E-Commerce Shipping Data set(DataPre-Processing)

Getting Started :

- Handle missing values
  - Handle duplicated data
  - Handle outliers
  - Feature transformation
  - Feature encoding
  - Handle class imbalance

A. How to Handle missing values

Missing value in a dataset is a very common phenomenon in the reality, yet a big problem in real-life scenarios. MissingData can also refer to as NA(Not Available) values in pandas

```
The df.info() method prints information about the DataFrame. The
information contains the number of columns, column labels, column data
types, memory usage, range index,and the number of cells in each column
(non-null values).
```

Count NaN values in Pandas DataFrame

```
df. isna(). sum() returns the number of missing values in each column
```

Grouping coloumn by type

The group by is one of the most frequently used Pandas functions in data analysis. It is used for grouping the data points based on the distinct values in the given column or columns

```
cat = ['Warehouse_block', 'Mode_of_Shipment', 'Product_importance',
'Gender']
num = ['Customer_care_calls', 'Customer_rating', 'Cost_of_the_Product',
'Prior_purchases', 'Discount_offered','Weight_in_gms']
num_cat = ['Reached.on.Time_Y.N']
```

Statistic summary from numeric coloumn

```
df[num].describe()
```

Summary from Categorical coloumn

Provides descriptive statistics that summarizes the central tendency, dispersion, and shape.

```
df[cat].describe()
```

- **From handle missing value in our dataset there is nothing to handle because our data is intact.**

B. How to handle duplicate data

Duplicate data is any record that inadvertently shares data with another record in a Database

```
df.duplicated().sum()
```

C. How to Handle Outliers

```
from scipy import stats
print(f'Jumlah baris sebelum memfilter outlier: {len(df)}')
filtered_entries = np.array([True] * len(df))
for col in ['Customer_care_calls', 'Prior_purchases']:
zscore = abs(stats.zscore(df[col]))
filtered_entries = (zscore < 3) & filtered_entries
df1 = df[filtered_entries]
print(f'Jumlah baris setelah memfilter outlier: {len(df1)}')
```

```
Result : Jumlah baris sebelum memfilter outlier: 10999
Jumlah baris setelah memfilter outlier: 10821
```

- **To handle outliers we need to remove the extreme outliers from the prior purchases column, before filltering the initial number of rows is 10999 and then after filletring the initial number become 10821 rows**

D. Feature transformation

Feature transformation is a mathematical transformation in which we apply a mathematical formula to a particular column (feature) and transform the values, which are useful for our further analysis.

Standard Scaler transformation Python sklearn library offers us with StandardScaler() function to standardize the data values into a standard format

```
from sklearn.preprocessing import MinMaxScaler, StandardScaler
df1['std_Prior_purchases'] =
StandardScaler().fit_transform(df1['Prior_purchases'].values.reshape(len(df1), 1))
```

Min MaxScaler transformation

```
df1['norm_Customer_care_calls'] =
MinMaxScaler().fit_transform(df1['Customer_care_calls'].values.reshape(len(df1),1))
df1['norm_Cost_of_the_Product'] =
```

```
MinMaxScaler().fit_transform(df1['Cost_of_the_Product'].values.reshape(len(df1),1))
df1['norm_Weight_in_gms'] =
MinMaxScaler().fit_transform(df1['Weight_in_gms'].values.reshape(len(df1), 1))
```

```
std_num = ['std_Prior_purchases', 'norm_Customer_care_calls',
'norm_Cost_of_the_Product', 'norm_Weight_in_gms']
```

```
plt.figure(figsize = (20, 10))
for i in range(0, len(std_num)):
  plt.subplot(3, len(std_num), i+1)
  sns.kdeplot(df1[std_num[i]])
  plt.tight_layout()
```

Method 2 using function normalizer

```
num = ['Customer_care_calls', 'Cost_of_the_Product', 'Prior_purchases',
'Weight_in_gms']
plt.figure(figsize=(12,8))

for i, column in enumerate (df[num].columns, 1):
    plt.subplot(4,4,i)
    sns.kdeplot(data=df[num], x=df[column])
    plt.tight_layout()
```

```
from sklearn.preprocessing import Normalizer
num = ['Customer_care_calls', 'Cost_of_the_Product', 'Prior_purchases',
'Weight_in_gms']
features = df1[num]
scaler = Normalizer(norm = 'l2')
# norm = 'l2' is default
df1[num] = scaler.fit_transform(features.values)
plt.figure(figsize=(12,8))
for i, column in enumerate (df1[num].columns, 1):
    plt.subplot(2,2,i)
    sns.kdeplot(data=df1, x=df1[column])
    plt.tight_layout()
```

- **In the Feature transformation, there are several features that are handled, namely Prior purchases as standardization values and Customer care calls features, Cost of the product, and Weight in gms as normalization values. after getting a new feature, the original value feature will be dropped.**

E. Feature Encoding

Machine learning models can only work with numerical values. For this reason, it is necessary to transform the categorical values of the relevant features into numerical ones. This process is called feature encoding.

```
mapping_Product_importance = {
  'low' : 0,
  'medium' : 1,
  'high' : 2
}
mapping_Gender = {
  'F' : 0,
  'M' : 1
}
mapping_Discount_offered = {
  '25%' : [int(x) for x in np.linspace(1,25,25)],
  '50%' : [int(x) for x in np.linspace(26,50,25)],
  '75%' : [int(x) for x in np.linspace(51,75,25)]
}
df1['Product_importance'] =
df1['Product_importance'].map(mapping_Product_importance)
df1['Gender'] = df1['Gender'].map(mapping_Gender)
df1['Discount_offered'] = df1['Discount_offered'].map(mapping_Discount_offered)
```

- *One hot encoder** One hot encoding can be defined as the essential process of converting the categorical data variables to be provided to machine and deep learning algorithms which in turn improve predictions as well as classification accuracy of a mode

```
for cat in ['Mode_of_Shipment', 'Warehouse_block']:
   onehots = pd.get_dummies(df1[cat], prefix=cat)
   df1 = df1.join(onehots)
```

```
# drop kolom yang di encoding menggunakan OHE : 'Mode_of_Shipment' &
'Warehouse_block'
df1 = df1.drop(columns=['Mode_of_Shipment', 'Warehouse_block'])
df1 = df1.drop(columns=['ID'])
```

- **In Feature Encoding, there are several features that will be labeled as encoding and one hot endcoder including. gender feature, discount offered and product importance will labeled as encoding with mapping method, and then for mode of shipment feature and warehouse block will be one hot encoder. and there are also some feature no need to we do feature encoding such the customer rating feature.and after that the original value coloumn already in the one hot encoder will be dropped.**

F. Handle class imbalance

handle Imbalanced Classification Problems

```
df1['Reached.on.Time_Y.N'].value_counts()
```

```
Result : 1    6461
       0    4360
Name: Reached.on.Time_Y.N, dtype: int64
```

```
df1['Reached.on.Time_class'] = df1['Reached.on.Time_Y.N'] > 0.8
print(df1['Reached.on.Time_class'].value_counts())
```

```
Result:
True     6461
False    4360
Name: Reached.on.Time_class, dtype: int64
```

```
X = df1[[col for col in df1.columns if col not in['Reached.on.Time_class',
'Reached.on.Time_Y.N']]].values
y = df1['Reached.on.Time_class'].values
print(X.shape)
print(y.shape)
```

```
Result : (10821, 16)
        (10821,)
```

```
from imblearn import under_sampling, over_sampling
X_over, y_over = over_sampling.RandomOverSampler(1).fit_resample(X, y)
print(pd.Series(y_over).value_counts())
```

```
Result : True     6461
False    6461
dtype: int64
/usr/local/lib/python3.7/dist-packages/imblearn/utils/_validation.py:591:
FutureWarning: Pass sampling_strategy=1 as keyword args. From version 0.9
passing these as positional arguments will result in an error FutureWarning,
```

```
Train_test_split
X = df1.drop(columns=['Reached.on.Time_Y.N'])
y = df1['Reached.on.Time_Y.N']
from sklearn.model_selection import train_test_split
# X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2,
random_state = 42)
```

- **In handle class imbalance , the method we using is oversampling**

Feature Engineering

A. Feature Selection(remove irrelevant or redundant features)

- There are some features that will be deleted, like ID according to our logic ID will not affected if the item gonna be late or not, and then the original value from Discount_offered,Mode_Of_shipment,Warehouse_block, Cost_of_the_Product, Weight_in_gms because the transformation has been done.

B.Feature extraction (create new features from existing features)

- the Prior_purchases, Customer_care_calls, Cost_of_the_Product and Weight_in_gms feature transformation is performed result in new features in the form of standardization and normalization values. Feature encoding is performed on the Mode_of_Shipment and Warehouse_block features, the result in a new feature with the values '0' and '1'

C. Adding 4 additional features

- Customer location (If realized can be based on warehouse block)
- Calculation of how many days late is based on a predetermined estimate
- Type of delivery (ex: express, regular)
- Delivery service
- Delivery date
- Weather

Source

Source	Link
Kaggle Dataset Download	[https://www.kaggle.com/datasets/prachi13/customer-analytics
Google Collabs	[https://colab.research.google.com/drive/1_OKxNUFYdj2M4vRm0T EXsnR?usp=sharing#scrollTo=c2x-2fKUOGQ2]