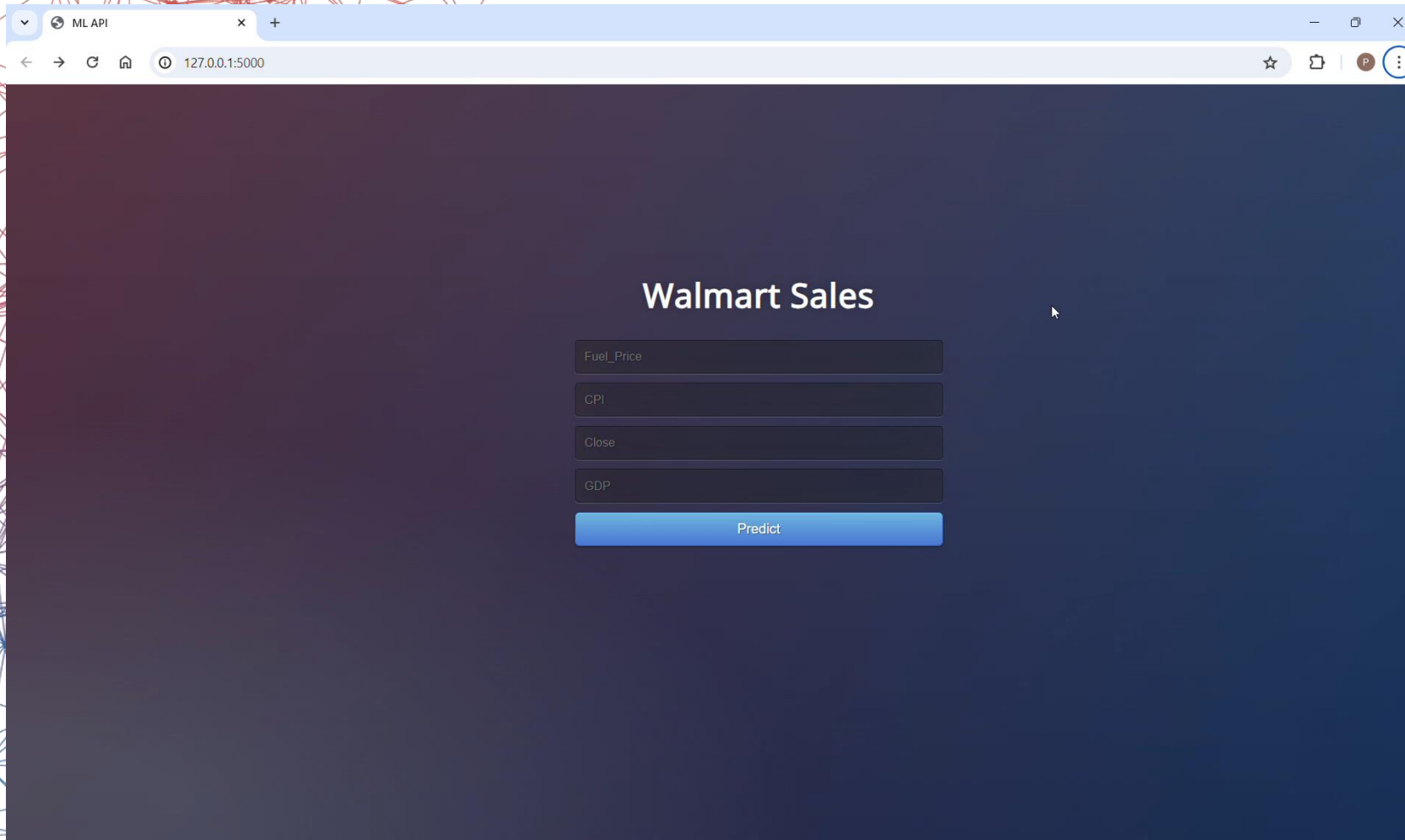
    # Return the prediction as JSON response
    return render_template('home.html', prediction_text="Walmart Sales {:.2f}".format(predicted_value))

@app.route('/predict_api',methods=['POST'])
def predict_api():
    """
    For direct API calls through request
    """
    data = request.get_json(force=True)
    prediction = model.predict([np.array(list(data.values()))])

    output = prediction[0]
    return jsonify(output)

if __name__ == '__main__':
    app.run(debug=True)
```

# Flask API



The screenshot shows a web browser window with a single tab titled "ML API". The address bar displays the URL "127.0.0.1:5000". The main content area has a dark blue gradient background. At the top center, the text "Walmart Sales" is displayed in white. Below this, there are four stacked input fields with the following labels: "Fuel\_Price", "CPI", "Close", and "GDP". Each input field is a dark gray rectangle with its label in a lighter gray font. Below these input fields is a prominent blue button with the text "Predict" in white. The browser's window controls (minimize, maximize, close) are visible in the top right corner, and standard navigation icons (back, forward, refresh, home) are in the top left corner of the address bar area.

ML API

127.0.0.1:5000

## Walmart Sales

Fuel\_Price

CPI

Close

GDP

Predict

## สรุปผลการทดลอง



- การทดลองนี้ใช้ข้อมูลยอดขายและปัจจัยทางเศรษฐกิจจากช่วงเวลาก่อนหน้าเพื่อให้โมเดลเรียนรู้และทำนายยอดขายในอนาคตในช่วง 2012-04-15 ถึง 2012-10-28 พบว่ายอดขายมีความผันผวนขึ้นและลงอยู่ในระดับค่าเฉลี่ยตามข้อมูลจริง
- ปัจจัยที่ส่งผลต่อยอดขายคือวันหยุดพิเศษได้แก่วันขอบคุณพระเจ้า
- เมื่อเข้าสู่เดือนพฤศจิกายนร้านค้าจะมียอดขายที่เพิ่มสูงขึ้นกว่าค่าเฉลี่ยซึ่งเกิดขึ้นในช่วงวันขอบคุณพระเจ้าเนื่องจากเป็นวันที่เฉลิมฉลองกันทั่วทั้งประเทศและใกล้เคียงกับวัน Black Friday ซึ่งเป็นวันที่ร้านค้ามีการจัดการส่งเสริมลดราคาเป็นพิเศษ เพื่อดึงดูดให้ผู้คนออกมาใช้จ่าย รวมถึงมีการเริ่มซื้อของขวัญเพื่อเตรียมตัวสำหรับช่วงเทศกาลคริสต์มาสและปีใหม่จึงทำให้สัปดาห์วันหยุดคริสต์มาสมียอดขายลดลงต่ำกว่าค่าเฉลี่ย
- จากการประเมินค่าความคลาดเคลื่อนจากการทำนายยอดขายพบว่าโมเดล LSTM ให้ค่าเฉลี่ยร้อยละความคลาดเคลื่อนน้อยที่สุดเมื่อเทียบกับโมเดลอื่นที่ 2.20 โดยมีปัจจัยทางเศรษฐกิจที่สำคัญที่ใช้ในการทำนายยอดขายได้แก่ ราคาน้ำมัน, ดัชนีราคาผู้บริโภค, ราคาหุ้นปิดของวัน และผลิตภัณฑ์มวลรวมภายในประเทศ ซึ่งสามารถนำปัจจัยที่สำคัญที่ได้จากการทดลองนี้ไปใช้เพื่อทำนายยอดขายกับชุดข้อมูลร้านค้าอื่นได้



## แนวทางการพัฒนาในอนาคต



1. เพิ่มช่วงระยะเวลาของข้อมูลในการ train model ให้มากขึ้นเพื่อเพิ่มประสิทธิภาพในการเรียนรู้ทำนายยอดขาย
2. เพิ่มปัจจัยภายนอกเพื่อหาปัจจัยที่สำคัญที่อาจส่งผลต่อการทำนายยอดขายในอนาคต

The background features two complex, interconnected wireframe structures. The upper structure is composed of orange lines and dots, while the lower structure is composed of blue lines and dots. These structures overlap and extend across the left side of the image.

T H A N K   Y O U