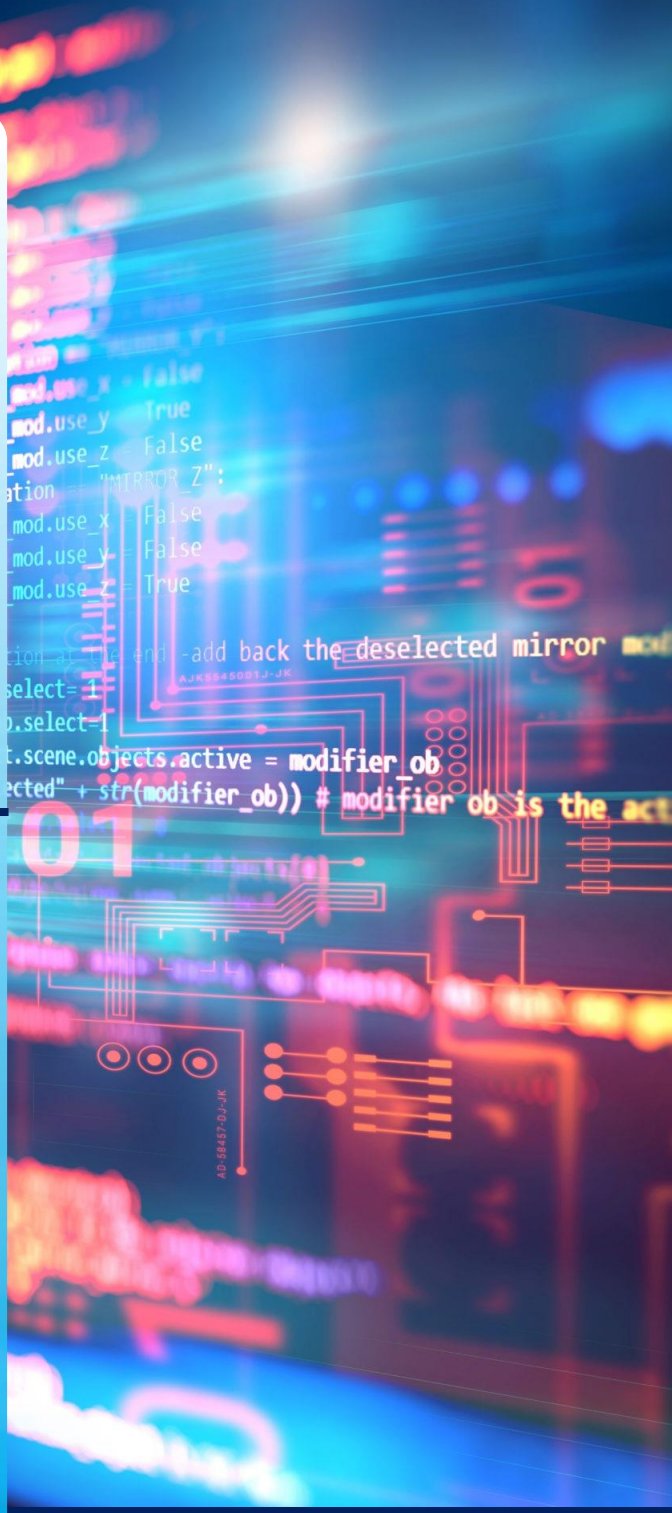
Online Retail Recommendation System

Using Python



July 22, 2023

INTRAINZ EDUTECH

Major project python code

Authored by: RAJ SHEKHAR SINHA

InTrAinZ

IGNITING A BRIGHTER FUTURE

Online Retail Recommendation System

Essential imports and outlier removal

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity


Retail_data=pd.read_excel("OnlineRetail.xlsx",index_col=[0])


q1=Retail_data['Quantity'].quantile(.25)
q3=Retail_data['Quantity'].quantile(.75)
iqr=q3-q1
lower_limit=q1-1.5*iqr
upper_limit=q3+1.5*iqr
lower_limit=q1-1.5*iqr
upper_limit=q3+1.5*iqr
def limit_imputer(value):
    if value<lower_limit:
        return lower_limit
    if value>upper_limit:
        return upper_limit
    else:
        return value
```

Popularity based recommendation system

```
Retail_data['Quantity']=Retail_data['Quantity'].apply(limit_imputer)

Retail_data['StockCode']=Retail_data['StockCode'].astype(str)

Num_of_times_item_sold=Retail_data.groupby('Description').count()['Quantity'].reset_index()

Num_of_times_item_sold.rename(columns={'Quantity':'Times_sold'},inplace=True)

Avg_quantity_of_each_item=Retail_data.groupby('Description')['Quantity'].mean().reset_index()

Avg_quantity_of_each_item.rename(columns={'Quantity':'Avg_quantity_sold'},inplace=True)

Popular_item=Num_of_times_item_sold.merge(Avg_quantity_of_each_item,on='Description')

#Only Those items will be considered which were sold on al least 100 different #instances

Popular_item=Popular_item[Popular_item['Times_sold']>=100].sort_values('Avg_quantity_sold',ascending=False)

Popular_item=Popular_item.merge(Retail_data,on='Description').drop_duplicates('Description')[['StockCode', 'Description','UnitPrice', 'Times_sold', 'Avg_quantity_sold']].reset_index()


list_of_globally_popular=list(Popular_item['Description'])
```

Collaborative filtering-based Recommendation system

Country based recommendation

```
def recommend_by_country(Description):
    l=[]
    index=np.where(popular_by_country.index==Description)[0][0]
    similar_items=sorted(list(enumerate(similarity_score[index])),key=lambda x:x[1],reverse=True)[1:50]
    for i in similar_items:
        l.append(popular_by_country.index[i[0]])
    return l


Retail_data=Retail_data.dropna()
#only those items will be considered which were sold on at least
#100 different instances or transactions
x=Retail_data.groupby('Description').count()['Country']>=100
considered_items=x[x].index
filterd_items=Retail_data[Retail_data['Description'].isin(considered_items)]
#Only those countries will be selected where number of times any item
#sold is at least 300
y=filterd_items.groupby('Country').count()['Description']>=300
top_countries=y[y].index
filterd_items=filterd_items[filterd_items['Country'].isin(top_countries)]
popular_by_country=filterd_items.pivot_table(index='Description',columns='Country',values='Quantity')
popular_by_country.fillna(0,inplace=True)
similarity_score=cosine_similarity(popular_by_country)
```

Month based recommendation

```
def recommend_by_month(Description):
    l1=[]
    index=np.where(popular_by_month.index==Description)[0][0]

similar_items_by_month=sorted(list(enumerate(similarity_score_by_month[ind
ex])),key=lambda x:x[1],reverse=True)[1:50]
    for i in similar_items_by_month:
        l1.append(popular_by_month.index[i[0]])
    return l1


Retail_data['Purchase_Month'] =
pd.DatetimeIndex(Retail_data['InvoiceDate']).month
months={1:'January',2:'February',3:'March',4:'April',5:'May',
        6:'June',7:'July',8:'August',9:'September',10:'October',
        11:'November',12:'December'}
Retail_data['Purchase_Month']=Retail_data['Purchase_Month'].map(months)
Retail_data.groupby('Purchase_Month').count()['Description'].sort_values(ascen
ding=False)
Retail_data=Retail_data.dropna()
#only those items will be considered which were sold on at least
#100 different instances or transactions
m=Retail_data.groupby('Description').count()['Purchase_Month']>=100
considered_items_by_month=m[m].index
filterd_items_by_month=Retail_data[Retail_data['Description'].isin(considered_
items_by_month)]
popular_by_month=filterd_items_by_month.pivot_table(index='Description',col
umns='Purchase_Month',values='Quantity')
popular_by_month.fillna(0,inplace=True)
similarity_score_by_month=cosine_similarity(popular_by_month)
```

Final recommendation

```
#Enter product name in place of ROBOT BIRTHDAY CARD
Product=r"ROBOT BIRTHDAY CARD"
list_of_popular_by_month=recommend_by_month(Product)
list_of_country_wise_popular=recommend_by_country(Product)
most_popular=[]
for i in list_of_country_wise_popular:
    for j in list_of_popular_by_month:
        if i==j:
            most_popular.append(j)
if most_popular!=False:
    print(f"You know what else you should buy\nalongside a {Product}, these:-
\n")
    for i in most_popular:
        print(i)
else:
    print(list_of_globally_popular[:20])
```

Sample Outputs: -

```
You know what else you should buy
alongside a ROBOT BIRTHDAY CARD, these:-

SPACEBOY GIFT WRAP
SPACEBOY BIRTHDAY CARD
6 RIBBONS RUSTIC CHARM
JUMBO BAG SPACEBOY DESIGN
TEA PARTY BIRTHDAY CARD
CARD CIRCUS PARADE
ROUND SNACK BOXES SET OF 4 FRUITS
```

```
You know what else you should buy
alongside a PINK POLKADOT WRAP, these:-

BLUE POLKADOT WRAP
RED RETROSPOT WRAP
MINI PAINT SET VINTAGE
WRAP ENGLISH ROSE
PACK OF 72 RETROSPOT CAKE CASES
WRAP RED APPLES
HOMEMADE JAM SCENTED CANDLES
WRAP DOLLY GIRL
WRAP PINK FAIRY CAKES
WRAP ALPHABET DESIGN
SPACEBOY GIFT WRAP
```