* DS_PROJECT_72@360DIGITMG

• CREDIT SCORING MODEL FOR RETAILERS

```
#Importing Libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.

import time, warnings
import datetime as dt

#visualizations
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
import klib
warnings.filterwarnings("ignore")

#impoing excel file
```

```
retail_df =
pd.read_excel("C:\\Users\shamanth\Downloads\CreditAnalysis_data
.xlsx")

retail_df.head()

retail_df.describe()

retail_df.isnull().sum()

klib.dist_plot(retail_df)

klib.corr_mat(retail_df)

retail_df.drop(['master_order_id','master_order_status'],inplace=Tru
e,axis = 1)

retail_df.info()
```

Data Insights

```
##Since the data is from an cridite analysis, we will look at where
their order_id originate from i.e. 'group'

# filter retailiers by top 10 states in percentage

retail_df.group.value_counts(normalize=True)[:10]

# visualize in bar chart

retail_df.group.value_counts(normalize=True)[:10].plot(kind="bar")

from dataprep.eda import create_report

create_report(retail_df)

retail_df.columns ## treating missing values

#remove rows where orderid are NA

retail_df.dropna(subset=['ordereditem_product_id'],how='all',inplac
e=True)

retail_df.shape
```

```
retail_df['created'] = retail_df['created'].dt.strftime('%Y-%m-%d')

retail_df.created = retail_df.created.astype('str')

retail_df.dtypes


#restrict the data to one full year because it's better to use a metric per Months or Years in RFM

retail_df = retail_df[retail_df['created']>= "2018-3-31"]

retail_df.shape


# # RFM Analysis

# RFM (Recency, Frequency, Monetary) analysis is a customer segmentation technique that uses past purchase behavior to divide customers into groups. RFM helps divide customers into various categories or clusters to identify customers who are more likely to respond to promotions and also for future personalization services.

# RECENCY (R): Days since last purchase

# FREQUENCY (F): Total number of purchases

# MONETARY VALUE (M): Total money this customer spent.

# We will create those 3 customer attributes for each customer.

# # Recency

# To calculate recency, we need to choose a date point from which we evaluate how many days ago was the retailers's last order.

#last date available in our dataset

retail_df['created'].min()

#create a new column called date which contains the date of invoice only
```

```
retail_df['date'] = pd.DatetimeIndex(retail_df['created']).date

retail_df.head()

#group by customers and check last date of purshace

recency_df = retail_df.groupby(by='retailer_names',
as_index=False)['date'].max()

recency_df.columns = ['Retailernames','LastPurshaceDate']

recency_df.head()

now = dt.date(2018,3,31)

print(now)


#calculate recency

recency_df['Recency'] =
recency_df['LastPurshaceDate'].apply(lambda x: (now - x).days)

recency_df.head()

#drop LastPurchaseDate as we don't need it anymore

recency_df.drop('LastPurshaceDate',axis=1,inplace=True)


# Frequency

# Frequency helps us to know how many times a customer
purchased from us. To do that we need to check how many orders
are registered by the same retailers.


# drop duplicates

retail_df_copy = retail_df
```

```
retail_df_copy.drop_duplicates(subset=['order_id', 'retailer_names'],
keep="first", inplace=True)

#calculate frequency of purchases

frequency_df = retail_df_copy.groupby(by=['retailer_names'],
as_index=False)['order_id'].count()

frequency_df.columns = ['Retailernames','Frequency']

frequency_df.head()


# Monetary

# Monetary attribute answers the question: How much money did
the retailer spent over time?

# To do that, first, we will create a new col

#create column total cost

retail_df['TotalCost'] = retail_df['ordereditem_quantity'] *
retail_df['ordereditem_unit_price_net']


monetary_df =
retail_df.groupby(by='retailer_names',as_index=False).agg({'TotalCos
t': 'sum'})

monetary_df.columns = ['Retailernames','Monetary']

monetary_df.head()


# # Create RFM Table

#merge recency dataframe with frequency dataframe

temp_df = recency_df.merge(frequency_df,on='Retailernames')
```

```
temp_df.head()

#merge with monetary dataframe to get a table with the 3 columns
rfm_df = temp_df.merge(monetary_df,on='Retailernames')
#use retailersID as index
rfm_df.set_index('Retailernames',inplace=True)
#check the head
rfm_df.head()
retail_df[retail_df['retailer_names']=='RetailerID1']

#RFM Quartiles
quantiles = rfm_df.quantile(q=[0.25,0.5,0.75])
quantiles

quantiles.to_dict()
# Creation of RFM Segments
# Arguments (x = value, p = recency, monetary_value, frequency, d =
quartiles dict)
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
```

```
        return 2
    else:
        return 1
# Arguments (x = value, p = recency, monetary_value, frequency, k =
quartiles dict)
def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4
#create rfm segmentation table
rfm_segmentation = rfm_df
rfm_segmentation['R_Quartile'] =
rfm_segmentation['Recency'].apply(RScore,
args=('Recency',quantiles,))
rfm_segmentation['F_Quartile'] =
rfm_segmentation['Frequency'].apply(FMScore,
args=('Frequency',quantiles,))
rfm_segmentation['M_Quartile'] =
rfm_segmentation['Monetary'].apply(FMScore,
args=('Monetary',quantiles,))
```

```
rfm_segmentation.head()
```

Now that we have the score of each customer, we can represent our customer segmentation. First, we need to combine the scores (R_Quartile, F_Quartile,M_Quartile) together.

```
rfm_segmentation['RFMScore'] =
rfm_segmentation.R_Quartile.map(str)  +
rfm_segmentation.F_Quartile.map(str)   +
rfm_segmentation.M_Quartile.map(str)
```

```
rfm_segmentation.head()
```

```
rfm_segmentation.tail()
```

Best Recency score = 4: most recently purchase. Best Frequency score = 4: most quantity purchase. Best Monetary score = 4: spent the most.

Let's see who are our Champions (best retailers).

```
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary', ascending=False).head(10)
```

```
print("Best Retailers:
",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Loyal Retailers:
',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Big Spenders:
",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Almost Lost: ',
len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
```

print('Lost Retailers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))

print('Lost Cheap Retailers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))

out put

jupyter CRIDITE SCORING MODEL OF RETAILERS Last Checkpoint: Last Wednesday at 12:40 PM (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Run Code

In [1]:
```
#Importing Libraries
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.

import time, warnings
import datetime as dt

#visualizations
import matplotlib.pyplot as plt
from pandas.plotting import scatter_matrix
get_ipython().run_line_magic('matplotlib', 'inline')
import seaborn as sns
import klib
warnings.filterwarnings("ignore")
```

In [2]:
```
retail_df = pd.read_excel("C:\\Users\shamanth\Downloads\CreditAnalysis_data.xlsx")
```

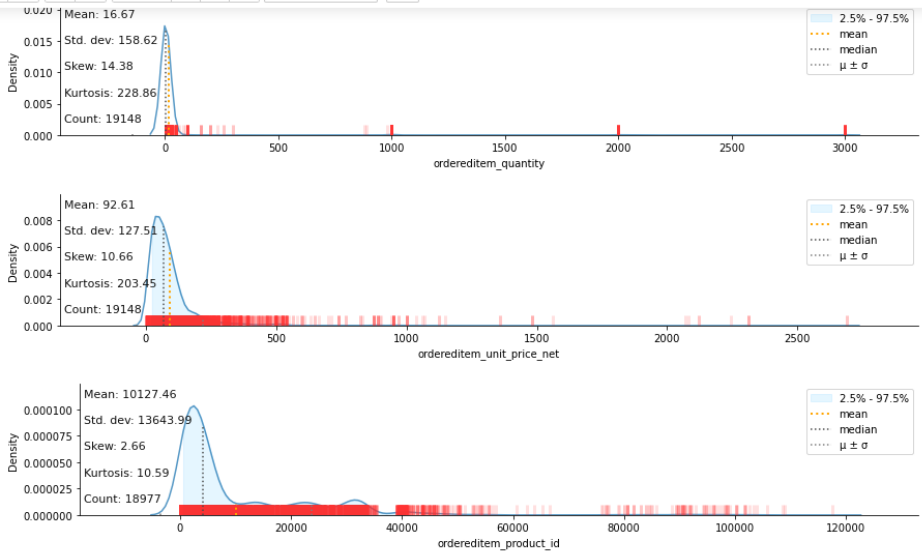
In [3]:
```
retail_df.head()
```

Out[3]:

	Unnamed: 0	master_order_id	master_order_status	created	order_id	order_status	ordereditem_quantity	prod_names	ordereditem_unit_price_net	orderedit
0	1.0	25651.0	delivered	12/18/2017 10:41	25651.0	delivered	5.0	ProdName1	43.00	
1	2.0	25651.0	delivered	12/18/2017 10:41	25651.0	delivered	5.0	ProdName2	35.54	

jupyter CRIDITE SCORING MODEL OF RETAILERS Last Checkpoint: Last Wednesday at 12:40 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)



jupyter CRIDITE SCORING MODEL OF RETAILERS Last Checkpoint: Last Wednesday at 12:40 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

```
In [45]: create_report(retail_df)
```

DataPrep Report | Overview | Variables | Interactions | Correlations | Missing Values

Overview

Dataset Statistics

Number of Variables	15
Number of Rows	6689
Missing Cells	0
Missing Cells (%)	0.0%
Duplicate Rows	0
Duplicate Rows (%)	0.0%
Total Size in Memory	3.5 MB
Average Row Size in Memory	545.8 B
Variable Types	Numerical: 8 Categorical: 7

Dataset Insights

`value` and `TotalCost` have similar distributions	Similar Distribution
`ordereditem_quantity` is skewed	Skewed
`ordereditem_unit_price_net` is skewed	Skewed
`ordereditem_product_id` is skewed	Skewed
`value` is skewed	Skewed
`bill_amount` is skewed	Skewed
`TotalCost` is skewed	Skewed
`created` has a high cardinality: 878 distinct values	High Cardinality
`prod_names` has a high cardinality: 3464 distinct values	High Cardinality
`retailer_names` has a high cardinality: 204 distinct values	High Cardinality

1 2

jupyter CRIDITE SCORING MODEL OF RETAILERS Last Checkpoint: Last Wednesday at 12:40 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3 (ipykernel)

Out[39]:

Retailernames	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile
RetailerID1	0	82	17906.76	4	4	4
RetailerID10	1	17	2794.02	3	3	3
RetailerID100	0	19	5463.70	4	3	3
RetailerID101	0	114	18862.77	4	4	4
RetailerID102	5	2	407.14	1	1	1

In [40]:
```
rfm_segmentation['RFMScore'] = rfm_segmentation.R_Quartile.map(str)  + rfm_segmentation.F_Quartile.map(str)
rfm_segmentation.head()
```

Out[40]:

Retailernames	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore
RetailerID1	0	82	17906.76	4	4	4	444
RetailerID10	1	17	2794.02	3	3	3	333
RetailerID100	0	19	5463.70	4	3	3	433
RetailerID101	0	114	18862.77	4	4	4	444
RetailerID102	5	2	407.14	1	1	1	111

In [41]:
```
rfm_segmentation.tail()
# Best Recency score = 4: most recently purchase. Best Frequency score = 4: most quantity purchase. Best Monetary score = 4: sper
# Let's see who are our Champions (best retailers)
```

jupyter CRIDITE SCORING MODEL OF RETAILERS Last Checkpoint: Last Wednesday at 12:40 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Not Trusted Python 3

In [46]:
```
rfm_segmentation[rfm_segmentation['RFMScore']=='444'].sort_values('Monetary', ascending=False).head(5)
```

Out[46]:

Retailernames	Recency	Frequency	Monetary	R_Quartile	F_Quartile	M_Quartile	RFMScore
RetailerID13	0	525	108094.93	4	4	4	444
RetailerID37	0	284	63093.91	4	4	4	444
RetailerID34	0	275	62141.14	4	4	4	444
RetailerID21	0	125	59617.64	4	4	4	444
RetailerID35	0	265	57948.47	4	4	4	444

In [43]:
```
print("Best Retailers: ",len(rfm_segmentation[rfm_segmentation['RFMScore']=='444']))
print('Loyal Retailers: ',len(rfm_segmentation[rfm_segmentation['F_Quartile']==4]))
print("Big Spenders: ",len(rfm_segmentation[rfm_segmentation['M_Quartile']==4]))
print('Almost Lost: ', len(rfm_segmentation[rfm_segmentation['RFMScore']=='244']))
print('Lost Retailers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='144']))
print('Lost Cheap Retailers: ',len(rfm_segmentation[rfm_segmentation['RFMScore']=='111']))
```

```
Best Retailers:  43
Loyal Retailers:  51
Big Spenders:  51
Almost Lost:  0
Lost Retailers:  0
Lost Cheap Retailers:  28
```