

**Study on Online Retail Data Set**

Submitted by

**Nabin Kumar Adhikari**

**Registration No. : 11617707**

**Section : KEM39**

**Roll No : B42**

**School of Computer Science & Engineering**

**Lovely Professional University, Phagwara**

**Table of Contents**

1. Introduction

2. About The Online Retail DataSet

3. Cleaning The Data

4. Train and Test Split

5.Training and Testing the Data

5. Conclusions And Results

6. References

**INTRODUCTION**

Online retail industry is a blooming and a very important aspect of modern economy. The recent years has seen a significant rise in online commercial transactions. This pattern suggests that this industry is becoming an important part of economy. While sellers look for profits no matter the mode of sell, customers search for value for price and convenience. This fact affects the customer as well as the seller’s behaviours and notable path to be taken by both parties. Particularly, the seller will be more concerned to boost his/her sales and attract more customers. For this the study of the market must be done. The analysis of the commercial market for this very reason is known as Market Basket Analysis. For this project we will be using the Online Retail DataSet provided by UCI Machine Learning Repository.

The Online Retail DataSet is a transactional data record of all the transactions occurring between

1st December,2010 and 9th December,2011 for a UK-based and registered non-store online retail.The company whose this data belongs to mainly sells unique all-occasion gifts. The customers are mostly wholesalers while few are retailers.The data can be found for study or download on the following link :

<https://archive.ics.uci.edu/ml/datasets/online+retail>

**ABOUT THE DATA**

The provided data is in excel format with ‘.xlsx’ extension. This can however also be converted to ‘.csv’ format which is also done throughout the project. The data set is a Multivariate, Sequential and Time-Series type of data with about 541909 instances. The full data however is cropped into 1000 instances for simplicity while preparing the model for training and testing. Nevertheless, the model can also be used with all the instances though that may lead to more time consumption due to the huge amount data to be processed. Further, the data had 8 attributes with an extra column for Indexing purposes named ‘Sn’. The associated tasks related to this data are Classification or Clustering.

The attributes of the data with their particular information are given below :

InvoiceNo : Invoice number. Data type is nominal, a 6-digit integral number uniquely assigned

to each transaction. If this code starts with letter 'c', it indicates a cancellation.

StockCode : Product (item) code. Data type is nominal, a 5-digit integral number uniquely

assigned to each distinct product.

Description: Product (item) name. Data type is nominal.

Quantity : The quantities of each product (item) per transaction. Data type is numeric.

InvoiceDate: Invoice Date and time. Data type is numeric, the day and time when each

transaction was generated.

UnitPrice : Unit price. Data type is numeric, Product price per unit in sterling.

CustomerID: Customer number. Data type is nominal, a 5-digit integral number uniquely

assigned to each customer.

Country : Country name. Data type is nominal, the name of the country where each customer

resides.

**CLEANING DATA**

The data has undergone few cleaning steps in which empty rows were removed.Striping Spaces from Descriptions, removing Rows that don’t have InvoiceNo, removing Rows that have Cancelled Transactions, removing waste transactions. The waste transactions were the ones with some specific keywords in the description of the data. Some of the waste keywords used were as

["WRONG","LOST", "CRUSHED", "SMASHED", "DAMAGED", "FOUND", "THROWN", "MISSING","AWAY","\\?","CHECK","POSTAGE","MANUAL", "CHARGES", "DAMAGES", "FEE", "FAULT", "SALES", "ADJUST", "COUNTED","LABEL","INCORRECT","SOLD", "BROKEN", "BARCODE", "CRACKED", "RETURNED", "MAILOUT", "DELIVERY","MIX UP", "MOULDY", "PUT ASIDE", "ERROR", "DESTROYED", "RUSTY"]

After this the data had some significantly reduced rows which means the data cleaning had done pretty good.

Following is the code snippet of the clean.py for cleaning of the data:

***print("CLEANING DATA...")***

***print("\tCurrent Shape of Data : ",np.shape(data))***

***print("\n\tStriping Spaces from Descriptions...")#***

***data['Description'] = data['Description'].str.strip()***

***print("Current Shape :",np.shape(data))***

***print("\tRemoving Rows that dont have Invoice No...")#***

***data.dropna(axis=0, subset=['InvoiceNo'], inplace=True)***

***data.dropna(axis=0, subset=['Description'], inplace=True)***

***print("Current Shape :",np.shape(data))***

***print("\tRemoving Rows that have Cancelled Transactions...")#***

***data['InvoiceNo'] = data['InvoiceNo'].astype('str')***

***data = data[~data['InvoiceNo'].str.contains('C')] #C=Cancelled Transactions***

***#data = data[~data['InvoiceNo'].str.contains('B')]***

***print("Current Shape :",np.shape(data))***

***print("\tRemoving WASTE Transactions...")#***

***waste = ["WRONG","LOST", "CRUSHED", "SMASHED", "DAMAGED", "FOUND", "THROWN", "MISSING", "AWAY", "\\?",***

***"CHECK", "POSTAGE", "MANUAL", "CHARGES", "DAMAGES", "FEE", "FAULT", "SALES", "ADJUST", "COUNTED",***

***"LABEL","INCORRECT", "SOLD", "BROKEN", "BARCODE", "CRACKED", "RETURNED", "MAILOUT", "DELIVERY",***

***"MIX UP", "MOULDY", "PUT ASIDE", "ERROR", "DESTROYED", "RUSTY"]***

***waste = '|'.join(waste)***

***data= data[-data['Description'].str.contains(waste)]***

***print("Current Shape :",np.shape(data))***

**TRAIN TEST SPLIT**

The train test split can be done either using the module called ‘train\_test\_split’ from ‘sklearn.model\_selection’ or, by using the splitter made during this project. The function is also called as ‘train\_test\_split1’ which takes the original data and the test size as input. This returns test and train split as inputs and target for the model training.

The code snippet of this function from the actual project is given below :

***def train\_test\_split(data,percent):***

***print("\nRunning Train\_Test\_Split...\n")***

***global row; //int with row number in the input data***

***test\_size= int((percent/100)\*row)***

***train\_size=row-test\_size***

***print(" The size of Training Data is : ",train\_size)***

***print(" The size of Test Data is : ",test\_size,)***

***print(" Total : ",(train\_size+test\_size))***

***length=0***

***rows\_t=[]***

***rows\_s=[]***

***print("\n\tRandomly Choosing Training Data...\n")***

***train=pd.DataFrame(rows\_t,columns=names)***

***while(length<train\_size):***

***x1=np.random.choice(data['Sn'])***

***if (((x1==train.Sn).any())==False):***

***#print("DATA : ",data.iloc[x1-1:x1,:])***

***dict1=(data.iloc[x1-1:x1,:]).to\_dict(orient='dict')***

***rows\_t.append(dict1)########***

***length=length+1***

***train=pd.DataFrame(rows\_t,columns=names)***

***#print(np.shape(train))***

***else:***

***print("",end="")***

***length=0***

***test=pd.DataFrame(rows\_s,columns=names)***

***print("\tRandomly Choosing Test Data...\n")***

***while(length<test\_size):***

***x1=np.random.choice(data['Sn'])***

***if (((x1==train.Sn).any())==False) and (((x1==test.Sn).any())==False):***

***dict2=(data.iloc[x1-1:x1,:]).to\_dict(orient='dict')***

***rows\_s.append(dict2)***

***length=length+1***

***test=pd.DataFrame(rows\_s,columns=names)***

***print("\t\tThe Training and Test Data ARE Split")***

***return train,test***

The cleaned data was taken in for split. The data was reformatted into a Pandas DataFrame using ‘.pivot\_table’ in which the ‘CustomerID’ was used as Index, ‘Descriptions’ as Columns and ‘Quantity’ as values within the DataFrame. The empty values were filled with 0. This was done sorting the data from country basis because the entire dataset huge. This DataFrame now had a detailed information on customers about their purchases related to an item. The Code snippet is shown below for United Kingdom :

***\_uk =(data[data['Country'] == "United Kingdom"]***

***.pivot\_table(index="CustomerID",***

***columns="Description",***

***values="Quantity",***

***aggfunc="sum",***

***fill\_value=0))***

This DataFrame was then used to make inputs and targets for training and testing. The ‘CustomerID’ was taken as Input, while the ‘Descriptions’ were taken as target. The values mapped between them was the ‘Quantity’ attribute. The inputs and targets were then feed for split in train and test data.

**TRAINING AND TESTING**

The training was done using DecisionTreeClassifier. The inputs were given along with the target for training. The whole dataset was huge so the training can take significant amount of time with heavy performance hit of the device due to training process.

The max\_depth of the tree was suitable between 4-6 which produces good results.

The testing included predicting the output from test data split. The misclassified samples are also displayed using the following code :

***print("\nThe Misclassified Samples per Descriptions are : \n",np.sum(np.not\_equal(y\_test, predictions)))***

The ‘accuracy\_score’ of skearn cannot be used here due to multiclass-multioutput not being supported in sklearn.

**CONCLUSIONS AND RESULTS**

The output was pretty good given the scale of data columns with actual data being less per sample. There were misclassified samples from 0 to one ninth of the sample size, which boasts significant accuracy in the prediction of samples. The predicted output can be mapped with actual output for a proper graphical representation.

Further, the data was analyzed using ‘mlxtend’ module to observe the frequencies and the similarities between transactions. The observation first saw mapping of data as ‘Description’ for columns and ‘InvoiceNo’ as Index while using ‘Quantity’ as the values. In this mapping the data with 0 quantities were removed and the quantities less than 0 (Returned Items) were made equal to 0, while others to 1. This mapping now indicated the relationship between different items purchased by various customers either separately or together. This also gave information on likeliness of the customer on what they will purchase together if they buy an item. The mapping was done on data from country ‘United Kingdom’ only , however this can be applied to other countries as well. There shall be an interesting observation while comparing mapping from different countries. This will give the behaviour of the customers of a country of different countries. The comparison was done using ‘apriori’ module from mlxtend.

This gave us some set of rules as an observation on behaviour of customers on various purchases.Some rules are given below:

‘GLASS STAR FROSTED T-LIGHT HOLDER’ and ‘CREAM CUPID HEARTS COAT HANGER’ have following statistics:

|  |  |  |
| --- | --- | --- |
| support | confidence | lift |
| 0.08620689 | 1 | 11.6 |

This indicates that there is almost full certainty that the customer who purchases 1st item will also purchase the 2nd item.

'KNITTED UNION FLAG HOT WATER BOTTLE' and 'CREAM CUPID HEARTS COAT HANGER’ have following statistics :

|  |  |  |
| --- | --- | --- |
| support | confidence | lift |
|  |  |  |
|  |  |  |
|  |  |  |
| 0.08620689 | 0.833333333 | 9.666667 |

This indicates that there is almost 83 percent certainty that the customer who purchases 1st item will also purchase the 2nd item.

**REFERENCES**

<https://archive.ics.uci.edu/ml/datasets/online+retail>

<https://pandas.pydata.org/pandas-docs/stable/reference/api.html>

<https://stackoverflow.com/questions/16476924/how-to-iterate-over-rows-in-a-dataframe-in-pandas?rq=1>

<https://stackoverflow.com/questions/36921951/truth-value-of-a-series-is-ambiguous-use-a-empty-a-bool-a-item-a-any-o>

<https://stackoverflow.com/questions/11707586/how-do-i-expand-the-output-display-to-see-more-columns>

<https://marketbasketlearning.wordpress.com/decision-trees/market-basket-analysis/>

<https://towardsdatascience.com/preprocessing-large-datasets-online-retail-data-with-500k-instances-3f24141f511>