

Contents

* **Introduction**
* [**Data Extraction**](about:blank)
* [**Data**](about:blank) **Overview**
* **Process breakdown** 
  + **Phase I: Plain Model & Objectives**
    - [**Data Processing**](about:blank)
    - [**Target Variables**](about:blank)
    - [**Predictive Variables**](about:blank)
    - [**Pre-modeling**](about:blank)
    - [**Modeling**](about:blank)
    - [**Algorithmic Solution Results**](about:blank)
  + **Phase II: More Data Preprocessing & Advanced Mode**l
    - [**Data Processing**](about:blank)
    - [**Target Variables**](about:blank)
    - [**Predictive Variables**](about:blank)
    - [**Pre-modeling**](about:blank)
    - [**Modeling**](about:blank)
    - [**Algorithmic Solution Results**](about:blank)
  + **Phase III: Embrace GPT & Improve Forecasting**
    - [**Data Processing**](about:blank)
    - [**Target Variables**](about:blank)
    - [**Predictive Variables**](about:blank)
    - [**Pre-modeling**](about:blank)
    - [**Modeling**](about:blank)
    - [**Algorithmic Solution Results**](about:blank)

Introduction

* This project is essential to IEOR4574: Forecasting: A Real-World Application. In this project, we leverage time series analysis and models which we learned during classes and implement them to extract insights from real-world data. More importantly, it is a learning experience.
* The data we are interested in is an actual online retail transaction data set of two years. We focus on distinct product sales data in the United Kingdom.
* Apply explanatory data analytics to develop a robust approach to develop machine learning models to study stationary and non-stationary components among different products’ sale data
* Standardize a robust methodology (including analytical approach and data requirements) for the development of forecasting that will be leveraged in the coming months

The purpose of this document is to provide a detailed technical overview of:

1. Design specifications and parameters
2. Data inclusions and exclusions
3. Predictive variable creation process
4. Target variable definition
5. Reproducible model-building process
6. Metric evaluation process

Data Extraction

The section explains the source and storage of data.

The whole online retail transaction data is manually pulled from UCI Machine Learning Repository. The primary source is Dr. Daqing Chen, who is MSc in Data Science at London South Bank University.

The format of the data is Microsoft Excel worksheets.

To process data efficiently, data is uploaded and stored in Google Drive. Data are adjusted and integrated using Python scripts

Data Overview

The purpose of this section is to provide overview of data we assigned.

This Online Retail II data set contains all the transactions occurring for a UK-based and registered, non-store online retail between 01/12/2009 and 09/12/2011. The company mainly sells unique all-occasion giftware. Many customers of the company are wholesalers.

All transactions are recorded chronologically, split into two sections based on the time transaction came about and noted on two worksheets, whose names are ‘Year 2009-2010’ and ‘Year 2010-2011’.

The total record count for each worksheet is 525461 and 541911 respectively.

There are 8 attributes that illustrate the details of trades. They are the following:

**InvoiceNo**: Invoice number. Nominal. A 6-digit integral number is uniquely assigned to each transaction. If this code starts with the letter 'c', it indicates a cancellation.

**StockCode**: Product (item) code. Nominal. A 5-digit integral number is uniquely assigned to each distinct product.

**Description**: Product (item) name. Nominal.

**Quantity**: The quantities of each product (item) per transaction. Numeric.

**InvoiceDate**: Invoice date and time down to minutes. Numeric. The day and time when a transaction was generated.

**UnitPrice**: Unit price. Numeric. Product price per unit in sterling (Â£).

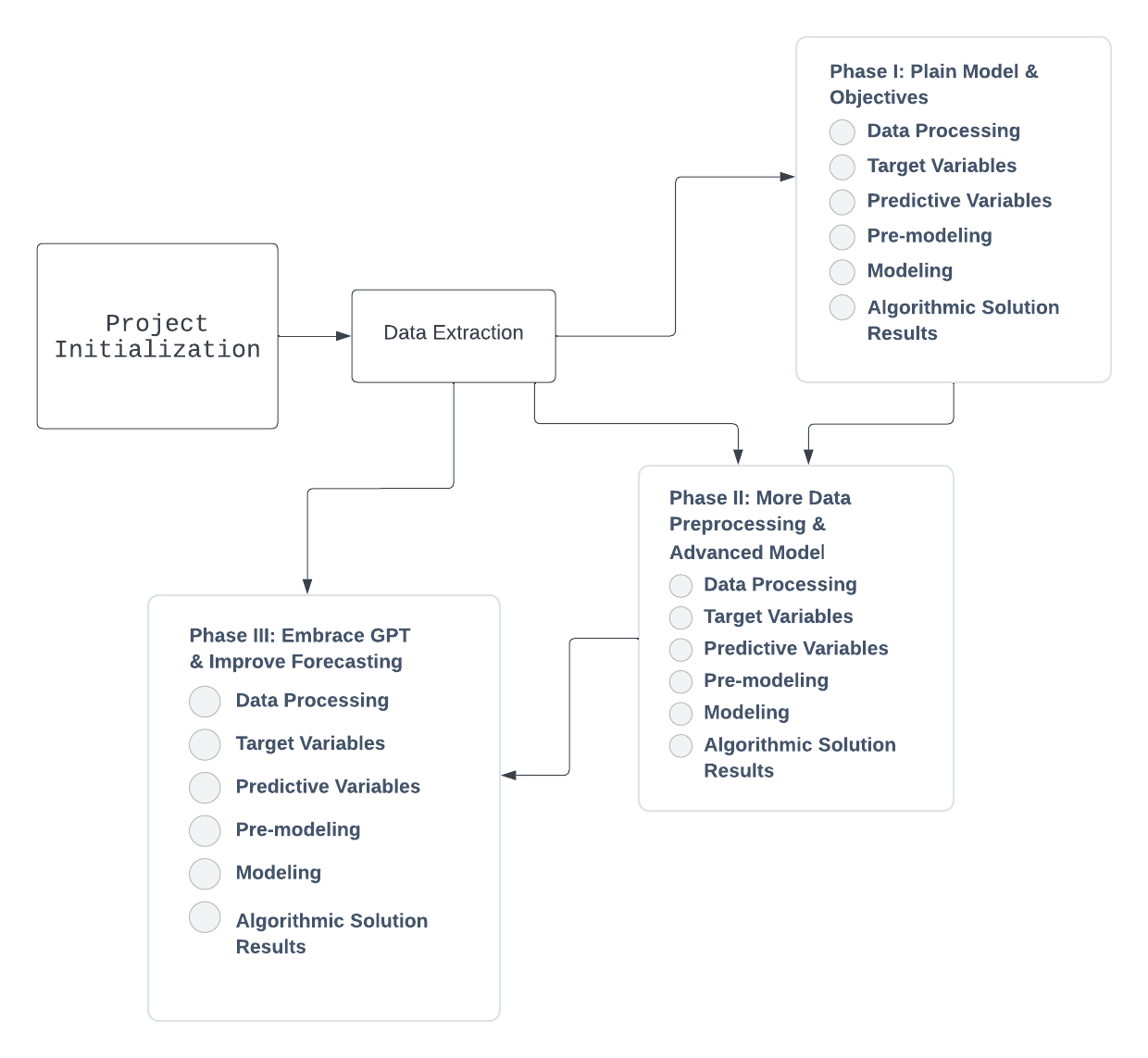
**CustomerID**: Customer number. Nominal. A 5-digit integral number is uniquely assigned to each customer.

**Country**: Country name. Nominal. The name of the country where a customer resides.

**Demo:**

| Invoice | StockCode | Description | Quantity | InvoiceDate | Price | Customer ID | Country |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 489434 | 85048 | 15CM CHRISTMAS GLASS BALL 20 LIGHTS | 12 | 12/1/09 7:45 | 6.95 | 13085 | United Kingdom |
| 489434 | 79323P | PINK CHERRY LIGHTS | 12 | 12/1/09 7:45 | 6.75 | 13085 | United Kingdom |
| 489434 | 79323W | WHITE CHERRY LIGHTS | 12 | 12/1/09 7:45 | 6.75 | 13085 | United Kingdom |
| 489434 | 22041 | RECORD FRAME 7" SINGLE SIZE | 48 | 12/1/09 7:45 | 2.1 | 13085 | United Kingdom |

Process Breakdown



Phase I: Plain Model & Objectives [Data Processing](about:blank)

This section illustrates the nascent stage of project workflow, where data are preprocessed and models are tried out for the first time. Nothing too profound or complicated. But the goal of this phase is to find straightforward methods for forecasting time series given certain constraints.

[Data Processing](about:blank)

**Data diagnostics**

**Invoice** is the first attribute scrutinized. According to attribute information, an **Invoice** is a 6-digit integral number uniquely assigned to each transaction. If this code starts with the letter ‘**C**’, it indicates a cancellation. However, it does not specify what codes with the initial letter ‘**A**’ mean. After inspection, ‘**A**’ indicates adjustments to bad debts.

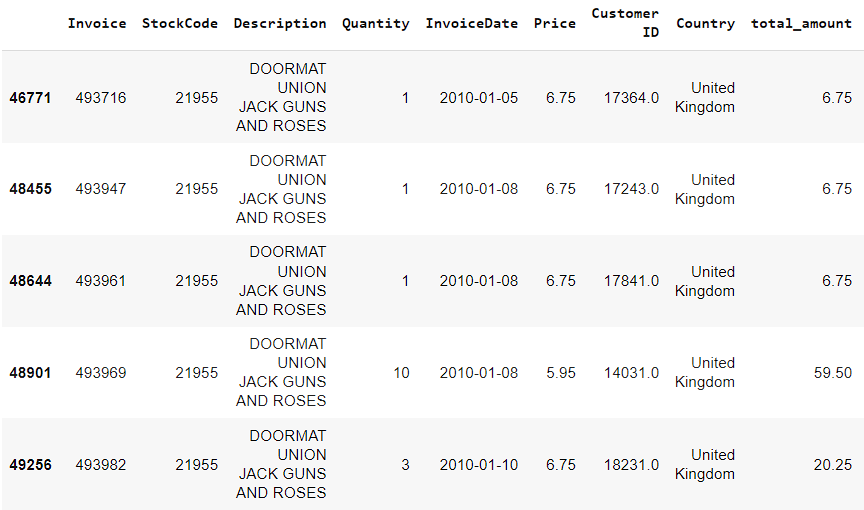


Then, since both **Description** and **StockCode** are signatures of products and each product has its own unique **Description** and **StockCode**, it is necessary to check if the products’ **Description** and **StockCode** are consistent. The consistency of these two attributes is verified by calling unique().

Last but not least, in order to cut to the point and dive into models’ trial and error straight away, some key steps in data cleaning, such as replacing missing values, handling outliers, and adding auxiliary data, are being skipped. **Aggregating data into time periods** is implemented directly.

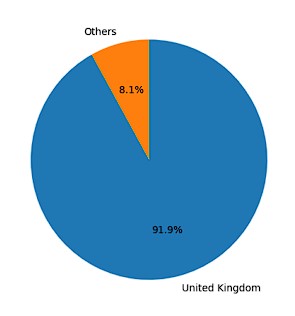
**Data Cleaning**

Firstly, the feature called **total\_amount**, which is calculated by multiplication of **Price** and **Quantity**, is added to the data, because **Total\_amount** is a key market performance indicator.



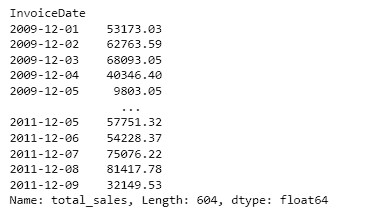
Afterward, **InvoiceDate** is reformatted into datetime down to specific **day**s.

Moreover, data is filtered by **Country**. Those records whose Country is not the **United Kingdom** are dropped off because the majority of records happen within the United Kingdom market.

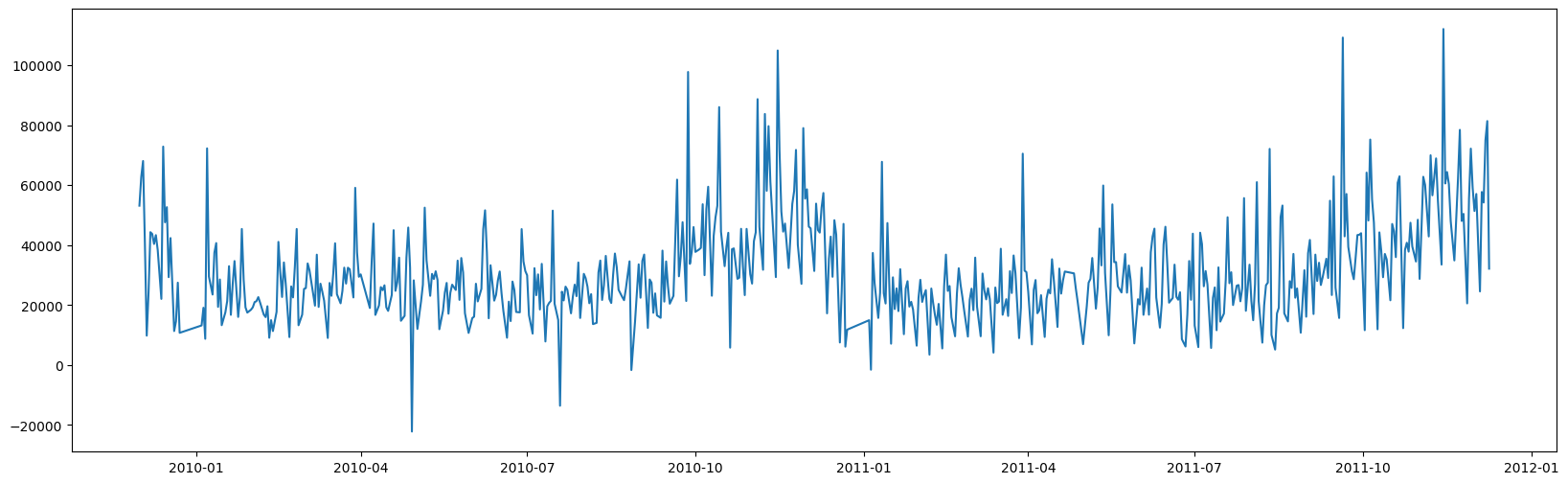


Two worksheet data are processed in accordance with the logic above and merged into one worksheet. Then duplicate transactions are cut off and **total\_amount** is added up by **InvoiceDate**.

The final version is represented in the following table.



**Data Visualization**

****

Target Variable

A target variable is the variable to predict as a result in algorithmic solutions.

**Total\_amount** in sterling (Â£) per InvoiceDate is the **target variable**.

Predictive Variable

A predictive variable is a variable used in algorithmic solutions to predict the target variable. During our analysis, we categorized predictive variables into two categories:

1. Direct variable – These variables were directly from the dataset that was provided by direct customers
2. Derived variable – These variables were created by manipulating the direct variables

**Direct Variable List:**

1. InvoiceDate: Features of time are critical in the modeling because they capture seasonality

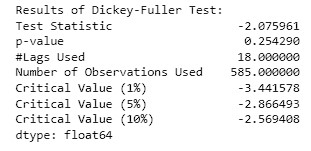
**Derived Variable List:**

1. const: constant dummies
2. lags: serial dependence
3. trend, trend\_squared, trend\_cubed: time trends
4. s(2,7), s(3, 7) …: weekly seasonality
5. sin(23,freq=A-DEC), cos(23,freq=A-DEC) …: annual seasonality
6. Holiday: US national holidays. The holiday has an effect on people’s purchasing patterns.

[Pre-modeling](about:blank)

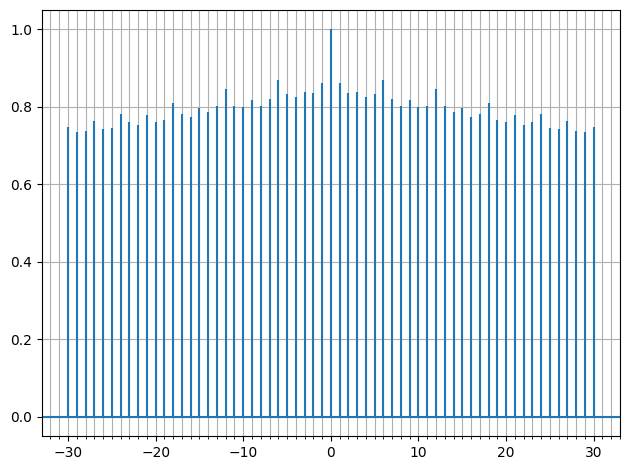
**Data stationarity check:**

Augmented Dickey-Fuller is a statistical significance test, especially widely used for time series stationarity checks. As the Dickey-Fuller test resulted in a p-value a lot higher than 0.05, it means the null hypothesis of non-stationarity cannot be rejected at a 0.05 significance level.

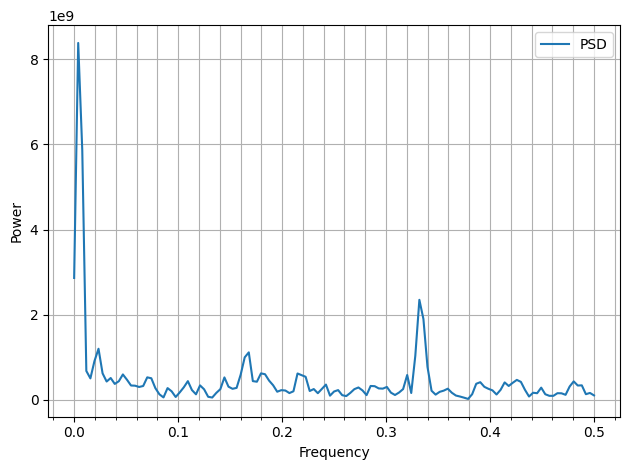


**Data seasonality and trend analysis:**

Since non-stationary factors disturb sales data, the following data seasonality and trend analysis proceeds. Following are the auto-correlation graphs with a maximum lag of 30 and 600 days, As lag increases, the auto-correlation gradually decreases and they perform a wave shape with a frequency of 7 days. That makes sense, since the sales may perform periodicity in the time scale of a single week.



The graph below displays the Power Spectral Density (PSD), which reveals a peak near the frequency of 0.002, which indicates that there exists a seasonality in the sales of a year. Moreover, the second peak is located near the frequency of 0.34.

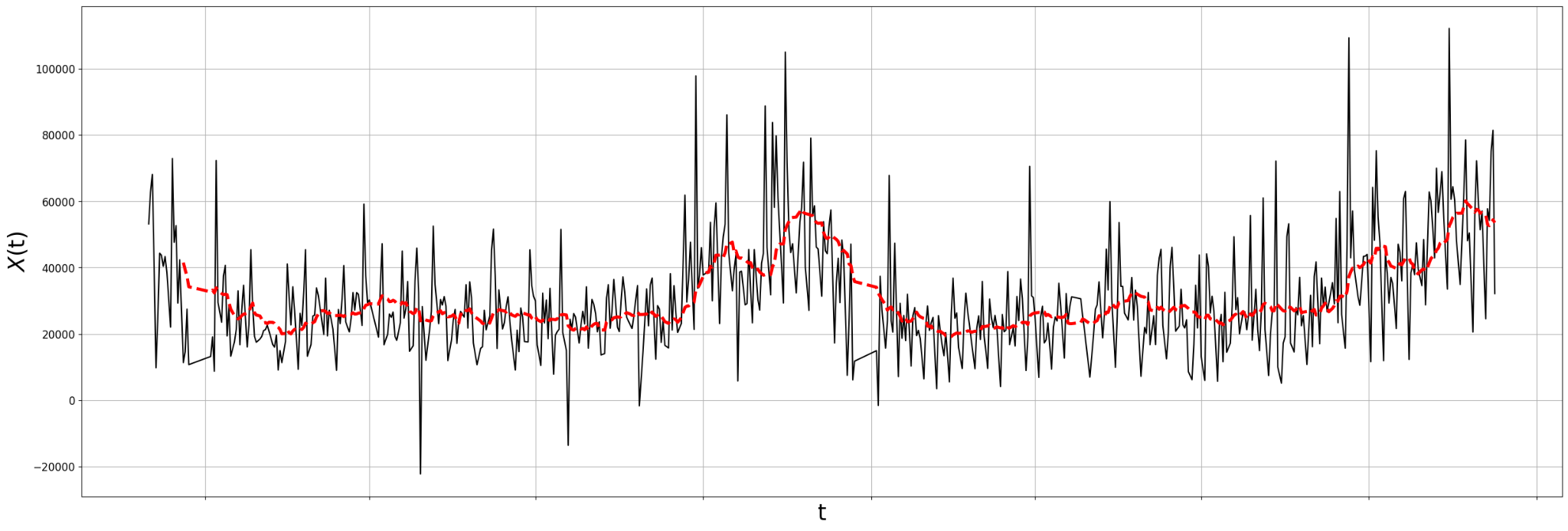


Furthermore, since the second peak in the previous diagram has a frequency of 0.34, sales data might have periodicity in the time scale of 3 days. As a result, by implementing **seasonal\_decompose** and setting the parameter of period equal to 3, the following group of graphs is manufactured.



As shown, there are lots of ups and downs in the trend subfigure and residuals of data after teasing seasonality and trend are approximately normally distributed with a mean equal to 0. Thence, seasonal trends, and stationary residuals are deduced.

To validate the theory that there exists a seasonal trend in sales data, the moving average technique is leveraged. The sliding window is defined as 18 days. The result aligns with the theory.



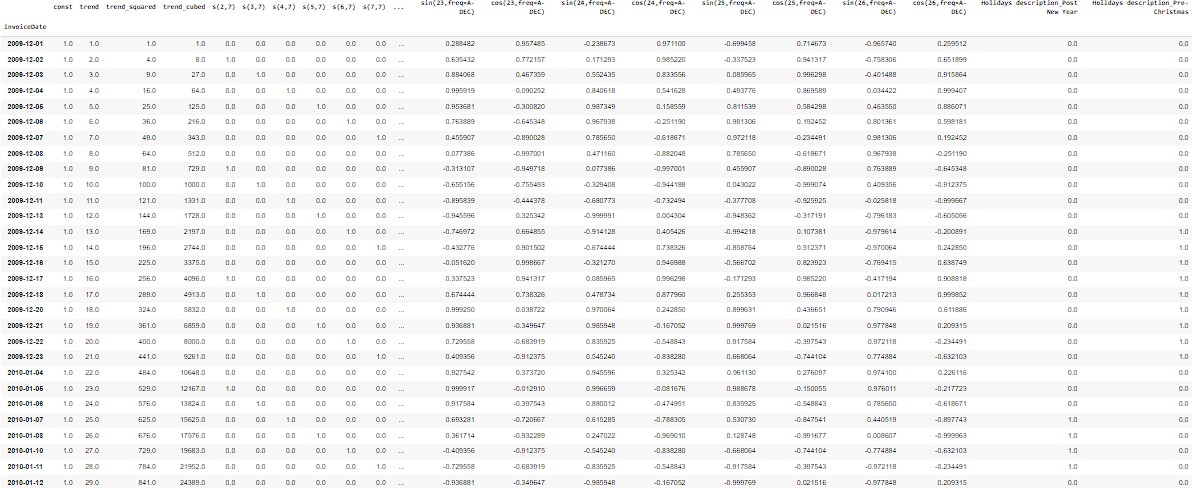
**Derived variable creation:**

Weekly, annual seasonal feature and trend features:

Source: **DeterministicProcess, CalendarFourier** from **statsmodels.tsa.deterministic**

Holiday features:

Source: Manually noting down **two week**s’ dates **prior to Christmas** and another two weeks’ dates **post New Year** and **One-Hot encoding**

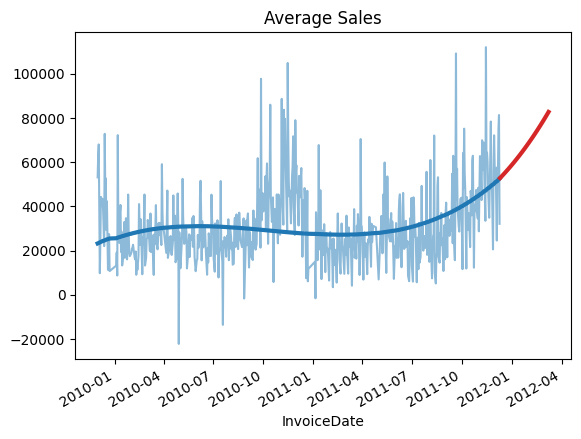


[Modeling](about:blank)

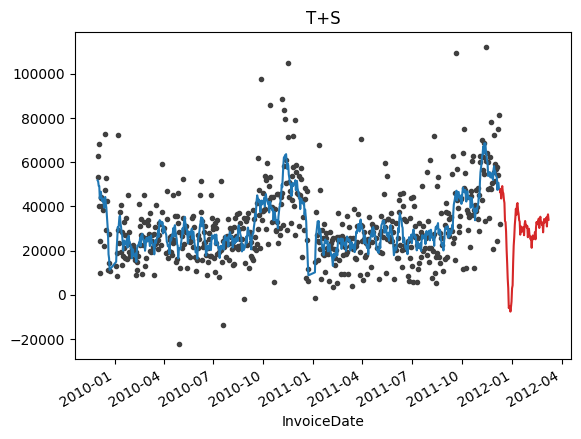
**Linear Regression** and **SARIMAX** are built to predict the UK market daily sale data during this phase.

**Linear Regression:**

The first Linear Regression model is set up to forecasting trends with high-order polynomials. Because neither underfitting nor overfitting is desirable, the order of polynomials is defined as 3. As the following graph demonstrates, the blue line captures significant seasonal trends while the red one represents the model’s prediction on an out-of-sample time frame.

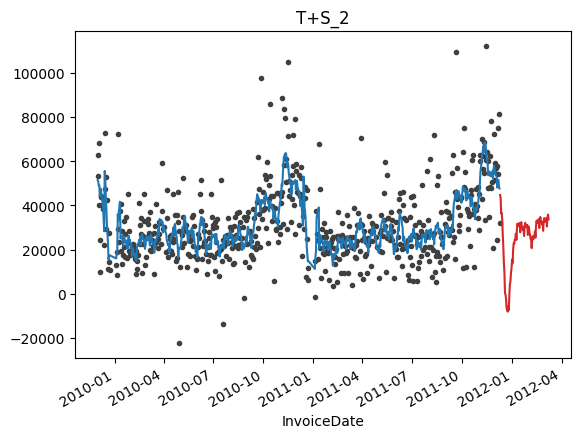


Then, to take account of weekly and annual seasonality, the second Linear Regression model is created and trained on weekly, annual seasonal features and trend features.



The figure above shows this model is capable of learning daily changes in sales data and predicting granularly.

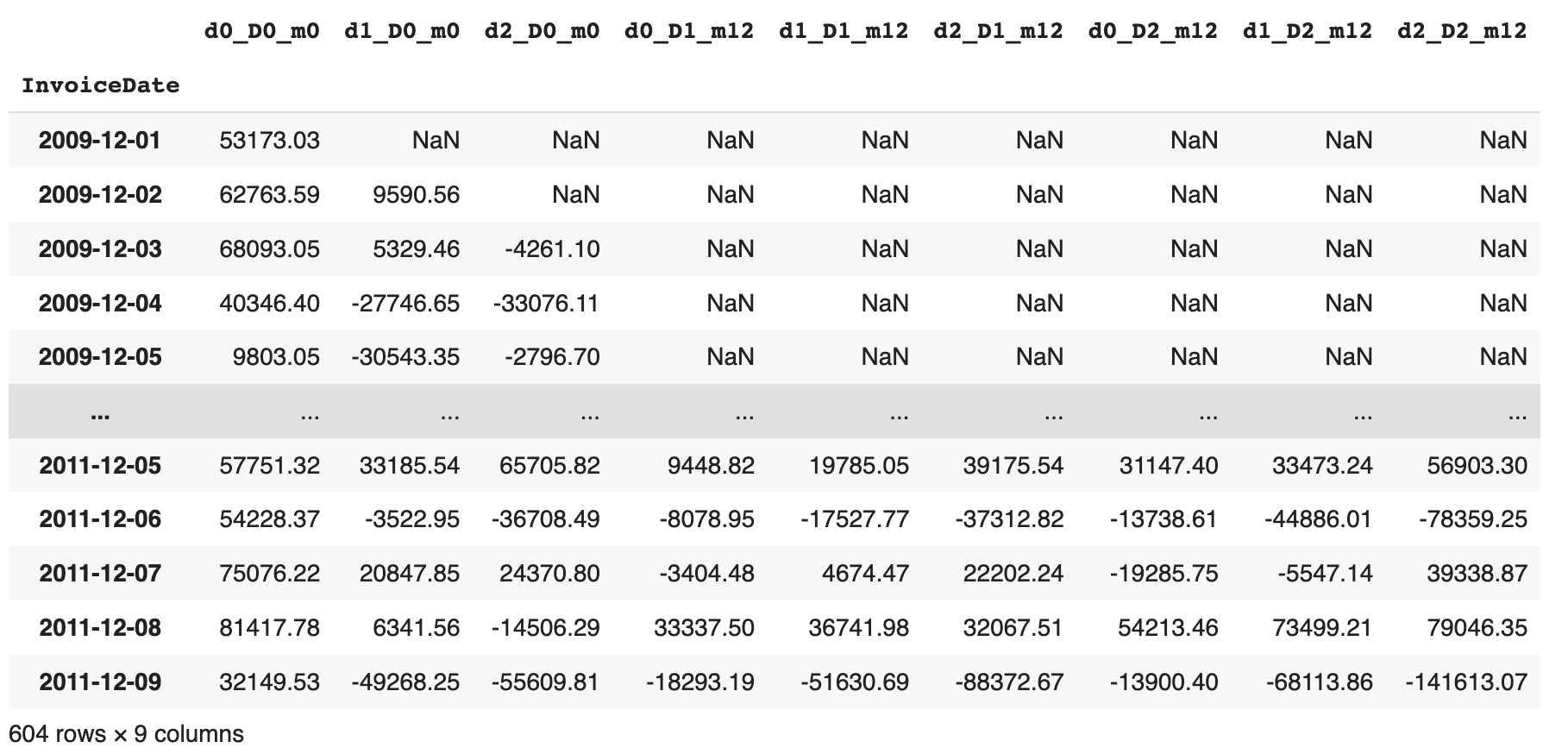
Then with Holiday features coming in, the third Linear Regression model is implemented. This model yields the same result as the second model except for a few nuances.



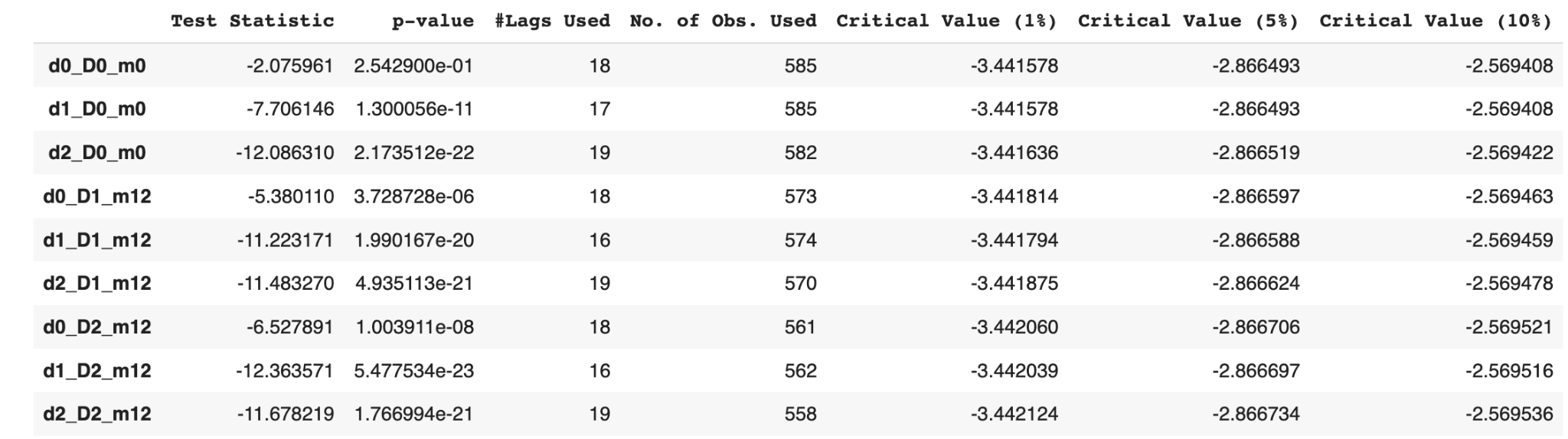
**SARIMAX:**

The parameters of SARIMAX are hard to determine depending on explanatory data analysis. Accordingly, it is necessary to proceed by trial and error.

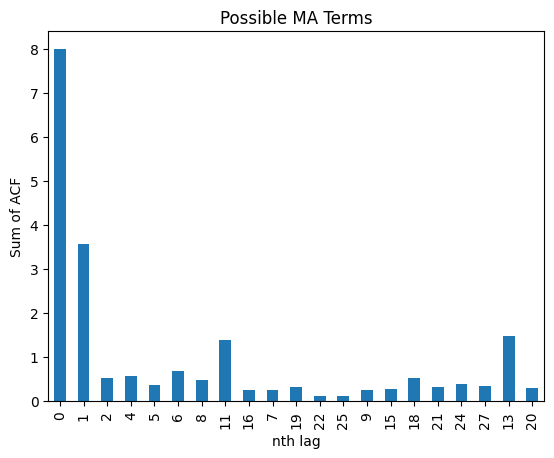
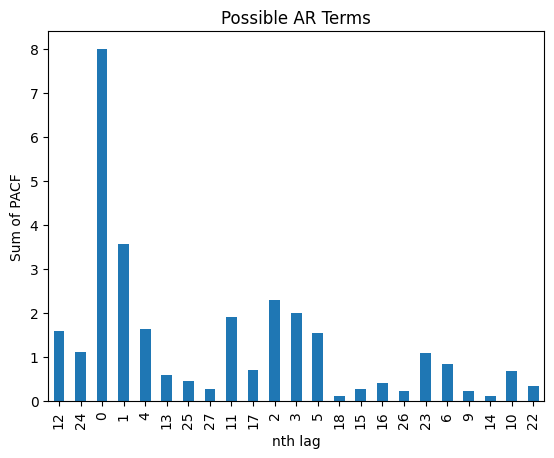
To begin with, all combinations of differencing orders as needed are created to obtain proper d&D indicating the integration order of the process.



Next, the Augmented Dickey-Fuller test is used to filter away results that are not stationary and retain results that are stationary.

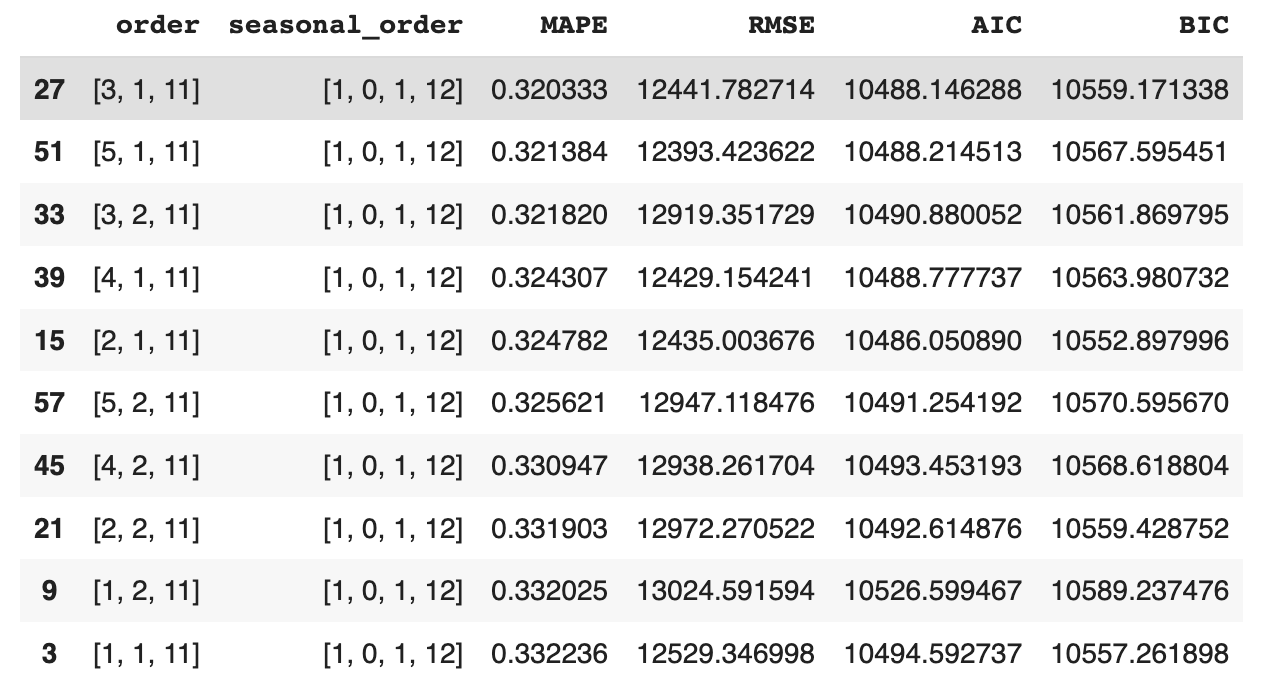


Then, the partial autocorrelation function plays an important role in finding all potential MA and AR terms of the SARIMAX model. All we need is to spot on significant spikes by n-lags in ACF plots and the corresponding horizontal coordinates of spikes are appropriate parameters.

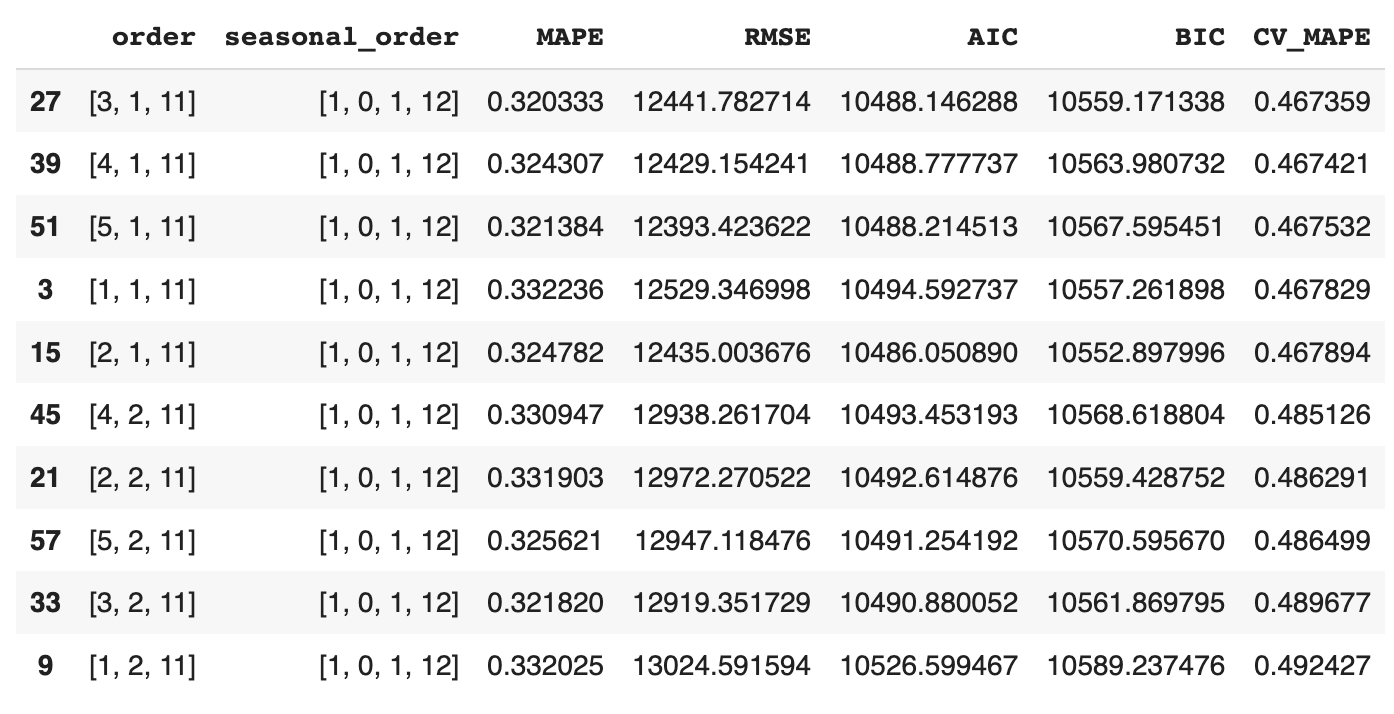


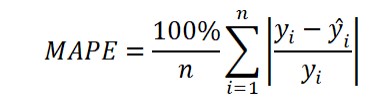
Therefore, p = [1, 2, 3, 4, 5], where P is the AR term and the list consists of all viable values of AR terms. q = [1, 11], where Q is the MA term and the list consists of all viable values of MA terms.

Finally, different SARIMAX models with distinct parameter combinations are founded and trained on the entire sale data.



10 least MAPE models are selected and cross-validated. Ultimately, the results are shown in the table below.





In conclusion, the lowest MAPE out of SARIMAX is 0.32, which is not quite favorable. For the purpose of improving the accuracy of predictions and reducing the Mean Absolute Percentage Error (MAPE) value, the clustering model is involved to group up products within similar categories, and Facebook Prophet/ Random Forest Regression model is considered in the next phase.

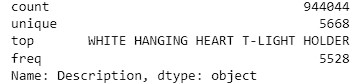
Phase II: More Data Preprocessing & Advanced Model

This section illustrates the intermediate stage of project workflow, where we already have experience as we go through phase I, and we are grinding to promote our forecasting accuracy.

[Data Processing](about:blank)

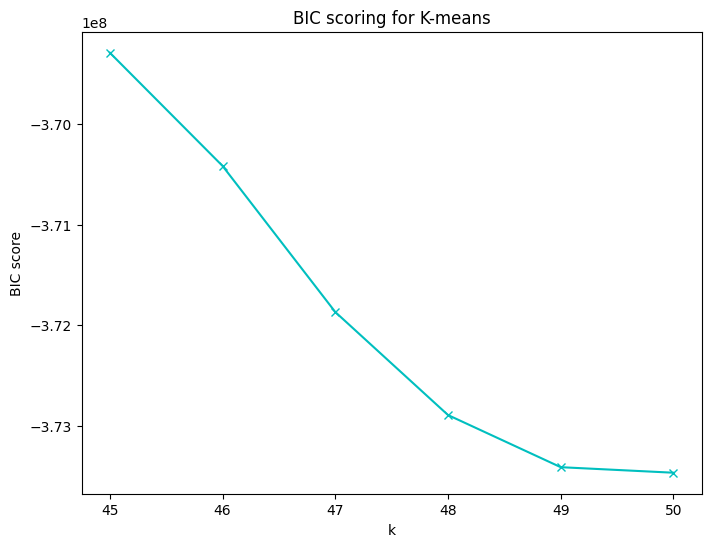
**Data Diagnostics**

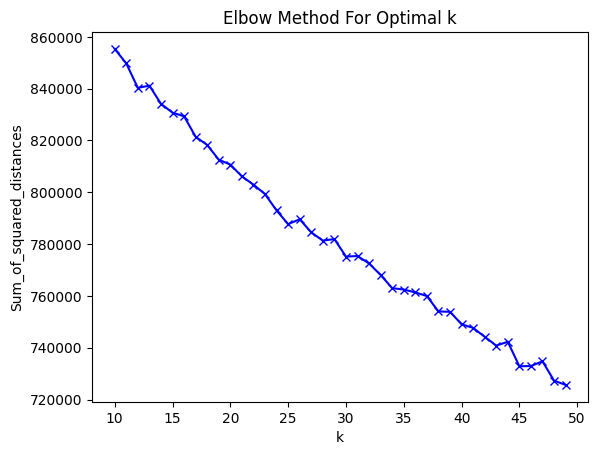
Customers have different needs for different commodities per day. Demands for goods vary. However, some products have similar patterns in demand. So, it makes sense that products are bucketed and each bucket is studied separately according to its own characteristics. Additionally, because the **Description** offers a lot of literal information about products and thence could be perfectly used to compute the degree of similarity of two products based on the similarity, this attribute should be checked if they are applicable.



After breezing through the **Description,** we find that a lot of strings, which contain special redundant characters, need sanitizing.

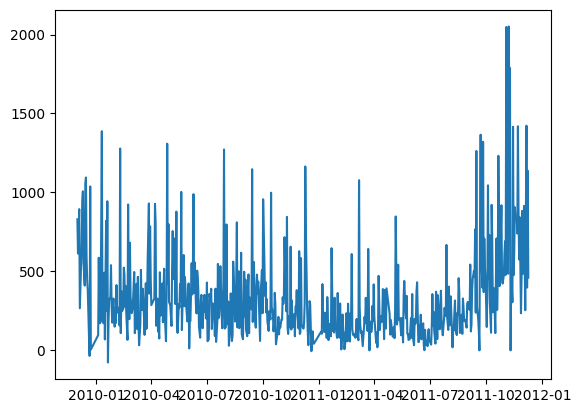
**Data Cleaning**

Procedure in Phase I is replicated excluding data aggregation. Prior to data aggregation, each product’s **Description** is winnowed by the function preprocess\_text() so that the **Description** is literal and useful in differentiating products. For the sake of clustering, transactions with blank under **Description** are deleted. What’s more, the **Description** is vectorized through the TF-IDF calculation and the KMeans Clustering model takes vectorized **Description** as input and produces 47 categories. The metrics to determine why 47 is the number of buckets we need are Elbow Method and BIC scoring. 



Furthermore, sales data is aggregated by **cluster** and **InvoiceDate.** But missing data and outliers are still popping up. Thus, **interpolate()** is called to fix missing data in the time series, and **IsolationForest()** catches outliers and replaces them with reasonable data.

**Sample Data Visualization**



Target Variable

**Total\_amount** in sterling (Â£) per InvoiceDate/ Week is the **target variable**.

Predictive Variable

**Direct Variable List:**

1. InvoiceDate: Features of time are critical in the modeling because they capture seasonality

**Derived Variable List:**

1. const: constant dummies
2. trend, trend\_squared, trend\_cubed: time trends
3. s(2,7), s(3, 7) …: weekly seasonality
4. sin(23,freq=A-DEC), cos(23,freq=A-DEC) …: annual seasonality
5. Holiday: US national holidays. The holiday has an effect on people’s purchasing patterns.
6. clusters

[Pre-modeling](about:blank)

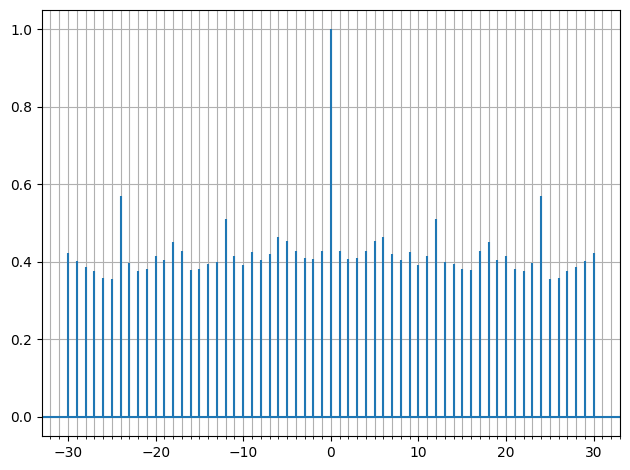
1. Facebook Prophet: simply create a holiday feature by hand



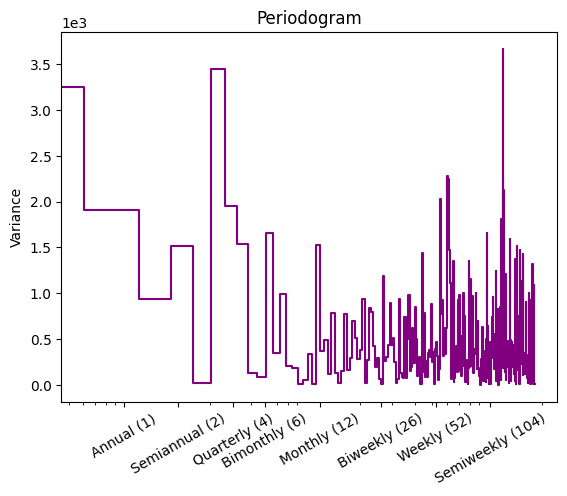
1. Random Forest Regression:

**Data seasonality and trend analysis:**

Since non-stationary factors disturb sales data, the following data seasonality and trend analysis proceeds. Following are the auto-correlation graphs with a maximum lag of 30 and 600 days, As lag increases, the auto-correlation gradually decreases and they perform a wave shape with a frequency of 7 days. That makes sense, since the sales may perform periodicity in the time scale of a single week.



The periodogram gives out the strength of the frequencies in a time series. Obviously, the periodogram below drops off between monthly and biweekly. Therefore, let’s use 18 Fourier pairs.



**Derived variable creation:**

**Weekly, annual seasonal feature and trend features:**

Source: **DeterministicProcess, CalendarFourier from statsmodels.tsa.deterministic**

**Holiday features:**

Source: Manually noting down **two week**s’ dates **prior to Christmas** and another two weeks’ dates **post New Year** and **One-Hot encoding**

**Facebook Prophet prediction features:**

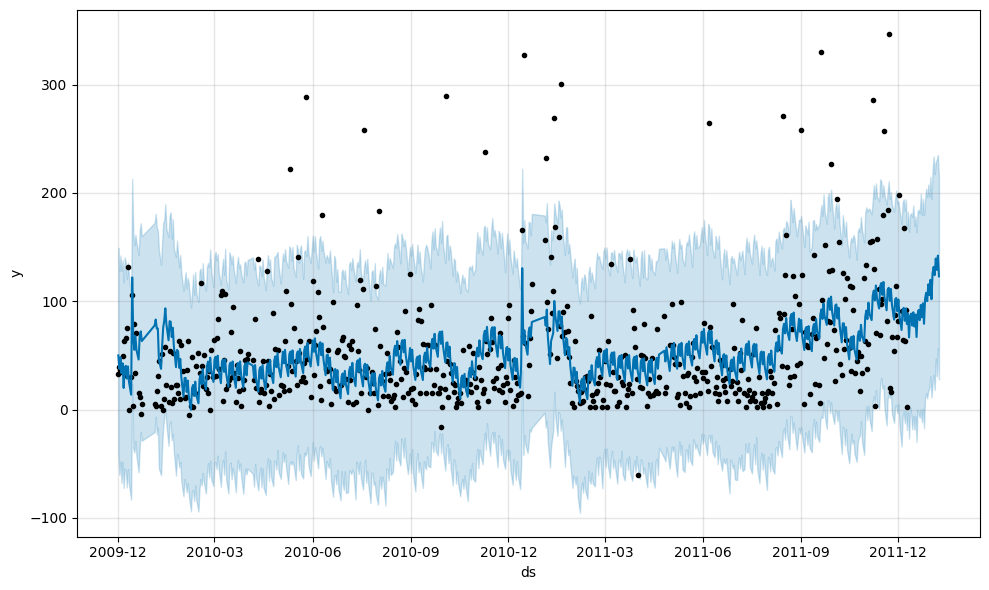
Source: **Prophet.predict() from prophet**

[Modeling](about:blank)

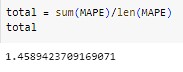
Firstly, we take advantage of **Facebook Prophet** model to make predictions. Secondly, the **Facebook Prophet** model is considered as a baseline and its predictions are treated as input in training the **Random Forest Regression** model.

**Facebook Prophet:**

Entire sales data in each cluster is chopped up into three different parts: train, valid, and test. For each cluster, we build a specific Facebook Prophet model with minimum MAPE after hyperparameter tuning and cross-validation. Further, FB has the functionality to generate future time frames so that we could predict the future even though there is no sales data available yet.

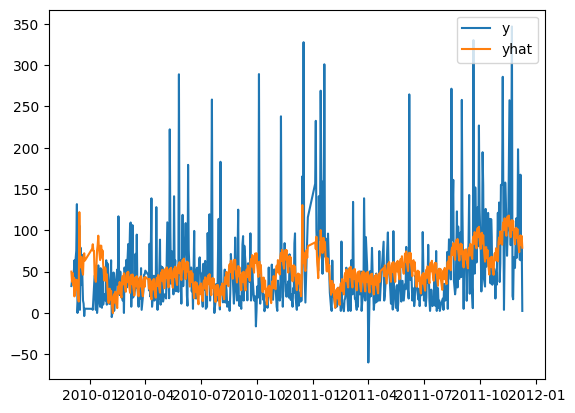


Despite all the effort in clustering and hyperparameters tunning, the final MAPE is far from acceptable.

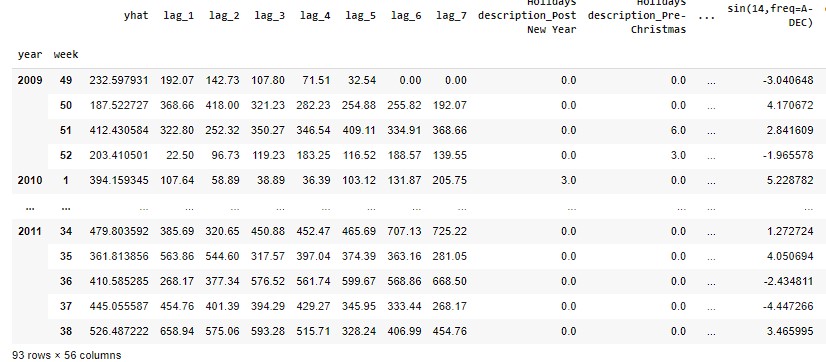


**Random Forest Regression:**

The unsatisfied outcome from Facebook Prophet get me to think if Facebook Prophet is no help in forecasting sales data. Yet, a closer look at FB’s prediction and actual data graph makes me realize that FB is good but not good enough. Thus, we decide to use FB’s predictions as one of the features in training the Random Forest Regression model.



One more little detail before hopping into training models, this time instead of concentrating on daily sales data in each cluster, we shift our focus to weekly sales data created by summing up all features in daily sales data within one week.



Surprisingly, MAPE out of multiple Random Forest Regression models is extremely low.



In hindsight, clustering really works out in terms of promoting models’ prediction accuracy. However, what each cluster stands for is still a disturbing puzzle. Also, 47 clusters may be considered too many for classification purposes. Hence, we decide to give it a try on GPT and Google Bard when clustering in the next phase.

Phase III: Embrace GPT & Improve Forecasting

This section illustrates the ultimate stage of project workflow, where we already have experience as we go through Phase I and Phase II, and we are grinding to promote our forecasting accuracy. To be clear, this phase is called ultimate due to the fact that I already pushed it to my limit. Admittedly, there is a lot to improve, such as pipeline automation.

[Data Processing](about:blank)

**Data Diagnostics & Data Cleaning:**

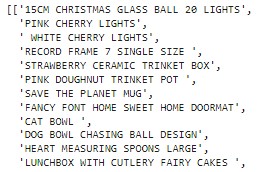
Unlike diagnostics before, this time’s diagnostics is more comprehensive.

Firstly, after inspection, the **Price** of the null **Description** in the records is all zero. As a result, all these records are meaningless and removable.

Secondly, there are several codes starting with the letter ‘**A**’ in the **Invoice.** We pull the records containing special codes up and figure out they represent adjustments on bad debt, which is needed in bookkeeping. Thus, we retain them.



Thirdly, nonsense records are identified by looking at the **Description** and dropped off. Moreover, **the Description** is further cleansed by my getting rid of special characters.



Fourthly, we check if there is abnormal data in **Price.** After analyzing the attribute **Price**,I understand that because price = 0 has no impact on statistics for sales, we could delete all records with unit price = 0.

Eventually, we concatenate two worksheets into one.

Target Variable

**total\_sales** in sterling (Â£) per transaction week is the **target variable**.

Predictive Variable

**Direct Variable List:**

1. InvoiceDate: Features of time are critical in the modeling because they capture seasonality

**Derived Variable List:**

1. const: constant dummies
2. trend, trend\_squared, trend\_cubed: time trends
3. s(2,7), s(3, 7) …: weekly seasonality
4. sin(23,freq=A-DEC), cos(23,freq=A-DEC) …: annual seasonality
5. Holiday: US national holidays. The holiday has an effect on people’s purchasing patterns.
6. clusters

[Pre-modeling](about:blank)

**GPT cluster:**

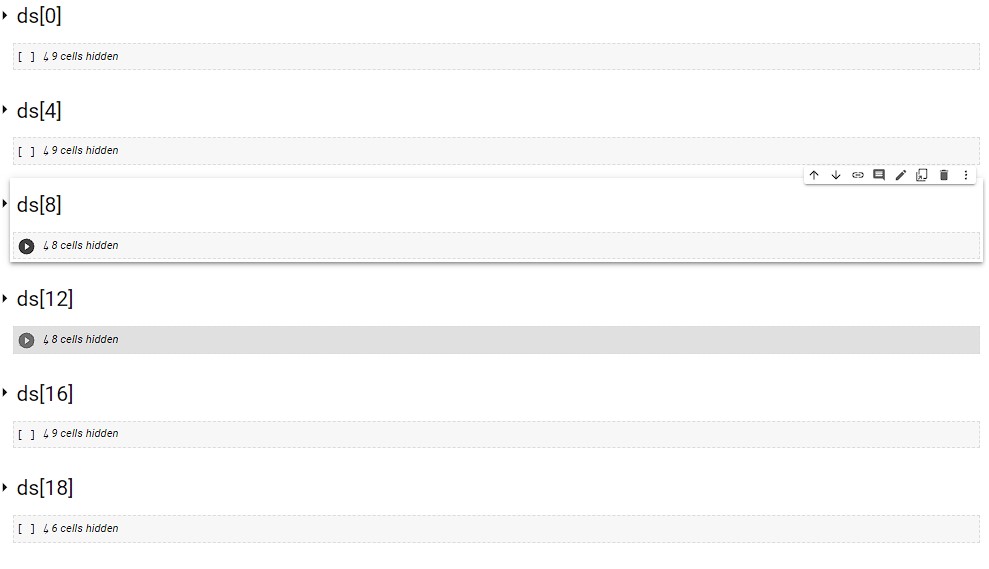
Thanks to Professor Syed Waseem Haider, I am motivated to apply GPT in the work of clustering sales data. Below is a snippet of codes when using the openai API and building up the GPT clustering model.

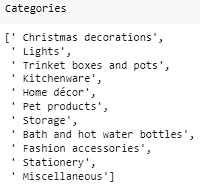


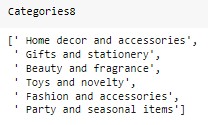
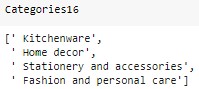
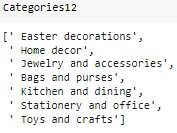
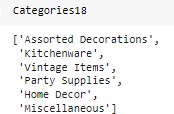
First and foremost, 5385 unique descriptions are extracted from the original sales data and transformed into the format accepted by the GPT-cluster model.

Then, it does not take me a long time to realize that over 5000 messages are way too large for GPT to digest and understand. Consequently, 5385 phrases are divided into 21 groups so that each group could fit into the GPT-cluster model.

Next, to shorten the time spent on clustering and at the same time make clusters as much comprehensive as possible, I sample the first group in every four groups.

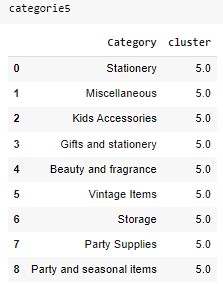


For each sampled group, the GPT-cluster model yields different bunches of categories with meaning. **However, input and output tokens are limited to GPT policies, which means only a small portion of descriptive phrases are clustered.**



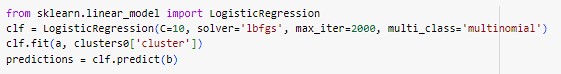
Since the literal meaning of these categories is well extended to cover all product descriptions, it is safe to conclude that **Description** could be classified under the categories listed above. But there are so many overlapping between obtained categories, ergo further cluster is in need.

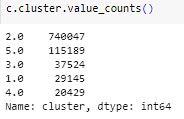
**Google Bard further cluster:**

Since Google Bard API is not open to the public yet, I have to manually type input into the Google Bard online platform, retrieve the [outcome documentation](https://drive.google.com/file/d/17dO9Mf44LT90d5X9pDT-2x5kR32g0s_A/view?usp=share_link), and reverse engineer **Categories** into **cluster.**

**Semi-supervised classification:**

With the help of GPT-cluster and Google Bard, approximately 500 unique descriptive phrases are labeled with **Category** and **cluster**. However, compared to 5385, 500 is just a tiny fraction. Now the problem is evolved into a classification problem and semi-supervised machine learning is the only way out.

Firstly, literal data are vectorized through the TF-IDF calculation, and Logistic Regression is trained on labeled records, then able to recognize similarities among different descriptive phrases under **Description**. Its predictions on unlabeled data fill up blanks under **cluster**. 

The figure beneath shows the distribution of each **cluster**.

**Data aggregation:**

Firstly, sales data is aggregated by **cluster** and **week** which is derived from **InvoiceDate.**

**Data seasonality and trend analysis:**

Seasonality and trend are analyzed by autocorrelation, PSD, seasonal decomposition, and periodogram graphs following the same logic in previous phases. For more details, check out the source code in Trends + seasonality on Jupyter Notebook called GPT/Bard\_Clustering/XGBoost.ipynb.

**Derived variable creation:**

**Seasonal features and trend features:**

Source: **DeterministicProcess, CalendarFourier from statsmodels.tsa.deterministic**

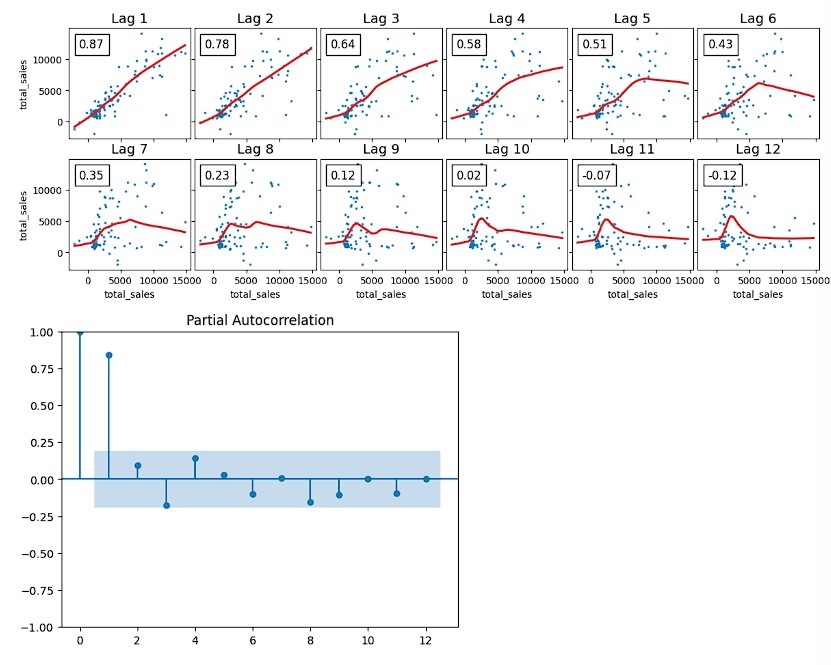
**Holiday features:**

Source: Manually noting down **two week**s’ dates **prior to Christmas** and another two weeks’ dates **post New Year** and **One-Hot encoding**

**Lag features:**

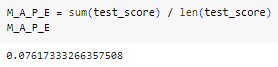
Source:**plot\_pacf from statsmodels.graphics.tsaplots**

**Directly read off from sequential graphs below.**

****

[Modeling](about:blank)

Since each cluster’s weekly sales data has distinct features, it is legitimate to create different models and have them trained on a specific cluster accordingly. For each cluster, we build an XGBoost model with minimum MAPE after hyperparameter tuning and cross-validation. Finally, after all these efforts put in, a little bit of reduction in MAPE is acquired.



Next Step

Pipeline automation would be a research direction worthy of digging deeper into. It would save me a lot of time if GPT and Google Bard are integrated with some handy API so that I could just write Python scripts to implement functions instead of going back and forth between different platforms and programming languages to search for answers.

Another potential research direction is the trade-off between clusters’ explainability and the speed of getting reliable forecasting models.