## Predict the sales demand for consumer goods

Vinayaka O

28 September 2020

**Contents**

**1. Introduction**

1.1 Problem Statement

1.2 Data

**2. Methodology**

2.1 Data Pre-Processing

2.2 Exploratory Data Analysis

2.3 Missing Value Analysis

2.4 Model Development

2.4.1 Model Selection: Random Forest Regression

**3. Conclusion**

3.1 Model Evaluation

4. **Python Code**

1. **Introduction**

**1.1 Problem Statement**

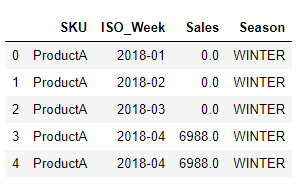
Predict the sales demand for consumer goods.

The objective of this exercise is to predict sales of 2018-42 to 2018-52 of each SKU. The data present in the mentioned week are only for computation of accuracy and bias only. So please use the sales data till 2018-41.

# 1.2 Data

The data set which is given by client has target variable which means it is comes under supervised machine learning algorithm and it is Regression problem so here we are going to use supervised Regression machine learning algorithm.

Sample dataset (top 5 Observation)



**Data Dictionary: -** The details of data attributes in the dataset are as follows let’s understand each attribute in detail

1. SKU
2. ISO\_Week
3. Sales
4. Season

The target column is the *Sales* column.

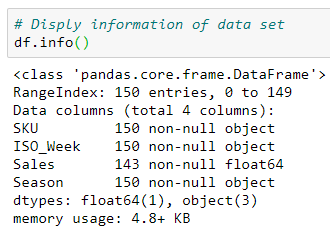
**2**. **Methodology**

**2.1 Data Pre-Processing:**

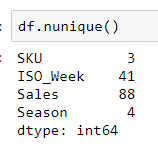
Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we preprocess our data before feeding it into our model. As we already know the quality of our inputs decide the quality of our output. So, once we have got our business hypothesis ready, it makes sense to spend lot of time and efforts here. Approximately, data exploration, cleaning and preparation can take up to 70% of our total project time. This process is often called as Exploratory Data Analysis

**2.2 Exploratory Data Analysis**

Exploratory Data Analysis (EDA) is an approach to analysing data sets to summarize their main characteristics. In the given data set there are 4 variables and data types of all variables are object, float64 or int64. There are 150 observations and 4 columns in our data set.

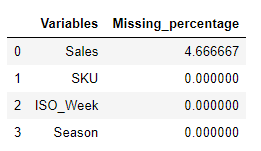


From EDA we have concluded the number of unique values in each variable



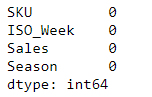
**2.3 Missing Value Analysis:**

Missing data or missing values occur when no data value is stored for the variable in an observation. In any real world dataset there are always few null values. It doesn’t really matter whether it is a regression, classification or any other kind of problem, no model can handle these NULL or NaN values on its own so we need to intervene. They are often encoded as NaNs, blanks or any other placeholders. If columns have more than 30% of data as missing value either we ignore the entire column or we ignore those observations.



In missing value analysis we got two idea one is drop that data and another one is imputation method. Here we got only 4.66% of missing value so we must impliment this. So I am going to use mean method to sortout this problem

After Missing value analysis:



## 2.4 Model Development

**2.4.1 Model Selection**

After Data pre-processing the next step is to develop a model using a train or historical data which can perform to predict accurate result on test data or new data. Here we have tried with Random Forest Regression Model

The data set which is given by client has target variable which means it is comes under supervised machine learning algorithm and it is Regression problem so here we are going to use supervised Regression machine learning algorithm.

**Random Forest Regression**

Random Forest is an ensemble technique that consists of many decision trees. The idea behind Random Forest is to build n number of trees to have more accuracy in dataset. It is called random forest as we are building n no. of trees randomly. In other words, to build the decision trees it selects randomly n no of variables and n no of observations. It means to build each decision tree on random forest we are not going to use the same data. The higher no of trees in the random forest will give higher number of accuracy, so in random forest we can go for multiple trees. It can handle large no of independent variables without variable deletion and it will give the estimates that what variables are important.

**3. Conclusion**

In methodology we have done data cleaning and then applied RF Regression machine learning algorithms on the data set to predict the sales

As per Requirement of output I have done the model and that output file saved and attached in this submission and it is in the excel format.

The attached Zip files contain Final Report, jupyter notebook (.ipynb), jupyter notebook (.html), output file (.xlsx).

.

**4. Python Code**

**Problem Statement: Predict the sales demand for consumer goods.**

**The objective of this exercise is to predict sales of 2018-42 to 2018-52 of each SKU. The data present in the mentioned week are only for computation of accuracy and bias only. So please use the sales data till 2018-41.**

The attached excel contains 4 columns:

a. SKU

b. ISO\_Week

c. Sales

d. Season

Below is the promotional details for the mentioned SKU (Stock keeping Unit)

In [1]:

*#importing all Required libraries*

**import** **os**

**import** **pandas** **as** **pd**

**import** **numpy** **as** **np**

**import** **matplotlib.pyplot** **as** **plt**

**from** **scipy.stats** **import** chi2\_contingency

**import** **seaborn** **as** **sns**

**from** **random** **import** randrange, uniform

**import** **warnings**

warnings.filterwarnings('ignore')

**Data Pre-Processing**

In [2]:

*# Assigning working directory*

os.chdir("C:/Users/Hp/Desktop/EY")

In [3]:

*# Assigning the data set*

df = pd.read\_excel("case\_study.xlsx")

In [4]:

*# Disply top 5 observation*

df.head()

Out[4]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 0 | ProductA | 2018-01 | 0.0 | WINTER |
| 1 | ProductA | 2018-02 | 0.0 | WINTER |
| 2 | ProductA | 2018-03 | 0.0 | WINTER |
| 3 | ProductA | 2018-04 | 6988.0 | WINTER |
| 4 | ProductA | 2018-04 | 6988.0 | WINTER |

**Exploratory Data Analysis (EDA)**

In [5]:

*# Disply information of data set*

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 4 columns):

SKU 150 non-null object

ISO\_Week 150 non-null object

Sales 143 non-null float64

Season 150 non-null object

dtypes: float64(1), object(3)

memory usage: 4.8+ KB

In [6]:

*# Shape of data set*

df.shape

Out[6]:

(150, 4)

In [7]:

df.describe()

Out[7]:

|  | **Sales** |
| --- | --- |
| count | 143.000000 |
| mean | 5279.776224 |
| std | 6184.437917 |
| min | -163.000000 |
| 25% | 290.500000 |
| 50% | 4341.000000 |
| 75% | 10191.500000 |
| max | 52524.000000 |

In [8]:

df.nunique()

Out[8]:

SKU 3

ISO\_Week 52

Sales 116

Season 4

dtype: int64

In [9]:

df = df[~df['SKU'].isin(['colgate', 'pepsodent', 'closeup'])]

df

Out[9]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 0 | ProductA | 2018-01 | 0.0 | WINTER |
| 1 | ProductA | 2018-02 | 0.0 | WINTER |
| 2 | ProductA | 2018-03 | 0.0 | WINTER |
| 3 | ProductA | 2018-04 | 6988.0 | WINTER |
| 4 | ProductA | 2018-04 | 6988.0 | WINTER |
| ... | ... | ... | ... | ... |
| 145 | ProductC | 2018-48 | 12927.0 | AUTUMN |
| 146 | ProductC | 2018-49 | 11595.0 | WINTER |
| 147 | ProductC | 2018-50 | 10061.0 | WINTER |
| 148 | ProductC | 2018-51 | 10859.0 | WINTER |
| 149 | ProductC | 2018-52 | 13093.0 | WINTER |

150 rows × 4 columns

**Missing Value Analysis**

In [10]:

*#Create dataframe with missing percentage*

missing\_val = pd.DataFrame(df.isnull().sum())

missing\_val

Out[10]:

|  | **0** |
| --- | --- |
| SKU | 0 |
| ISO\_Week | 0 |
| Sales | 7 |
| Season | 0 |

In [11]:

*# missing value percentege in asending order*

missing\_val = missing\_val.reset\_index()

missing\_val = missing\_val.rename(columns = {'index':'Variables', 0: 'Missing\_percentage'})

missing\_val['Missing\_percentage'] = (missing\_val['Missing\_percentage']/len(df))\*100

missing\_val = missing\_val.sort\_values('Missing\_percentage', ascending = **False**).reset\_index(drop = **True**)

missing\_val

Out[11]:

|  | **Variables** | **Missing\_percentage** |
| --- | --- | --- |
| 0 | Sales | 4.666667 |
| 1 | SKU | 0.000000 |
| 2 | ISO\_Week | 0.000000 |
| 3 | Season | 0.000000 |

In [12]:

*# filling missing values using mean method*

df['Sales'] = df['Sales'].fillna(df['Sales'].mean())

print (df.isnull().sum())

SKU 0

ISO\_Week 0

Sales 0

Season 0

dtype: int64

In [13]:

df = df.fillna(df.mean())

df1 = df.copy()

df

Out[13]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 0 | ProductA | 2018-01 | 0.0 | WINTER |
| 1 | ProductA | 2018-02 | 0.0 | WINTER |
| 2 | ProductA | 2018-03 | 0.0 | WINTER |
| 3 | ProductA | 2018-04 | 6988.0 | WINTER |
| 4 | ProductA | 2018-04 | 6988.0 | WINTER |
| ... | ... | ... | ... | ... |
| 145 | ProductC | 2018-48 | 12927.0 | AUTUMN |
| 146 | ProductC | 2018-49 | 11595.0 | WINTER |
| 147 | ProductC | 2018-50 | 10061.0 | WINTER |
| 148 | ProductC | 2018-51 | 10859.0 | WINTER |
| 149 | ProductC | 2018-52 | 13093.0 | WINTER |

150 rows × 4 columns

**Outlier treatment**

In [15]:

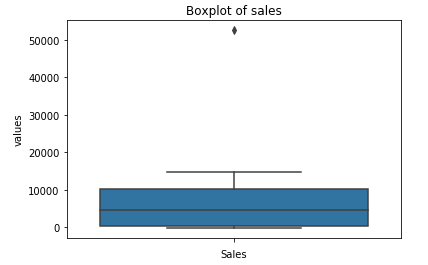
sns.boxplot(y=df['Sales'])

plt.xlabel("Sales")

plt.ylabel("values")

plt.title("Boxplot of sales")

plt.show()



In [16]:

*# Quartiles and IQR*

q25,q75 = np.percentile(df['Sales'],[25,75])

IQR = q75-q25

*# Lower and upper limits*

LL = q25 - (1.5 \* IQR)

UL = q75 + (1.5 \* IQR)

*# Capping with ul for maxmimum values*

*# For inliers*

df.loc[df['Sales'] < LL ,'Sales'] = LL

*# For ioutliers*

df.loc[df['Sales'] > UL ,'Sales'] = UL

Boxplot fot Sales after outlier treatment

In [17]:

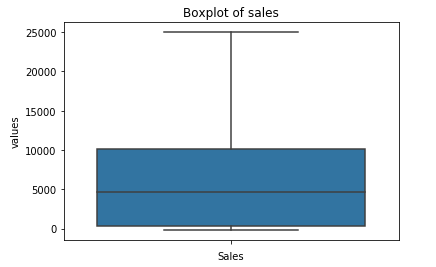
sns.boxplot(y=df['Sales'])

plt.xlabel("Sales")

plt.ylabel("values")

plt.title("Boxplot of sales")

plt.show()



**Visualisation of Sales Variable**

In [18]:

*# histogram plot of Sales Variable to check the distribution*

plt.hist(df['Sales'])

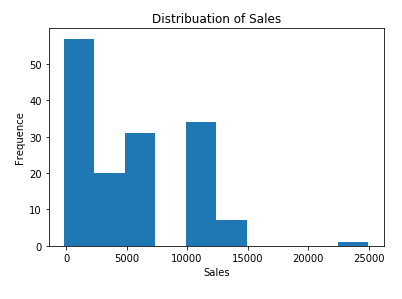
plt.xlabel("Sales")

plt.ylabel("Frequence")

plt.title('Distribuation of Sales')

Out[18]:

Text(0.5, 1.0, 'Distribuation of Sales')



**Feature Engineering**

In [19]:

df = df[~df['ISO\_Week'].isin(['2018-42', '2018-43', '2018-44', '2018-45', '2018-46', '2018-47', '2018-48', '2018-49', '2018-50', '2018-51', '2018-52'])]

In [20]:

df

Out[20]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 0 | ProductA | 2018-01 | 0.0 | WINTER |
| 1 | ProductA | 2018-02 | 0.0 | WINTER |
| 2 | ProductA | 2018-03 | 0.0 | WINTER |
| 3 | ProductA | 2018-04 | 6988.0 | WINTER |
| 4 | ProductA | 2018-04 | 6988.0 | WINTER |
| ... | ... | ... | ... | ... |
| 134 | ProductC | 2018-37 | 11856.0 | AUTUMN |
| 135 | ProductC | 2018-38 | 4185.0 | AUTUMN |
| 136 | ProductC | 2018-39 | 4357.0 | AUTUMN |
| 137 | ProductC | 2018-40 | 3730.0 | AUTUMN |
| 138 | ProductC | 2018-41 | 3158.0 | AUTUMN |

117 rows × 4 columns

In [21]:

df1 = df1[df1['ISO\_Week'].isin(['2018-42', '2018-43', '2018-44', '2018-45', '2018-46', '2018-47', '2018-48', '2018-49', '2018-50', '2018-51', '2018-52'])]

In [22]:

df1

Out[22]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 43 | ProductA | 2018-42 | 4740.0 | AUTUMN |
| 44 | ProductA | 2018-43 | 4341.0 | AUTUMN |
| 45 | ProductA | 2018-44 | 5363.0 | AUTUMN |
| 46 | ProductA | 2018-45 | 12120.0 | AUTUMN |
| 47 | ProductA | 2018-46 | 3314.0 | AUTUMN |
| 48 | ProductA | 2018-47 | 4673.0 | AUTUMN |
| 49 | ProductA | 2018-48 | 12376.0 | AUTUMN |
| 50 | ProductA | 2018-49 | 14853.0 | WINTER |
| 51 | ProductA | 2018-50 | 12447.0 | WINTER |
| 52 | ProductA | 2018-51 | 12247.0 | WINTER |
| 53 | ProductA | 2018-52 | 13679.0 | WINTER |
| 96 | ProductB | 2018-42 | -23.0 | AUTUMN |
| 97 | ProductB | 2018-43 | 0.0 | AUTUMN |
| 98 | ProductB | 2018-44 | 0.0 | AUTUMN |
| 99 | ProductB | 2018-45 | 0.0 | AUTUMN |
| 100 | ProductB | 2018-46 | 376.0 | AUTUMN |
| 101 | ProductB | 2018-47 | 483.0 | AUTUMN |
| 102 | ProductB | 2018-48 | 301.0 | AUTUMN |
| 103 | ProductB | 2018-49 | 489.0 | WINTER |
| 104 | ProductB | 2018-50 | 487.0 | WINTER |
| 105 | ProductB | 2018-51 | 464.0 | WINTER |
| 106 | ProductB | 2018-52 | 488.0 | WINTER |
| 139 | ProductC | 2018-42 | 3768.0 | AUTUMN |
| 140 | ProductC | 2018-43 | 3929.0 | AUTUMN |
| 141 | ProductC | 2018-44 | 4588.0 | AUTUMN |
| 142 | ProductC | 2018-45 | 3212.0 | AUTUMN |
| 143 | ProductC | 2018-46 | 14521.0 | AUTUMN |
| 144 | ProductC | 2018-47 | 14510.0 | AUTUMN |
| 145 | ProductC | 2018-48 | 12927.0 | AUTUMN |
| 146 | ProductC | 2018-49 | 11595.0 | WINTER |
| 147 | ProductC | 2018-50 | 10061.0 | WINTER |
| 148 | ProductC | 2018-51 | 10859.0 | WINTER |
| 149 | ProductC | 2018-52 | 13093.0 | WINTER |

In [23]:

Data = df1.copy()

**Model Development**

In [24]:

**from** **sklearn.preprocessing** **import** LabelEncoder

le = LabelEncoder()

objList = df.select\_dtypes(include = "object").columns

**for** feat **in** objList:

df[feat] = le.fit\_transform(df[feat].astype(str))

In [25]:

df.head()

Out[25]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 0 | 0 | 0 | 0.0 | 3 |
| 1 | 0 | 1 | 0.0 | 3 |
| 2 | 0 | 2 | 0.0 | 3 |
| 3 | 0 | 3 | 6988.0 | 3 |
| 4 | 0 | 3 | 6988.0 | 3 |

In [26]:

**from** **sklearn.preprocessing** **import** LabelEncoder

le = LabelEncoder()

objList = df1.select\_dtypes(include = "object").columns

**for** feat **in** objList:

df1[feat] = le.fit\_transform(df1[feat].astype(str))

In [27]:

df1.head()

Out[27]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** |
| --- | --- | --- | --- | --- |
| 43 | 0 | 0 | 4740.0 | 0 |
| 44 | 0 | 1 | 4341.0 | 0 |
| 45 | 0 | 2 | 5363.0 | 0 |
| 46 | 0 | 3 | 12120.0 | 0 |
| 47 | 0 | 4 | 3314.0 | 0 |

In [28]:

X = df.iloc[:, [0,1,3]].values

y = df.iloc[:, 2].values

In [29]:

X1 = df1.iloc[:, [0,1,3]].values

y1 = df1.iloc[:, 2].values

In [30]:

print(X)

[[ 0 0 3]

[ 0 1 3]

[ 0 2 3]

[ 0 3 3]

[ 0 3 3]

[ 0 4 3]

[ 0 5 3]

[ 0 6 3]

[ 0 7 3]

[ 0 8 1]

[ 0 9 1]

[ 0 10 1]

[ 0 11 1]

[ 0 12 1]

[ 0 13 1]

[ 0 14 1]

[ 0 14 1]

[ 0 15 1]

[ 0 16 1]

[ 0 17 1]

[ 0 18 1]

[ 0 19 1]

[ 0 20 1]

[ 0 21 1]

[ 0 22 2]

[ 0 23 2]

[ 0 24 2]

[ 0 25 2]

[ 0 26 2]

[ 0 27 2]

[ 0 28 2]

[ 0 29 2]

[ 0 30 2]

[ 0 31 2]

[ 0 32 2]

[ 0 33 2]

[ 0 34 2]

[ 0 35 0]

[ 0 36 0]

[ 0 37 0]

[ 0 38 0]

[ 0 39 0]

[ 0 40 0]

[ 1 0 3]

[ 1 1 3]

[ 1 2 3]

[ 1 3 3]

[ 1 4 3]

[ 1 5 3]

[ 1 6 3]

[ 1 7 3]

[ 1 7 3]

[ 1 8 1]

[ 1 9 1]

[ 1 10 1]

[ 1 11 1]

[ 1 12 1]

[ 1 13 1]

[ 1 14 1]

[ 1 15 1]

[ 1 16 1]

[ 1 17 1]

[ 1 18 1]

[ 1 19 1]

[ 1 20 1]

[ 1 21 1]

[ 1 22 2]

[ 1 23 2]

[ 1 24 2]

[ 1 25 2]

[ 1 26 2]

[ 1 27 2]

[ 1 28 2]

[ 1 29 2]

[ 1 30 2]

[ 1 31 2]

[ 1 32 2]

[ 1 33 2]

[ 1 34 2]

[ 1 35 0]

[ 1 36 0]

[ 1 37 0]

[ 1 38 0]

[ 1 39 0]

[ 1 40 0]

[ 2 10 1]

[ 2 11 1]

[ 2 12 1]

[ 2 13 1]

[ 2 14 1]

[ 2 14 1]

[ 2 15 1]

[ 2 16 1]

[ 2 17 1]

[ 2 18 1]

[ 2 19 1]

[ 2 20 1]

[ 2 21 1]

[ 2 22 2]

[ 2 23 2]

[ 2 24 2]

[ 2 25 2]

[ 2 26 2]

[ 2 27 2]

[ 2 28 2]

[ 2 29 2]

[ 2 30 2]

[ 2 31 2]

[ 2 32 2]

[ 2 33 2]

[ 2 34 2]

[ 2 35 0]

[ 2 36 0]

[ 2 37 0]

[ 2 38 0]

[ 2 39 0]

[ 2 40 0]]

In [31]:

print(y)

[ 0.00000000e+00 0.00000000e+00 0.00000000e+00 6.98800000e+03

6.98800000e+03 6.74300000e+03 4.11200000e+03 5.73200000e+03

5.27977622e+03 5.55900000e+03 5.38100000e+03 1.17090000e+04

1.04100000e+04 1.02110000e+04 1.03780000e+04 1.00120000e+04

1.00120000e+04 1.01780000e+04 1.13160000e+04 5.27977622e+03

1.13840000e+04 1.04700000e+04 5.47100000e+03 4.87400000e+03

6.74600000e+03 1.07770000e+04 3.73000000e+03 1.11590000e+04

6.56800000e+03 4.91500000e+03 5.27977622e+03 5.27977622e+03

4.57900000e+03 6.78000000e+03 3.64800000e+03 3.03600000e+03

3.42500000e+03 4.97900000e+03 3.11500000e+03 3.72200000e+03

6.54500000e+03 6.69900000e+03 6.41200000e+03 5.27977622e+03

5.27977622e+03 5.27977622e+03 -2.30000000e+01 4.46000000e+02

3.61000000e+02 2.29000000e+02 2.19000000e+02 2.19000000e+02

4.95000000e+02 2.78000000e+02 3.16000000e+02 -5.00000000e+01

0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00

0.00000000e+00 0.00000000e+00 0.00000000e+00 4.42000000e+02

4.59000000e+02 3.92000000e+02 3.81000000e+02 2.88000000e+02

2.40000000e+02 3.61000000e+02 0.00000000e+00 0.00000000e+00

-4.50000000e+01 0.00000000e+00 0.00000000e+00 0.00000000e+00

2.41000000e+02 3.28000000e+02 2.33000000e+02 2.93000000e+02

3.87000000e+02 3.85000000e+02 4.42000000e+02 0.00000000e+00

0.00000000e+00 5.49500000e+03 6.33000000e+03 6.14400000e+03

6.38300000e+03 5.53300000e+03 5.53300000e+03 6.61900000e+03

-1.11000000e+02 -1.49000000e+02 -1.63000000e+02 -1.19000000e+02

1.17930000e+04 1.15830000e+04 1.18280000e+04 1.16720000e+04

1.03170000e+04 1.04680000e+04 1.01860000e+04 1.08920000e+04

1.09610000e+04 2.49816250e+04 1.05320000e+04 1.01680000e+04

1.15330000e+04 1.16630000e+04 1.01970000e+04 1.15490000e+04

1.18560000e+04 4.18500000e+03 4.35700000e+03 3.73000000e+03

3.15800000e+03]

**Random Forest Regression**

In [32]:

*# Applying RF Regression Model*

**from** **sklearn.ensemble** **import** RandomForestRegressor

regressor = RandomForestRegressor(n\_estimators = 10, random\_state = 0)

regressor.fit(X, y)

Out[32]:

RandomForestRegressor(n\_estimators=10, random\_state=0)

In [33]:

pred = regressor.predict(X1)

In [34]:

pred

Out[34]:

array([1094. , 1094. , 1094. , 5286.8 ,

6298.8 , 4953.4 , 5662.3 , 5418.88811189,

5467.55524476, 5406.47762238, 7902.07762238, 5279.77622378,

5279.77622378, 5279.77622378, 3210.46573427, 306.6 ,

405. , 334.6 , 345. , 400.2 ,

325.2 , 282.1 , 5745.5 , 5745.5 ,

5745.5 , 5745.5 , 5745.5 , 5745.5 ,

5745.5 , 5745.5 , 5745.5 , 5745.5 ,

5745.5 ])

In [35]:

df2 = pd.DataFrame(pred)

df2.columns = ['Forecast']

df2

Out[35]:

|  | **Forecast** |
| --- | --- |
| 0 | 1094.000000 |
| 1 | 1094.000000 |
| 2 | 1094.000000 |
| 3 | 5286.800000 |
| 4 | 6298.800000 |
| 5 | 4953.400000 |
| 6 | 5662.300000 |
| 7 | 5418.888112 |
| 8 | 5467.555245 |
| 9 | 5406.477622 |
| 10 | 7902.077622 |
| 11 | 5279.776224 |
| 12 | 5279.776224 |
| 13 | 5279.776224 |
| 14 | 3210.465734 |
| 15 | 306.600000 |
| 16 | 405.000000 |
| 17 | 334.600000 |
| 18 | 345.000000 |
| 19 | 400.200000 |
| 20 | 325.200000 |
| 21 | 282.100000 |
| 22 | 5745.500000 |
| 23 | 5745.500000 |
| 24 | 5745.500000 |
| 25 | 5745.500000 |
| 26 | 5745.500000 |
| 27 | 5745.500000 |
| 28 | 5745.500000 |
| 29 | 5745.500000 |
| 30 | 5745.500000 |
| 31 | 5745.500000 |
| 32 | 5745.500000 |

In [36]:

Data = Data.reset\_index()

Data = Data.drop(['index'],axis=1)

In [37]:

Data['Forecast'] = df2['Forecast']

Data

Out[37]:

|  | **SKU** | **ISO\_Week** | **Sales** | **Season** | **Forecast** |
| --- | --- | --- | --- | --- | --- |
| 0 | ProductA | 2018-42 | 4740.0 | AUTUMN | 1094.000000 |
| 1 | ProductA | 2018-43 | 4341.0 | AUTUMN | 1094.000000 |
| 2 | ProductA | 2018-44 | 5363.0 | AUTUMN | 1094.000000 |
| 3 | ProductA | 2018-45 | 12120.0 | AUTUMN | 5286.800000 |
| 4 | ProductA | 2018-46 | 3314.0 | AUTUMN | 6298.800000 |
| 5 | ProductA | 2018-47 | 4673.0 | AUTUMN | 4953.400000 |
| 6 | ProductA | 2018-48 | 12376.0 | AUTUMN | 5662.300000 |
| 7 | ProductA | 2018-49 | 14853.0 | WINTER | 5418.888112 |
| 8 | ProductA | 2018-50 | 12447.0 | WINTER | 5467.555245 |
| 9 | ProductA | 2018-51 | 12247.0 | WINTER | 5406.477622 |
| 10 | ProductA | 2018-52 | 13679.0 | WINTER | 7902.077622 |
| 11 | ProductB | 2018-42 | -23.0 | AUTUMN | 5279.776224 |
| 12 | ProductB | 2018-43 | 0.0 | AUTUMN | 5279.776224 |
| 13 | ProductB | 2018-44 | 0.0 | AUTUMN | 5279.776224 |
| 14 | ProductB | 2018-45 | 0.0 | AUTUMN | 3210.465734 |
| 15 | ProductB | 2018-46 | 376.0 | AUTUMN | 306.600000 |
| 16 | ProductB | 2018-47 | 483.0 | AUTUMN | 405.000000 |
| 17 | ProductB | 2018-48 | 301.0 | AUTUMN | 334.600000 |
| 18 | ProductB | 2018-49 | 489.0 | WINTER | 345.000000 |
| 19 | ProductB | 2018-50 | 487.0 | WINTER | 400.200000 |
| 20 | ProductB | 2018-51 | 464.0 | WINTER | 325.200000 |
| 21 | ProductB | 2018-52 | 488.0 | WINTER | 282.100000 |
| 22 | ProductC | 2018-42 | 3768.0 | AUTUMN | 5745.500000 |
| 23 | ProductC | 2018-43 | 3929.0 | AUTUMN | 5745.500000 |
| 24 | ProductC | 2018-44 | 4588.0 | AUTUMN | 5745.500000 |
| 25 | ProductC | 2018-45 | 3212.0 | AUTUMN | 5745.500000 |
| 26 | ProductC | 2018-46 | 14521.0 | AUTUMN | 5745.500000 |
| 27 | ProductC | 2018-47 | 14510.0 | AUTUMN | 5745.500000 |
| 28 | ProductC | 2018-48 | 12927.0 | AUTUMN | 5745.500000 |
| 29 | ProductC | 2018-49 | 11595.0 | WINTER | 5745.500000 |
| 30 | ProductC | 2018-50 | 10061.0 | WINTER | 5745.500000 |
| 31 | ProductC | 2018-51 | 10859.0 | WINTER | 5745.500000 |
| 32 | ProductC | 2018-52 | 13093.0 | WINTER | 5745.500000 |

In [38]:

**del** Data['Sales']

In [39]:

**del** Data['Season']

In [40]:

Data

Out[40]:

|  | **SKU** | **ISO\_Week** | **Forecast** |
| --- | --- | --- | --- |
| 0 | ProductA | 2018-42 | 1094.000000 |
| 1 | ProductA | 2018-43 | 1094.000000 |
| 2 | ProductA | 2018-44 | 1094.000000 |
| 3 | ProductA | 2018-45 | 5286.800000 |
| 4 | ProductA | 2018-46 | 6298.800000 |
| 5 | ProductA | 2018-47 | 4953.400000 |
| 6 | ProductA | 2018-48 | 5662.300000 |
| 7 | ProductA | 2018-49 | 5418.888112 |
| 8 | ProductA | 2018-50 | 5467.555245 |
| 9 | ProductA | 2018-51 | 5406.477622 |
| 10 | ProductA | 2018-52 | 7902.077622 |
| 11 | ProductB | 2018-42 | 5279.776224 |
| 12 | ProductB | 2018-43 | 5279.776224 |
| 13 | ProductB | 2018-44 | 5279.776224 |
| 14 | ProductB | 2018-45 | 3210.465734 |
| 15 | ProductB | 2018-46 | 306.600000 |
| 16 | ProductB | 2018-47 | 405.000000 |
| 17 | ProductB | 2018-48 | 334.600000 |
| 18 | ProductB | 2018-49 | 345.000000 |
| 19 | ProductB | 2018-50 | 400.200000 |
| 20 | ProductB | 2018-51 | 325.200000 |
| 21 | ProductB | 2018-52 | 282.100000 |
| 22 | ProductC | 2018-42 | 5745.500000 |
| 23 | ProductC | 2018-43 | 5745.500000 |
| 24 | ProductC | 2018-44 | 5745.500000 |
| 25 | ProductC | 2018-45 | 5745.500000 |
| 26 | ProductC | 2018-46 | 5745.500000 |
| 27 | ProductC | 2018-47 | 5745.500000 |
| 28 | ProductC | 2018-48 | 5745.500000 |
| 29 | ProductC | 2018-49 | 5745.500000 |
| 30 | ProductC | 2018-50 | 5745.500000 |
| 31 | ProductC | 2018-51 | 5745.500000 |
| 32 | ProductC | 2018-52 | 5745.500000 |

In [41]:

Data.to\_excel('forecast\_data.xlsx', index = **False**)

Final Output which is saved as " forecast\_data.xlsx " in excel formate

**End Model**

**Thank You**