Project documentation:

Sprint1:

Firstly, required libraries and data are imported. Given files in the dataset which consists of information about 35 stores have been merged. Along with merging the given files, all the anomalies were taken care of. A pair plot (where each variable in the data will be shared in the y axis across a single row and in the x axis across a single column) was created.

A) using correlation heatmap, we visualized the strength of relationship between variables, and we created various subplots and finally identified the key variables.
B) For the first 10 stores we grouped the data by using attributes (store, department)

   and picked top 35 departments (where weekly sales are highest) out of the 100 departments (Assumption).

   We added additional columns (month year and week year) into the data frame just to make the data clear, this has to be done to avoid records having the same week and month number from being merged into one record.

   B.1->Identified the best department in each store by sorting the values according to

   Weekly sales.
C) Here we investigated the relationship between CPI and weekly sales and unemployment over a line plot(which is a way to display data along a number line)
D) With the help of heatmap we analyzed the impact of various types of discounts namely discount promotional, discount clearance, discount damaged good, discount competitive over weekly sales.
E) Using heat map the products which were highly impacted by the external factors (temperature, gas price, holiday) were identified. The correlation between overall sales and is holiday has been identified. Wherever gas prices are less we are observing better weekly sales across all the stores. Is holiday and weekly sales are inversely proportional across the stores.

Sprint2:

DATASET 01:

First things first, we imported the necessary libraries and loaded the csv file from the google drive. Then we appended all the files together where the new data frame has 12 columns.

1) we took only the first ten stores for designing the prediction model.
 a) The scikit-learn library along with the Linear Regression function in Python has been used to create the linear regression model. The training dataset

has been further divided into the training and validation datasets; the validation dataset is primarily created to test models based on different parameters to see what parameters result in higher accuracy. After implementing the linear regression model and generating predictions on both datasets (training and validation), the WMAE (weighted mean absolute error) was calculated (based on the previously defined function in the data cleaning and pre-processing section). Considering that the WMAE values for the validation data is extremely high, linear regression cannot be considered as a very efficient model for this analysis based on this model.

b) Regression, also known as regularized linear regression, is another regression analysis model that handles data that suffer from multicollinearity and is primarily suitable when there are several predictor variables in the data. Lasso regression is an extension of linear regression in which the loss function is modified to minimize the complexity of the model by limiting the sum of the absolute values of the model coefficients. One of the key differences between linear and several other regression models is that while linear regression does not exactly tune the parameters, the other models allow for tuning of the hyperparameter. Just like the linear regression model, the lasso regression model does not perform well on the validation data, giving a relatively higher WMAE value. Hence, the model cannot exactly be considered as an efficient model for the study.

c) Gradient boosting is a sequential technique, based on ensemble learning, which combines the decisions from multiple machine learning models to improve the overall performance of the model. In boosting, the model outcomes for any instance in the sequence are weighed based on the results of the last instance. The correct predictions are assigned a lower weight, while the incorrect predictions are assigned a higher weight. The model will then focus on higher weight points as it might go wrong with the lower weight points. After many iterations, a combined result is generated, using the accuracy of all the models. Using GBM provides multiple advantages: it usually provides better predictive accuracy compared to other models, it provides various hyperparameter tuning options, is time-efficient for larger datasets, and handles missing data. Using the XGBRegressor object from the xgboost library under 'scikit-learn', the training and validation datasets were used to create a basic gradient boosting model and the WMAE was calculated.

d) Arima, predicting sales using Arima is done. ARIMA, short for 'Auto Regressive Integrated Moving Average' is actually a class of models that 'explains' a given time series based on its own past values, that is, its own lags and the lagged forecast errors, so that equation can be used to forecast future values.

Model Performance: out of the all models we tried , ARIMA model with our selected features proved to be the finest for predicting future sales. It had the best RSME,MAE,MAPE values when compared.

Model	MAPE values
Linear Regression	0.15
ARIMA	0.13
XGBoost	0.09

2)As was given only first 10 stores were considered.

a) we used classification algorithms and performed hyper parameter tuning for the deep learning models.

1. Ensemble models (3 statistical methods
2. Recurrent neural network
3. Convolution neural network

RNN works on the principle of saving the output of a particular layer and feeding this back to the input in order to predict the output of the layer.

Deep learning part: In the first step we encoded the categorical variable is_a_holiday and then generated matrix of features (x) and (y).in the next step predictor and target variables were defined and then feature scaling is done. ANN is initialised which was followed by the addition of first layer, second layer and the output layer. The next is compiling and fitting ANN an then prediction is done for single observation.

DATASET 02:

1. Data pre-processing
2. Exploratory data analysis

In the first step merging of files is done ([[using wm_yr_wk]]\nand then that/ price.scsv with data_train.csvs [[using item_id]])) and EDA is performed on the merged dataframe.

(Graphs are attached in the presentation).

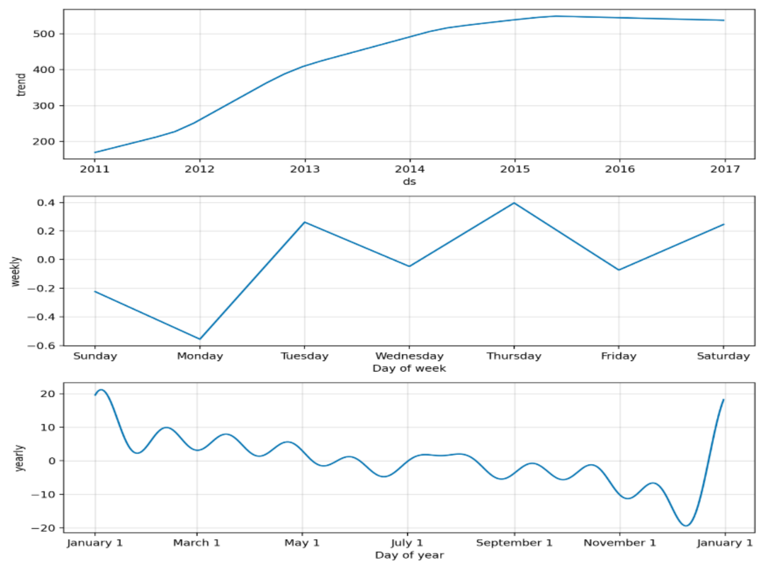
Feature engineering observations: weekly sales are in high in areas where the income median of the individuals is high. People tend to buy more in those areas where we have moderate to mild temperatures.

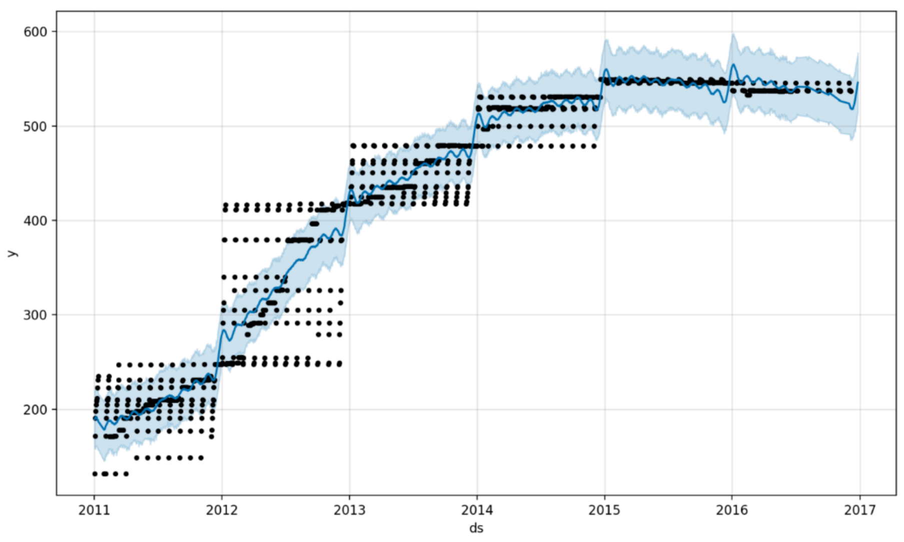
Sprint3:

1. Model Deployment using Streamlit

The ARIMA model developed in the last sprint, was deployed using streamlit in the local environment.

The model is trained with data until december of 2016 and the sales are predicted for the next N days that are passed as inputs to the model.





The actual (black dots) and predicted (blue line) values over time. High level trend of predicted values.

2A. PRODUCT SEGMENTATION based on demand variability (ABC analysis)

What exactly is ABC analysis?
ABC Analysis is a commonly used practice for inventory grouping that divides inventory into categories based on two factors, the cost per unit and the quantity held in stock. ABC Analysis divides the inventory into three major groups, which allows different inventory management techniques to be applied to different segments of the inventory in order to increase the revenue and decrease the cost.

Each category A, B, and C consists of part of the total quantity of the items and represents a part of the total value of the items in the warehouse.

Category A items generally represent approximately 15%-20% of an overall inventory by item, but represent 80% of value of an inventory. Items in this category are goods which annual consumption value is the highest.

Category B items represent 30%-35% of inventory items by item type, and about 15% of the value. The items in this category are goods with medium consumption value.

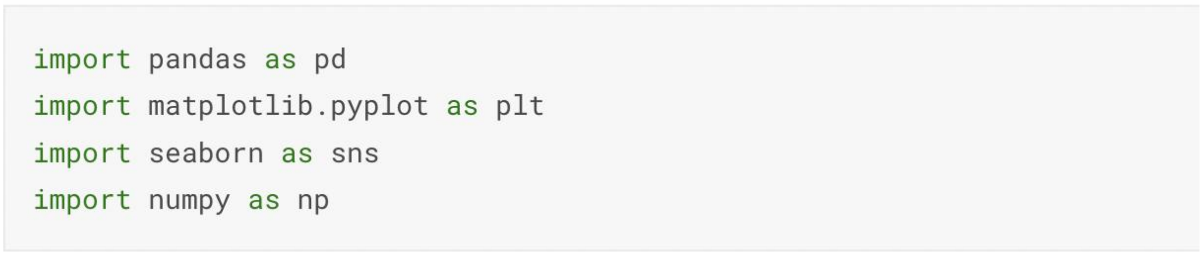
Category C items represent 50% of actual items but only 5% of the inventory value. The items in this category are goods with the lowest consumption value.

FOLLOWING STEPS ARE INVOVLED IN THE ANALYSIS

Step 1: Calculate the annual spend for all the parts by multiplying the unit cost with the annual unit demand
Step 2: Sort inventory in decreasing order of annual spend
Step 3: Calculate the cumulative annual spend and the percent spend
Step 4: Divide the inventory into classes
Step 5: Analyse the classes and make appropriate decisions

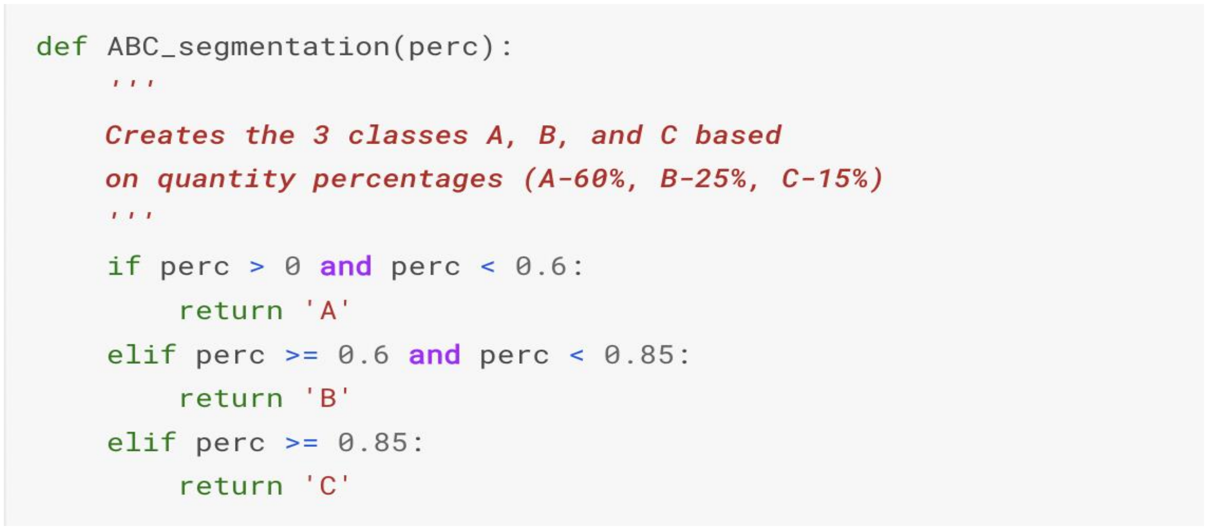
ABC analysis in Action:

- *First things first , all the required libraries were imported.*



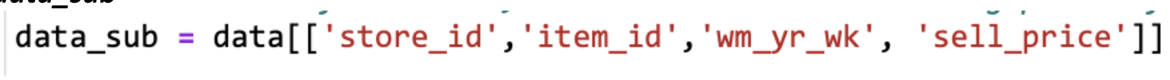
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
```

- *Reading the data from the csv file (sell_prices.csv file)*
- *Defining three classes based on the quantity percentages* ***(A-60% , B-25% , C-15%)***



```
def ABC_segmentation(perc):
    '''
    Creates the 3 classes A, B, and C based
    on quantity percentages (A-60%, B-25%, C-15%)
    '''
    if perc > 0 and perc < 0.6:
        return 'A'
    elif perc >= 0.6 and perc < 0.85:
        return 'B'
    elif perc >= 0.85:
        return 'C'
```

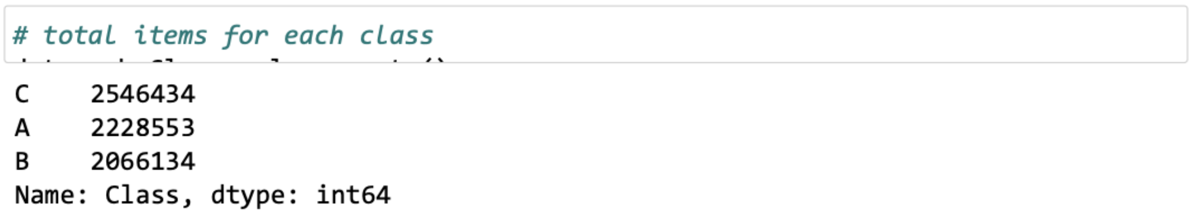
- *Taking a subset of data , we need to use the price and quantity of each item. Here we considered all the four columns of the dataset inorder to create a subset and named it as* ***data_sub***



```
data_sub = data[['store_id','item_id','wm_yr_wk', 'sell_price']]
```

- *Create the column of total sales per item and order it by the cumulative selling price*

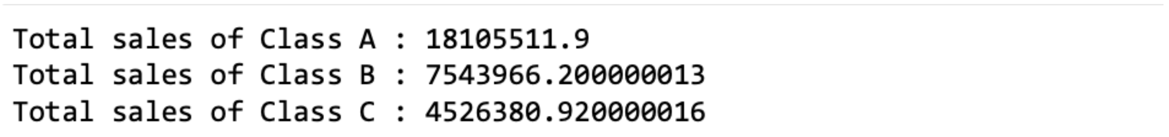
- *Next step , running cum sell_price of the cumulative selling price was created*
- *Column of tot_sum (total sum was created.*
- *In the next step a new column running percentage was created by performing a simple division operation (run_cumsellprice/tot_ sum)*
- *A column class was created where ABC segmentation was applied on running percentage.*



```
# total items for each class
```

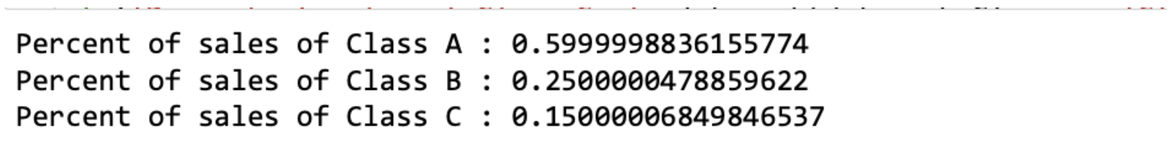
```
C    2546434
A    2228553
B    2066134
Name: Class, dtype: int64
```

- *Total cost per class*



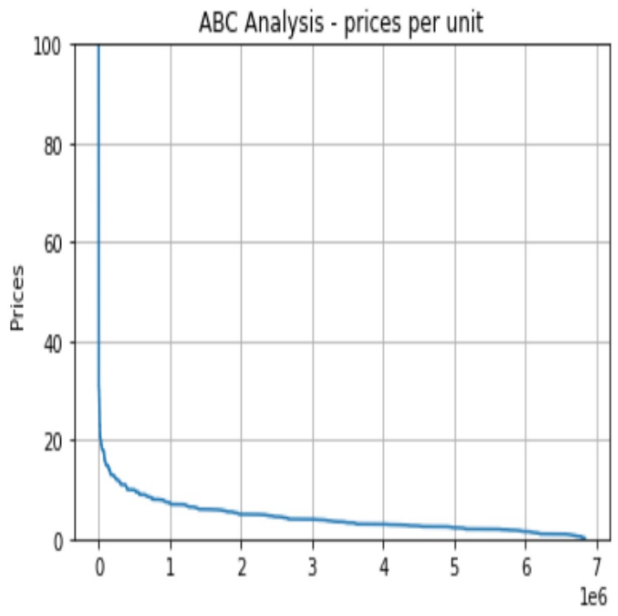
```
Total sales of Class A : 18105511.9
Total sales of Class B : 7543966.200000013
Total sales of Class C : 4526380.920000016
```

- *Percentage of total cost per class*

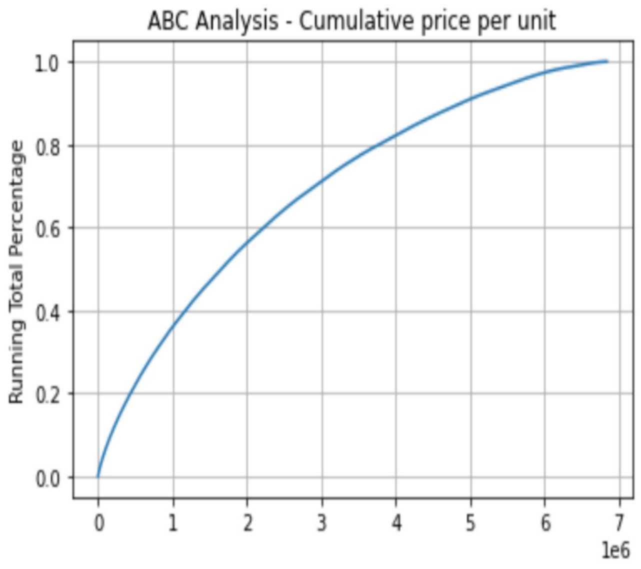


```
Percent of sales of Class A : 0.5999998836155774
Percent of sales of Class B : 0.2500000478859622
Percent of sales of Class C : 0.15000006849846537
```

- *Following graphs show the ABC analysis*
    1. *ABC analysis prices per unit*



2. *ABC analysis cumulative prices per unit*



2B. How stable is the customers' demand? (Coefficient of Variation)

To understand which products will bring planning and distribution challenges, compute the coefficient of variation of the yearly distribution of sales of each reference.

Coefficient of Variation:

The coefficient of variation (CV) is the ratio of the standard deviation to the mean and shows the extent of variability in relation to the mean of the population. The higher the CV, the greater the dispersion.

CV = Standard Deviation / Mean

The coefficient of variation is useful as it is dimensionless (i.e., independent of the unit in which the measurement was taken) and thus, comparable between data sets with different units or widely different means.

Demand Variability

How stable is your customers' demand?

- Average Sales: μ
- Standard Deviation:
- Coefficient of Variation: CV = σ/μ

For SKUs with **a high value of CV**, you may face **unstable customer demand** that would lead to workload peaks, forecasting complexity and stock-outs.

Code

- Filter on the first year of sales for HOBBIES SKU's
- Calculate Mean, Standard deviation and CV of sales
- Sorting (Descending) and Cumulative sales calculation for ABC analysis
- Normalization

Data Pre – processing:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
from scipy.stats import shapiro
```

Import Sales Evaluation:

```python
# Columns
COLS_ITM = ['id', 'item_id', 'dept_id', 'cat_id', 'store_id', 'state_id']
COLS_DATE = ['d_' + str(i) for i in range(1, 366)]
```

```python
df = pd.read_csv(PATH + '/sales_train_evaluation.csv')
print("{:,} records for train data set".format(len(df)))
# Change id to match with price dataframe
df['id'] = df['store_id'] + '-' + df['item_id']
# Set index
df.set_index(COLS_ITM, inplace = True)
# Scope = Year 1
df = df[COLS_DATE]

df.head()
```

```
30,490 records for train data set
```

						d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8	d_9	d_10	...	d_356	d_357	d_358	d_359	d_360
id	item_id	dept_id	cat_id	store_id	state_id																
CA_1-HOBBIES_1_001	HOBBIES_1_001	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
CA_1-HOBBIES_1_002	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	1	0
CA_1-HOBBIES_1_003	HOBBIES_1_003	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0
CA_1-HOBBIES_1_004	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	0	0	0	0	0	...	0	1	1	8	1
CA_1-HOBBIES_1_005	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	0	0	0	0	0	0	...	1	0	0	0	0

5 rows × 365 columns

Import Calendar:

```python
# Import Calendar
df_calendar = pd.read_csv(PATH + '/calendar.csv')
# Date to Week
dict_week = dict(zip(df_calendar.d.values, df_calendar.wm_yr_wk.values))
df_calendar.head()
```

	date	wm_yr_wk	weekday	wday	month	year	d	event_name_1	event_type_1	event_name_2	event_type_2	snap_CA	snap_TX	snap_WI
0	2011-01-29	11101	Saturday	1	1	2011	d_1	NaN	NaN	NaN	NaN	0	0	0
1	2011-01-30	11101	Sunday	2	1	2011	d_2	NaN	NaN	NaN	NaN	0	0	0
2	2011-01-31	11101	Monday	3	1	2011	d_3	NaN	NaN	NaN	NaN	0	0	0
3	2011-02-01	11101	Tuesday	4	2	2011	d_4	NaN	NaN	NaN	NaN	1	1	0
4	2011-02-02	11101	Wednesday	5	2	2011	d_5	NaN	NaN	NaN	NaN	1	0	1

Import Pricing:

```
# Import
df_price = pd.read_csv(PATH + '/sell_prices.csv')
print("{:,} records for sales price".format(len(df_price)))
# SKU Index
df_price['item_store_id'] = df_price['store_id'] + '-' + df_price['item_id']
# Pricing
df_price = df_price.pivot(index='item_store_id', columns='wm_yr_wk', values='sell_price').fillna(0)
# Matrix from pivot
matrix_price = df_price.to_numpy()
# Dict Matrix Index
pr_n = dict(zip(df_price.index, range(len(df_price.index))))
pr_p = dict(zip(df_price.columns, range(len(df_price.columns))))
print("{:,} records for sales price pivot".format(len(df_price)))
df_price.head()
```

```
6,841,121 records for sales price
30,490 records for sales price pivot
```

wm_yr_wk	11101	11102	11103	11104	11105	11106	11107	11108	11109	11110	...	11612	11613	11614	11615	11616	11617	11618	11619	11620	11621
item_store_id																					
CA_1-FOODS_1_001	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	2.00	...	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24	2.24
CA_1-FOODS_1_002	7.88	7.88	7.88	7.88	7.88	7.88	7.88	7.88	7.88	7.88	...	9.48	9.48	9.48	9.48	9.48	9.48	9.48	9.48	9.48	9.48
CA_1-FOODS_1_003	2.88	2.88	2.88	2.88	2.88	2.88	2.88	2.88	2.88	2.88	...	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23	3.23
CA_1-FOODS_1_004	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	...	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96	1.96
CA_1-FOODS_1_005	2.94	2.94	2.94	2.94	2.94	2.94	2.94	2.94	2.94	2.94	...	3.54	3.54	3.54	3.54	3.54	3.54	3.54	3.54	3.54	3.54

5 rows × 282 columns

Statistical Analysis:

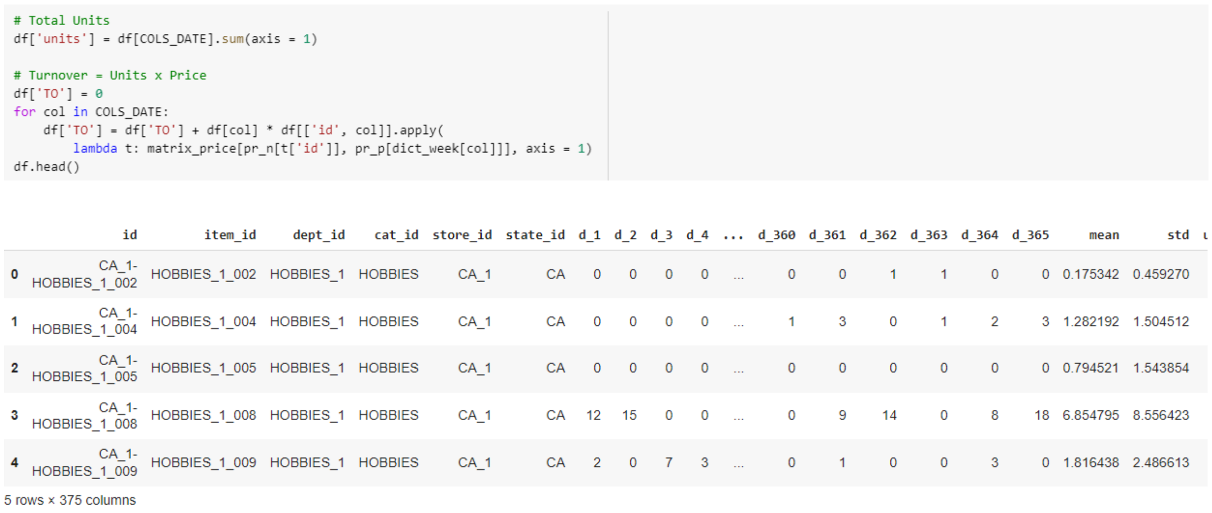
```
# Mean
df['mean'] = df[COLS_DATE].mean(axis = 1)
# Standard
df['std'] = df[COLS_DATE].std(axis = 1)

# Remove items not sold during the first year
print("{:,} records for the full scope".format(len(df)))
df = df[df['mean']>0]
print("{:,} records for after filter".format(len(df)))
df.reset_index(inplace = True)
# Sigma
df.head()
```

```
30,490 records for the full scope
17,055 records for after filter
```

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_358	d_359	d_360	d_361	d_362	d_363	d_364	d_365	me
0	CA_1-HOBBIES_1_002	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	0	0	1	1	0	0	0.1753
1	CA_1-HOBBIES_1_004	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	8	1	3	0	1	2	3	1.2821
2	CA_1-HOBBIES_1_005	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	0	0	0	0	0	0.7945
3	CA_1-HOBBIES_1_008	HOBBIES_1_008	HOBBIES_1	HOBBIES	CA_1	CA	12	15	0	0	...	20	26	0	9	14	0	8	18	6.8547
4	CA_1-HOBBIES_1_009	HOBBIES_1_009	HOBBIES_1	HOBBIES	CA_1	CA	2	0	7	3	...	1	0	0	1	0	0	3	0	1.8164

5 rows × 373 columns

Turnover Sales:

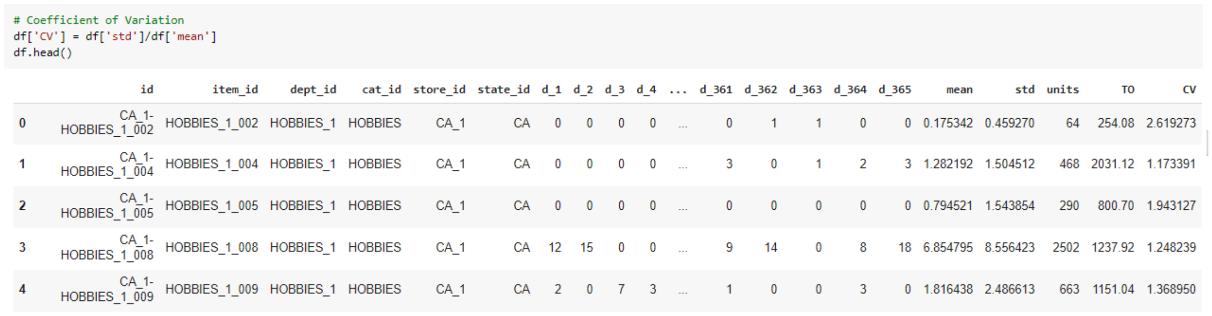


```python
# Total Units
df['units'] = df[COLS_DATE].sum(axis = 1)

# Turnover = Units x Price
df['TO'] = 0
for col in COLS_DATE:
    df['TO'] = df['TO'] + df[col] * df[['id', col]].apply(
        lambda t: matrix_price[pr_n[t['id']], pr_p[dict_week[col]]], axis = 1)
df.head()
```

	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_360	d_361	d_362	d_363	d_364	d_365	mean	std	u
0	CA_1-HOBBIES_1_002	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	1	1	0	0	0.175342	0.459270	
1	CA_1-HOBBIES_1_004	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	1	3	0	1	2	3	1.282192	1.504512	
2	CA_1-HOBBIES_1_005	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	0	0	0	0.794521	1.543854	
3	CA_1-HOBBIES_1_008	HOBBIES_1_008	HOBBIES_1	HOBBIES	CA_1	CA	12	15	0	0	...	0	9	14	0	8	18	6.854795	8.556423	
4	CA_1-HOBBIES_1_009	HOBBIES_1_009	HOBBIES_1	HOBBIES	CA_1	CA	2	0	7	3	...	0	1	0	0	3	0	1.816438	2.486613	

```
5 rows × 375 columns
```

Coefficient of Variation : $CV = \sigma/\mu$



```python
# Coefficient of Variation
df['CV'] = df['std']/df['mean']
df.head()
```

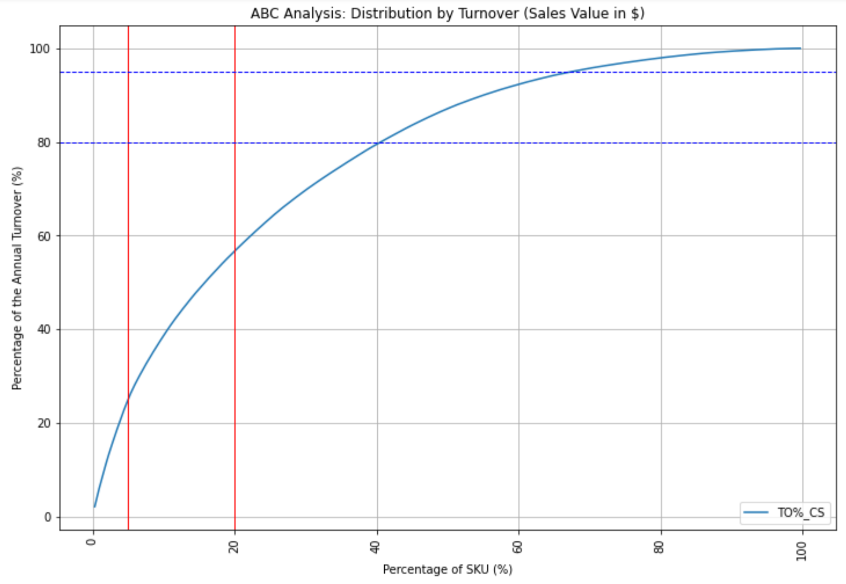
	id	item_id	dept_id	cat_id	store_id	state_id	d_1	d_2	d_3	d_4	...	d_361	d_362	d_363	d_364	d_365	mean	std	units	TO	CV
0	CA_1-HOBBIES_1_002	HOBBIES_1_002	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	1	1	0	0	0.175342	0.459270	64	254.08	2.619273
1	CA_1-HOBBIES_1_004	HOBBIES_1_004	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	3	0	1	2	3	1.282192	1.504512	468	2031.12	1.173391
2	CA_1-HOBBIES_1_005	HOBBIES_1_005	HOBBIES_1	HOBBIES	CA_1	CA	0	0	0	0	...	0	0	0	0	0	0.794521	1.543854	290	800.70	1.943127
3	CA_1-HOBBIES_1_008	HOBBIES_1_008	HOBBIES_1	HOBBIES	CA_1	CA	12	15	0	0	...	9	14	0	8	18	6.854795	8.556423	2502	1237.92	1.248239
4	CA_1-HOBBIES_1_009	HOBBIES_1_009	HOBBIES_1	HOBBIES	CA_1	CA	2	0	7	3	...	1	0	0	3	0	1.816438	2.486613	663	1151.04	1.368950

Product Segmentation:

The inventory is divided into three main groups by analysis, allowing different inventory management strategies to be applied to various inventory segments in an effort to boost revenue and cut costs.

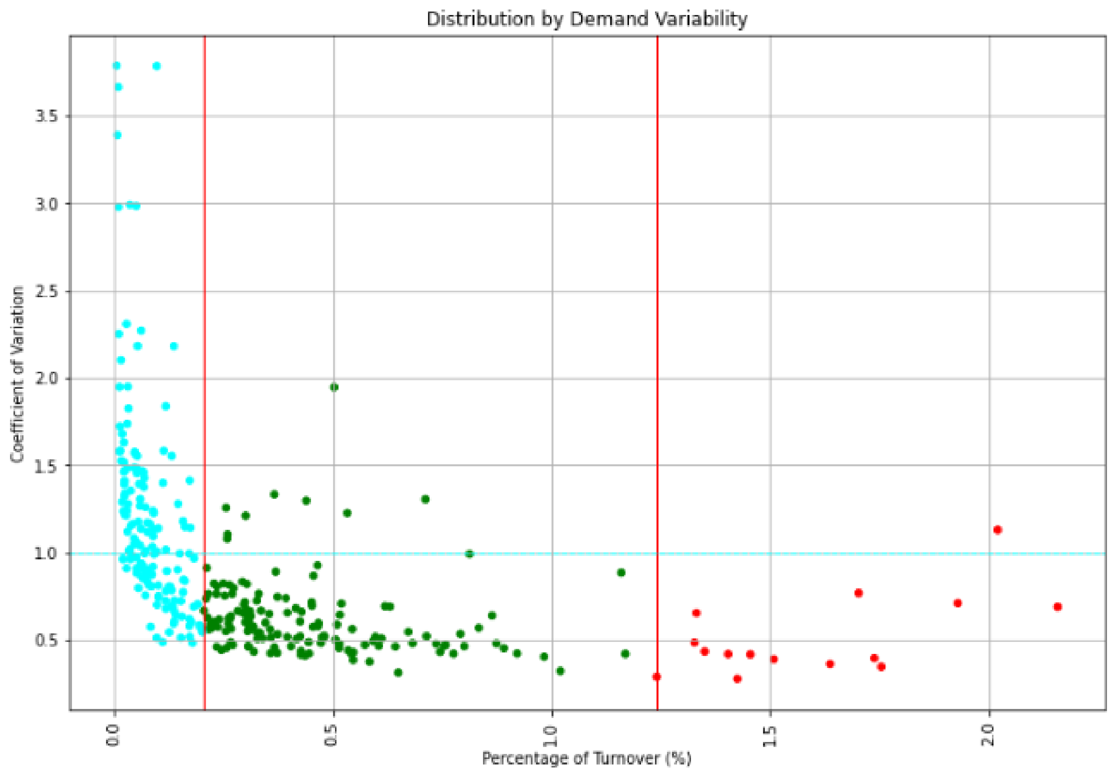
```
Class A: the top 5%
- Number of SKU: 16
- Turnover (%): 25%

Class B: the following 15%
- Number of SKU: 48
- Turnover (%): 31%

Class C: the 80% slow movers
- Number of SKU: 253
- Turnover (%): 43%
```

```python
# Distribution by Value
ax = plt.gca()
df_abc.plot(figsize=(12, 8), x='SKU_%', y='TO%_CS', ax =ax, grid = True)
# ABC
# 20%, 50% of SKU Number
ax.axvline(5 , color="red", linestyle="-", linewidth = 1.0)
ax.axvline(20 , color="red", linestyle="-", linewidth = 1.0)
# 20%, 50% of SKU Number
ax.axhline(80 , color="blue", linestyle="--", linewidth = 1.0)
ax.axhline(95 , color="blue", linestyle="--", linewidth = 1.0)
plt.xlabel('Percentage of SKU (%)')
plt.xticks(rotation=90)
plt.ylabel('Percentage of the Annual Turnover (%)')
plt.title('ABC Analysis: Distribution by Turnover (Sales Value in $)')
plt.show()
```

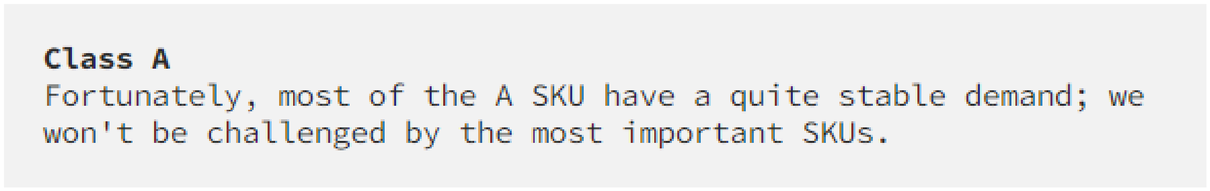


Segmentation by Demand Variability | Coefficient of Variation = f (% TO)

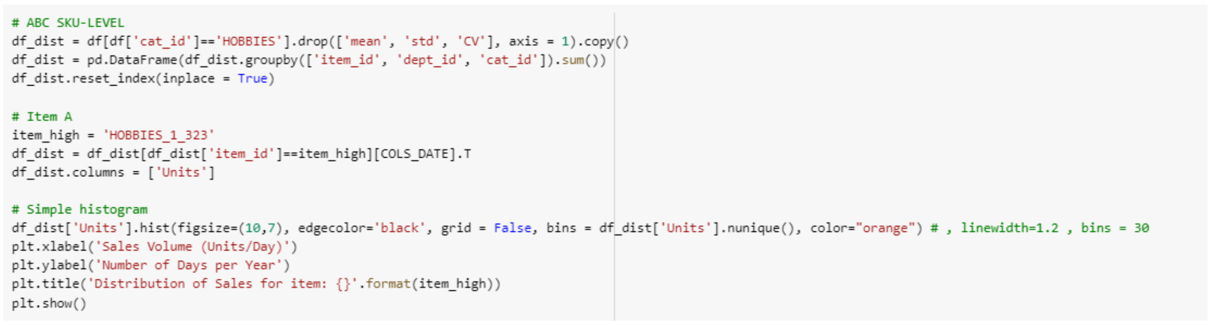
```python
# Bar Chart
ax = plt.gca()
colors = {'A':'red', 'B':'green', 'C':'aqua'}
# Remove Outliers
df_plot = df_abc[df_abc['CV']<4].copy()
df_plot.plot.scatter(figsize=(12, 8), x='TO%', y='CV', color=df_plot['ABC'].map(colors), ax =ax, grid = True)
# ABC
# A, B and C
ax.axvline(to_a , color="red", linestyle="-", linewidth = 1.0)
ax.axvline(to_b , color="red", linestyle="-", linewidth = 1.0)
# 20%, 50% of SKU Number
ax.axhline(1 , color="aqua", linestyle="--", linewidth = 1.0)
plt.xlabel('Percentage of Turnover (%)')
plt.xticks(rotation=90)
plt.ylabel('Coefficient of Variation')
plt.title('Distribution by Demand Variability')
plt.show()
```



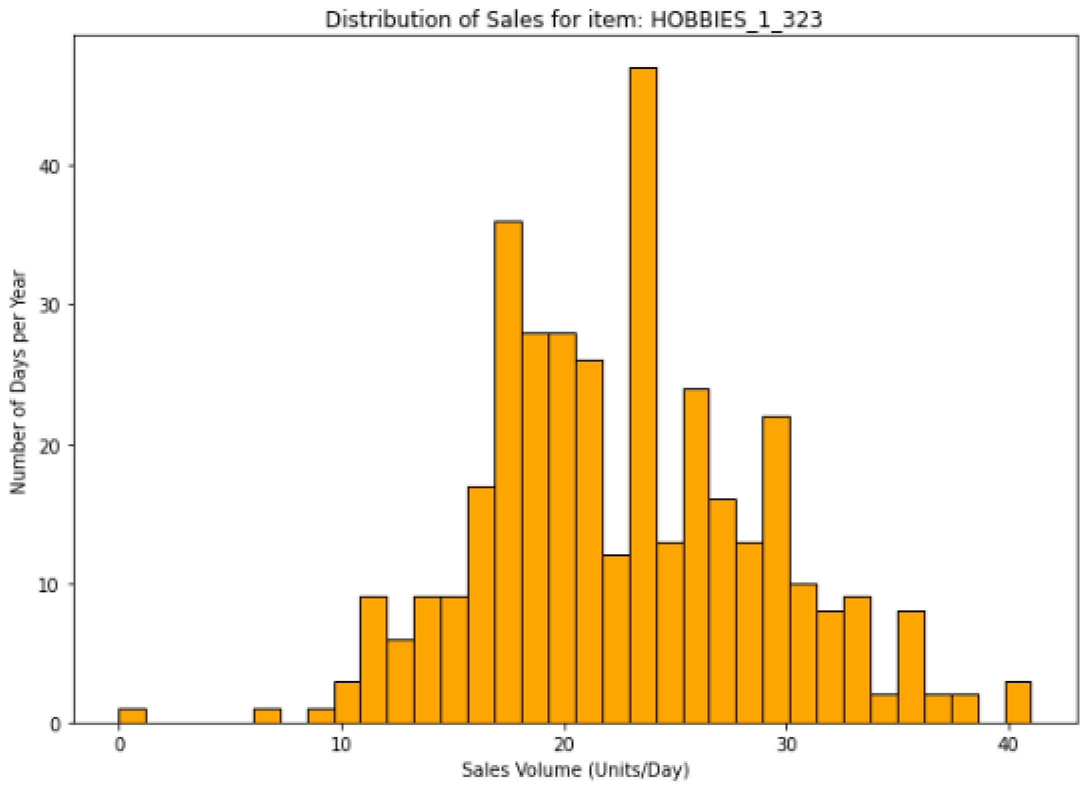
Distribution for Important Products with low variability | Example of item with low COV (<2)



Class A
Fortunately, most of the A SKU have a quite stable demand; we won't be challenged by the most important SKUs.

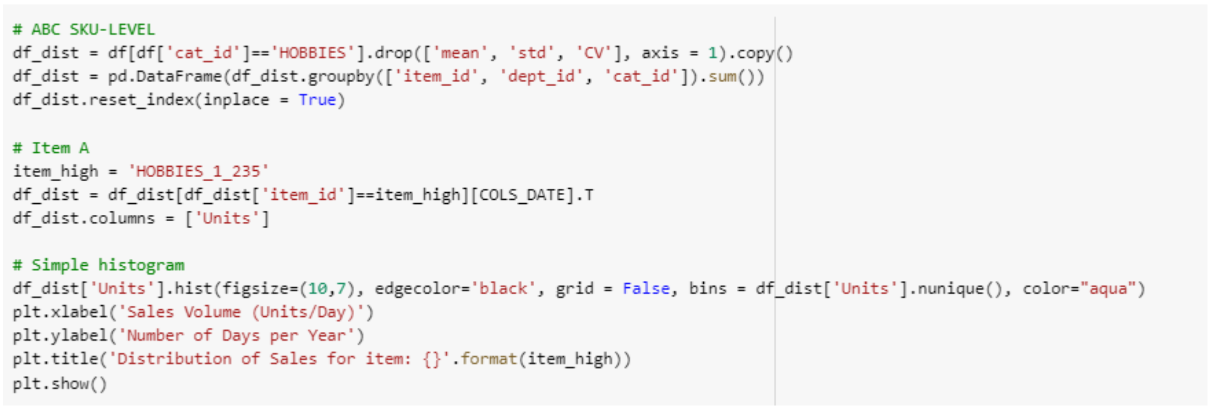


```python
# ABC SKU-LEVEL
df_dist = df[df['cat_id']=='HOBBIES'].drop(['mean', 'std', 'CV'], axis = 1).copy()
df_dist = pd.DataFrame(df_dist.groupby(['item_id', 'dept_id', 'cat_id']).sum())
df_dist.reset_index(inplace = True)

# Item A
item_high = 'HOBBIES_1_323'
df_dist = df_dist[df_dist['item_id']==item_high][COLS_DATE].T
df_dist.columns = ['Units']

# Simple histogram
df_dist['Units'].hist(figsize=(10,7), edgecolor='black', grid = False, bins = df_dist['Units'].nunique(), color="orange") # , linewidth=1.2 , bins = 30
plt.xlabel('Sales Volume (Units/Day)')
plt.ylabel('Number of Days per Year')
plt.title('Distribution of Sales for item: {}'.format(item_high))
plt.show()
```

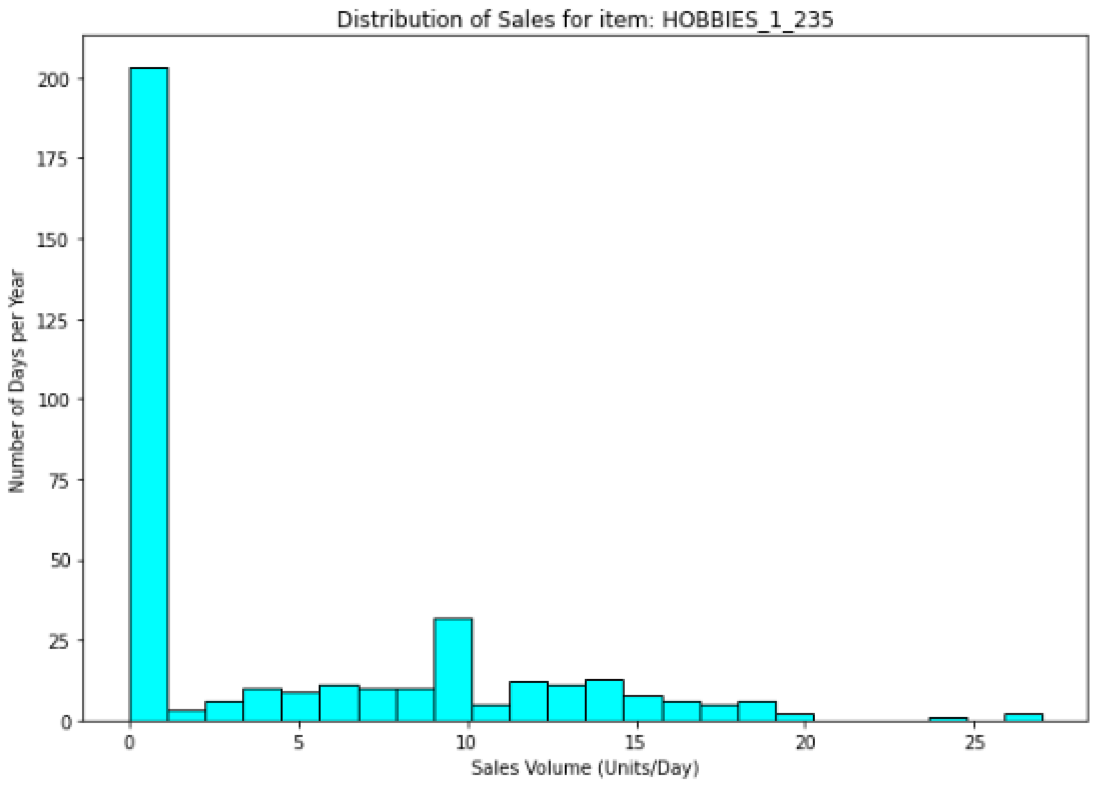


Distribution for Important Products with high variability | Example of product with High COV

```
Class B
The majority of SKUs are in the stable area; however we still
spend effort on ensuring optimal planning for the few
references that have a high CV.
```



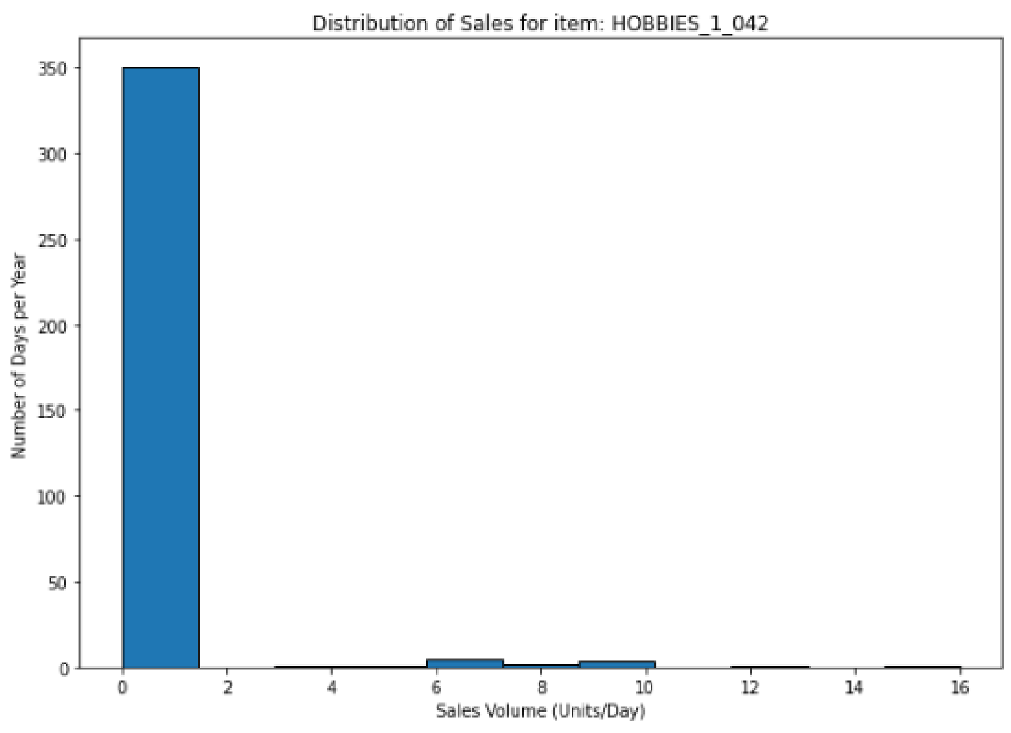
```
# ABC SKU-LEVEL
df_dist = df[df['cat_id']=='HOBBIES'].drop(['mean', 'std', 'CV'], axis = 1).copy()
df_dist = pd.DataFrame(df_dist.groupby(['item_id', 'dept_id', 'cat_id']).sum())
df_dist.reset_index(inplace = True)

# Item A
item_high = 'HOBBIES_1_235'
df_dist = df_dist[df_dist['item_id']==item_high][COLS_DATE].T
df_dist.columns = ['Units']

# Simple histogram
df_dist['Units'].hist(figsize=(10,7), edgecolor='black', grid = False, bins = df_dist['Units'].nunique(), color="aqua")
plt.xlabel('Sales Volume (Units/Day)')
plt.ylabel('Number of Days per Year')
plt.title('Distribution of Sales for item: {}'.format(item_high))
plt.show()
```



Distribution for Low rotation Products with very high variability | Example of item with very high COV (>2)

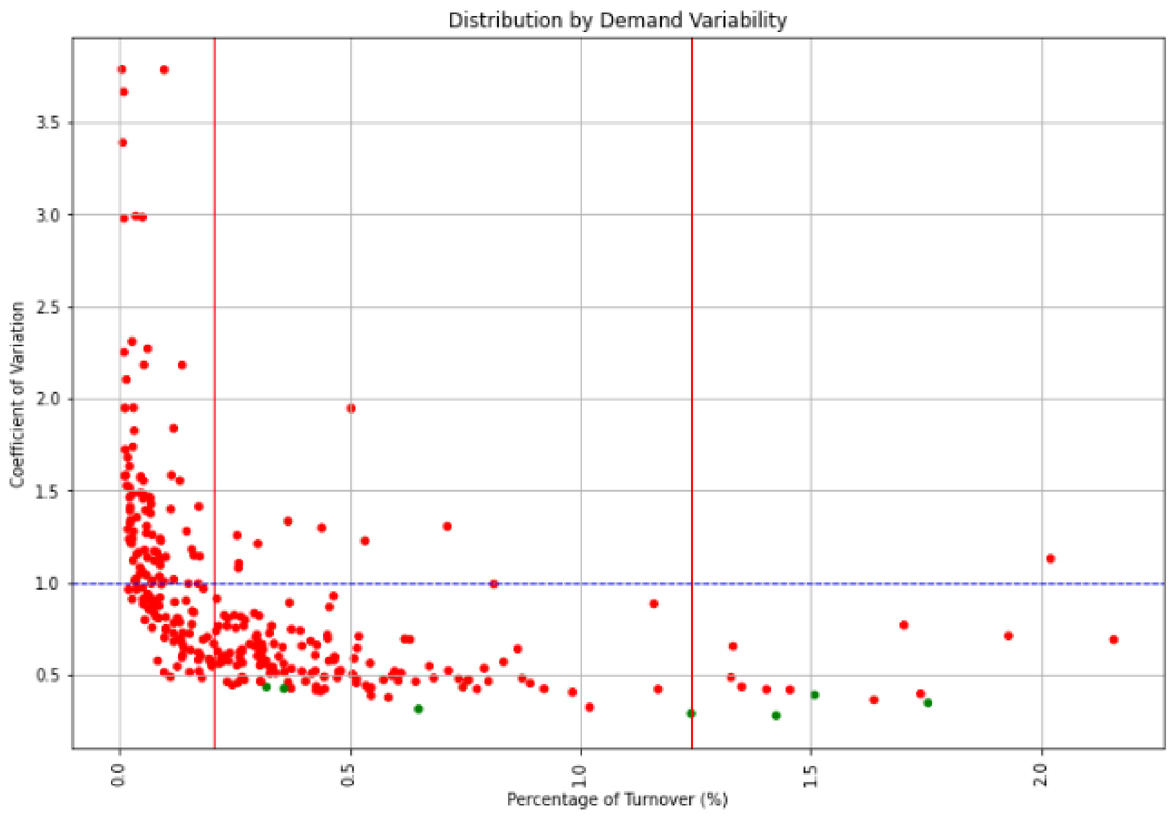
```
Class C
Most of the SKUs have a high value of CV;
For this kind of reference a cause analysis would provide
better results than a statistical approach for forecasting.
```

```
# ABC SKU-LEVEL
df_dist = df[df['cat_id']=='HOBBIES'].drop(['mean', 'std', 'CV'], axis = 1).copy()
df_dist = pd.DataFrame(df_dist.groupby(['item_id', 'dept_id', 'cat_id']).sum())
df_dist.reset_index(inplace = True)

# Item A
item_high = 'HOBBIES_1_042'
df_dist = df_dist[df_dist['item_id']==item_high][COLS_DATE].T
df_dist.columns = ['Units']

# Simple histogram
df_dist['Units'].hist(figsize=(10,7), edgecolor='black', grid = False, bins = df_dist['Units'].nunique()) # , linewidth=1.2 , bins = 30
plt.xlabel('Sales Volume (Units/Day)')
plt.ylabel('Number of Days per Year')
plt.title('Distribution of Sales for item: {}'.format(item_high))
plt.show()
```



Normality Test:

```python
# Bar Chart
ax = plt.gca()
colors = {False:'green', True:'red'}
# Remove Outliers
df_plot = df_abc[df_abc['CV']<4].copy()
df_plot.plot.scatter(figsize=(12, 8), x='TO%', y='CV', color=df_plot['Not_Normal'].map(colors), ax =ax, grid = True)
# ABC
# A, B and C
ax.axvline(to_a , color="red", linestyle="-", linewidth = 1.0)
ax.axvline(to_b , color="red", linestyle="-", linewidth = 1.0)
# 20%, 50% of SKU Number
ax.axhline(1 , color="blue", linestyle="--", linewidth = 1.0)
plt.xlabel('Percentage of Turnover (%)')
plt.xticks(rotation=90)
plt.ylabel('Coefficient of Variation')
plt.title('Distribution by Demand Variability')
plt.show()
```


Distribution by Demand Variability
Coefficient of Variation
Percentage of Turnover (%)