Product Demand Forecasting using Machine Learning

We have intiated ,Demand forecasting is the process of making estimations about future customer demand over a defined period, using historical data and other information.

Usually organisations follow traditional forecasting techniques/algorithms such as Auto Arima, Auto Arima, Sarima, Simple moving average and many more

Goal :

Due to the recent boost in AI world, companies have started researching the possibility of using machine learning in place of traditional approach

Tuning traditional algorithms takes a significant amount of effort and domain expertise as well.

In this repo, we are trying to figure out a way to predict the same using machine learning algorithms.

Data :

The dataset comprised of units sold on a daily basis along with details regarding the sales, eg. SKU(product id), Store, price etc.

record_ID, week, store_id, sku_id, total_price, base_price, is_featured_sku, is_display_sku, units_sold

Workflow :

- Handling missing values
- Feature selection based on my previous experience in Supply chain domain
- Converting dataset into time series format to apply supervised learning approach.
- Regression Modeling
- Random Forest
- XGBoost
- SVM (future scope)
- Hyperparameter Tuning

Sample dataset:

- Data Pre-processing: Date Conversion: Convert the date column to a datetime object if it's not already in that format.
- Sorting: Sort the Data Frame by date if it's not already in chronological order.
- Feature Engineering: Create additional features if needed, such as moving averages, daily returns, or technical indicators.
  Normalization/Scaling: If you plan to use machine learning models, you might want to normalize or scale the data. Common techniques include Min-Max scaling or standardization.

	record_ID	week	store_id	sku_id	total_price	base_price	is_featured_sku	is_display_sku	units_sold
0	1.0	17/01/11	8091.0	216418.0	99.0375	111.8625	0.0	0.0	20.0
1	2.0	17/01/11	8091.0	216419.0	99.0375	99.0375	0.0	0.0	28.0
2	3.0	17/01/11	8091.0	216425.0	133.9500	133.9500	0.0	0.0	19.0
3	4.0	17/01/11	8091.0	216233.0	133.9500	133.9500	0.0	0.0	44.0
4	5.0	17/01/11	8091.0	217390.0	141.0750	141.0750	0.0	0.0	52.0
...	...	...	...	...	...	...	...	...	...
150145	212638.0	09/07/13	9984.0	223245.0	235.8375	235.8375	0.0	0.0	38.0
150146	212639.0	09/07/13	9984.0	223153.0	235.8375	235.8375	0.0	0.0	30.0
150147	212642.0	09/07/13	9984.0	245338.0	357.6750	483.7875	1.0	1.0	31.0
150148	212643.0	09/07/13	9984.0	547934.0	141.7875	191.6625	0.0	1.0	12.0
150149	212644.0	09/07/13	9984.0	679023.0	234.4125	234.4125	0.0	0.0	15.0

150150 rows × 9 columns

About Data:

Easypreprocessing is a library used for common ML related preprocessing activities.

In [2]:

prep **=** EasyPreProcessing('train.csv') //data

data.csv

Initialization Parameters

1. output - Set output variable/dependent variable
2. dates.features - Set datetime field names (optional)

For example:

1. output = 'column_name'
2. dates.features = ['date_field_1','date_field_2']

In [3]:

Prep.info //data

General Preprocessing

missing_values	Display missing value report
remove_blank()	Remove empty/blank columns
correction()	Display correction heatmap
standardize()	Standardize entire dataset except dependent variable
encode_output()	Encode dependent feature/output variable
over_sample()	Oversample dataset. Parameters {'smote': SMOTE, 'ros': RandomOverSample}
clustering.apply()	Cluster dataset using elbow plot

General Template

```
from easypreprocessing import EasyPreProcessing
prep = EasyPreProcessing('filename_here.csv')
prep.df
prep.output = 'output_variable_here'
prep.remove_blank()
prep.missing_values
prep.categorical.impute()
prep.numerical.impute()
prep.categorical.encode()
prep.correction()
prep.standardize()
X_train, X_test, y_train, y_test = prep.split()
```

Categorical Preprocessing

categorical.fields	Display all categorical field names
categorical.unique	Display unique/distinct categorical values
categorical.impute()	Handle categorical missing values. Parameters {'mean', 'medium', 'mode'}
categorical.encode()	Encode categorical features. Parameters {'le': LabelEncoding, 'ohe': OneHotEncoding}

Numerical Preprocessing

numerical.fields	Display all numerical field names
numerical.impute()	Handle numerical missing values. Parameters {'mean', 'medium', 'mode'}

Summing units_sold group by key.

```
prep.dataset = prep.df.groupby('key').sum()
```

```
prep.df
```

key	units_sold
01/01/13_8023.0	2025.0
01/01/13_8058.0	682.0
01/01/13_8063.0	535.0
01/01/13_8091.0	210.0
01/01/13_8094.0	782.0
...	...
31/10/11_9890.0	531.0
31/10/11_9909.0	551.0
31/10/11_9954.0	431.0
31/10/11_9961.0	820.0
31/10/11_9984.0	506.0

9880 rows × 1 columns

Sample output: