

```
In [1]: #Import
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import datetime as dt
import seaborn as sns
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN

import plotly.express as px
import plotly.graph_objs as go
import plotly.io as pio
from plotly.subplots import make_subplots
import plotly.figure_factory as ff

import chart_studio as cs
import plotly.offline as po
import plotly.graph_objs as gobj
```

Attribute Information:

InvoiceNo: Invoice number. 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. Nominal, a 5-digit integral number uniquely assigned to each distinct product.

Description: Product (item) name. Nominal.

Quantity: The quantities of each product (item) per transaction. Numeric.

InvoiceDate: Invice Date and time. Numeric, the day and time when each transaction was generated.

UnitPrice: Unit price. Numeric, Product price per unit in sterling.

CustomerID: Customer number. Nominal, a 5-digit integral number uniquely assigned to each customer.

Country: Country name. Nominal, the name of the country where each customer resides.

در این مسئله به طور خلاصه به دنبال پیدا کردن بهترین مشتری ها هستیم. از روش

RFM

استفاده میکنیم

در این روش سه تا ویژگی به طور مجزا بررسی میشوند و میانگین این سه ویژگی به عنوان یک ویژگی کلی برای هر مشتری در نظر گرفته میشود.

اولین ویژگی:

Recency:

این ویژگی بیانگر فعالیت های مشتری در نزدیکترین زمان است(آخرین سفارش، آخرین تراکنش) واضح هست که مشتری هایی که در نزدیکترین زمان به امروز فعالیت داشتند در این مدل مشتری های بهتری هستند.

ویژگی دو

Frequency:

بسامد رفتار و فعالیت مشتری مثل خرید و مشاهده محصول و ... میباشد. اینجا هم مشخص است که هرچه این عدد بیشتر باشد مشتری بهتری خواهیم داشت

ویژگی آخر و سومی:

Monetary:

به عبارتی قدرت خرید مشتری میباشد. هرچه مشتری در تراکنش ها هزینه بالاتری پرداخت کند و با میانگین این هزینه ها بالاتر باشد مشتری بهتری محسوب میشود.

بنابراین قدم اول در پیدا کردن مشتری های خوب یافتن این سه پارامتر و سپس بررسی آن هاست.

اولین دسته بندی بر اساس ویژگی های مشترک تعدادی از مشتری هاست که در اینجا اون ویژگی -کشور - هست. بنابراین دسته بندی اول بر اساس کشور و یافتن بیشترین تعداد مشتری هست

```
In [2]: #Read Online Retail Data containing transactions from 01/12/2010 and 09/
data = pd.read_csv('Online Retail.csv', encoding = 'unicode_escape')

#unique Customer number per country
data[['CustomerID','Country']].drop_duplicates(subset=['CustomerID','Co
```

Out[2] :

Country	CustomerID
United Kingdom	3950
Germany	95
France	87
Spain	31
Belgium	25

به نظر میاد که تنها دیتای قابل بررسی بر اساس کشور ها انگلستان باشد. برای دو کشور بعدی هم شاید بتوان حرف هایی زد اما به لحاظ دقت و دیتای مورد نیاز کشور انگلستان نمونه مناسبی است

```
In [3]: #It seems the best country to work on,based on enough data is United Ki/
data=data[data['Country']=='United Kingdom'].reset_index(drop=True)
```

از آنجایی که بررسی نهایی بر روی هر تک مشتری انجام خواهد شد بنابراین داده های حالی رو از دیتا خارج میکنیم.

In [4]: `data[data['CustomerID'].isnull()]`

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Col
586	536414	22139	NaN	56	12/1/2010 11:52	0.00	NaN	U King
1298	536544	21773	DECORATIVE ROSE BATHROOM BOTTLE	1	12/1/2010 14:32	2.51	NaN	U King
1299	536544	21774	DECORATIVE CATS BATHROOM BOTTLE	2	12/1/2010 14:32	2.51	NaN	U King
1300	536544	21786	POLKA DOT RAIN HAT	4	12/1/2010 14:32	0.85	NaN	U King
1301	536544	21787	RAIN PONCHO RETROSPOT	2	12/1/2010 14:32	1.66	NaN	U King
...
495184	581498	85099B	JUMBO BAG RED RETROSPOT	5	12/9/2011 10:26	4.13	NaN	U King
495185	581498	85099C	JUMBO BAG BAROQUE BLACK WHITE	4	12/9/2011 10:26	4.13	NaN	U King
495186	581498	85150	LADIES & GENTLEMEN METAL SIGN	1	12/9/2011 10:26	4.96	NaN	U King
495187	581498	85174	S/4 CACTI CANDLES	1	12/9/2011 10:26	10.79	NaN	U King
495188	581498	DOT	DOTCOM POSTAGE	1	12/9/2011 10:26	1714.17	NaN	U King

133600 rows × 8 columns

In [5]: `#Check for missing values in the dataset and Remove missing values from
data = data[~data['CustomerID'].isnull()]`

در قسمت بعدی پیدا کردن مقادیر بی معنی هست. مثلا در مورد تعداد خرید ها اعداد منفی رو خارج میکنیم. در مورد قیمت ها هم همینطور عمل میکنیم و اگر عدد منفی وجود داشته باشه از دیتا خارج میکنیم.

```
In [6]: #Finding any meaningless values in Quantity and UnitPrice column
data['Quantity'].sort_values()
```

```
Out[6]: 494106    -80995
      57702    -74215
      4127     -9360
      147623    -3114
      147622    -2000
      ...
      189929     4300
      383650     4800
      458029    12540
      57697     74215
      494105    80995
Name: Quantity, Length: 361878, dtype: int64
```

```
In [7]: #it is Ok to have zero value for Unit Price
data['UnitPrice'].sort_values()
```

```
Out[7]: 256347      0.00
      32020      0.00
      327004     0.00
      382571     0.00
      442890     0.00
      ...
      246124    4287.63
      160147    6930.00
      160033    8142.75
      160138    8142.75
      204891   38970.00
Name: UnitPrice, Length: 361878, dtype: float64
```

به دست آوردن قیمت کل از ضرب تعداد به قیمت واحد بدست می آید

آخرین تراکنش هر فردی رو هم از آخرین داده ای که از تاریخ ها داریم بدست میاریم

```
In [8]: #Filter out records with meaningless values for Quantity
data = data[(data['Quantity']>0)].reset_index()

#Convert the string date field to datetime
data['InvoiceDate'] = pd.to_datetime(data['InvoiceDate'])

#total Price is:
data['TotalPrice'] = data['Quantity'] * data['UnitPrice']

#consider the last transaction time and a day after that for finding the
End_Date = dt.datetime(2011,12,10)
data['LastInvoice']=data.apply(lambda x : (End_Date - x['InvoiceDate']).
```

```
In [9]: data['InvoiceDate']
```

```
Out[9]: 0      2010-12-01 08:26:00  
1      2010-12-01 08:26:00  
2      2010-12-01 08:26:00  
3      2010-12-01 08:26:00  
4      2010-12-01 08:26:00  
...  
354340 2011-12-09 12:31:00  
354341 2011-12-09 12:49:00  
354342 2011-12-09 12:49:00  
354343 2011-12-09 12:49:00  
354344 2011-12-09 12:49:00  
Name: InvoiceDate, Length: 354345, dtype: datetime64[ns]
```

RFM Modelling

در این قسمت داده های ما برای گروپ شدن آماده هستند. برای هر مشتری و هر آی دی مستقل کوچکترین فاصله زمانی از آخرین تراکنش مشتری تا آخرین تراکنش کل داده ها ویژگی اول یا "Recency"

است. تعداد تراکنش های کل مشتری

"Frequency"

و جمع هزینه های پرداختی هر مشتری

"Monetary"

است

In [10]:

```
RFM=data.groupby('CustomerID', as_index=False).agg({'LastInvoice':'min',
RFM.columns=['CustomerID', 'Recency', 'Frequency', 'Monetary']

RFM
```

Out[10]:

	CustomerID	Recency	Frequency	Monetary
0	12346.0	325	1	77183.60
1	12747.0	2	103	4196.01
2	12748.0	0	4596	33719.73
3	12749.0	3	199	4090.88
4	12820.0	3	59	942.34
...
3916	18280.0	277	10	180.60
3917	18281.0	180	7	80.82
3918	18282.0	7	12	178.05
3919	18283.0	3	756	2094.88
3920	18287.0	42	70	1837.28

3921 rows × 4 columns

In [11]: RFM.info()

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3921 entries, 0 to 3920
Data columns (total 4 columns):
 #   Column      Non-Null Count  Dtype  
--- 
 0   CustomerID  3921 non-null   float64
 1   Recency     3921 non-null   int64  
 2   Frequency   3921 non-null   int64  
 3   Monetary    3921 non-null   float64
dtypes: float64(2), int64(2)
memory usage: 122.7 KB
```

In [12]: RFM.Recency.describe()

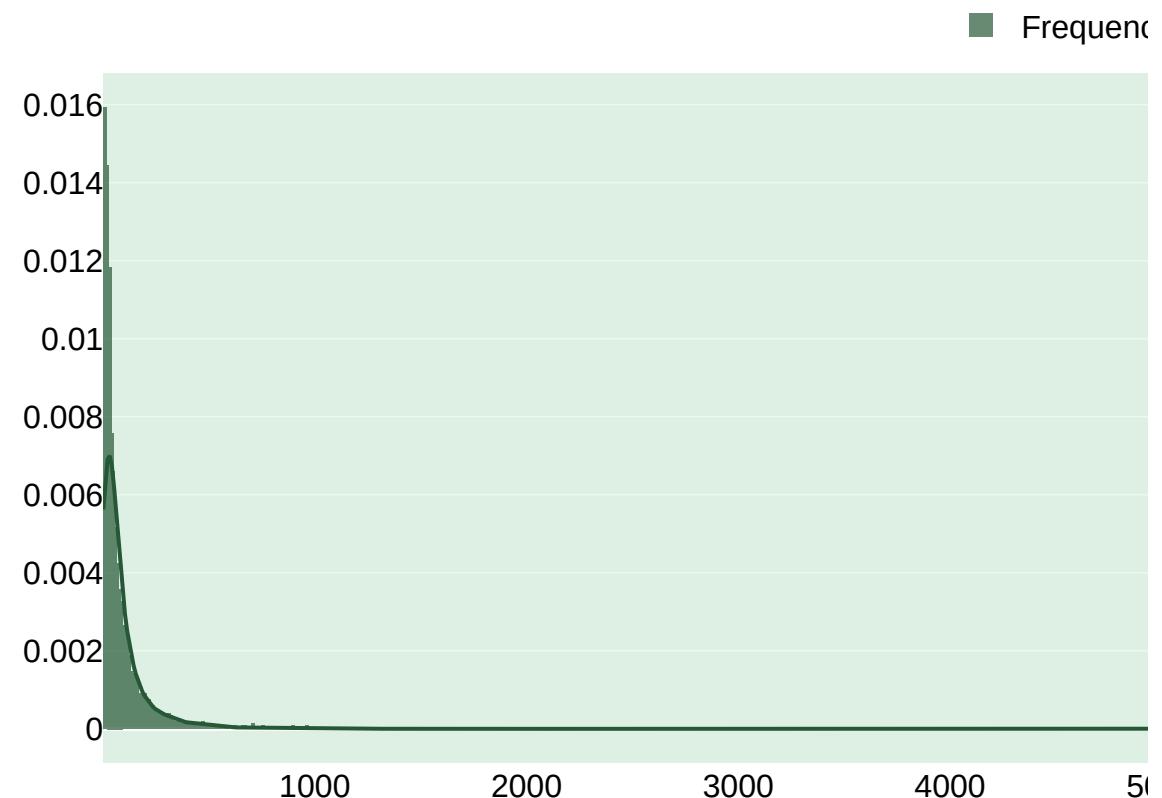
```
count    3921.000000
mean     91.722265
std      99.528532
min      0.000000
25%     17.000000
50%     50.000000
75%     142.000000
max     373.000000
Name: Recency, dtype: float64
```

برای هر ویژگی نمودار توزیع را رسم میکنیم:

```
In [13]: x = RFM[RFM['Frequency']<7000]['Frequency']
hist_data = [x]
group_labels = ['Frequency'] # name of the dataset

colors = ['#265736']
fig = ff.create_distplot(hist_data, group_labels, bin_size=10, show_rug=False)
colors = ['#265736']
fig.update_layout(width = 700, title="Distribution Plot (This plot is interactive)", font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor='#ddf0e3', legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.02,
    xanchor="right",
    x=1
))
fig.show()
```

Distribution Plot (This plot is interactive, you can zoom in/out)

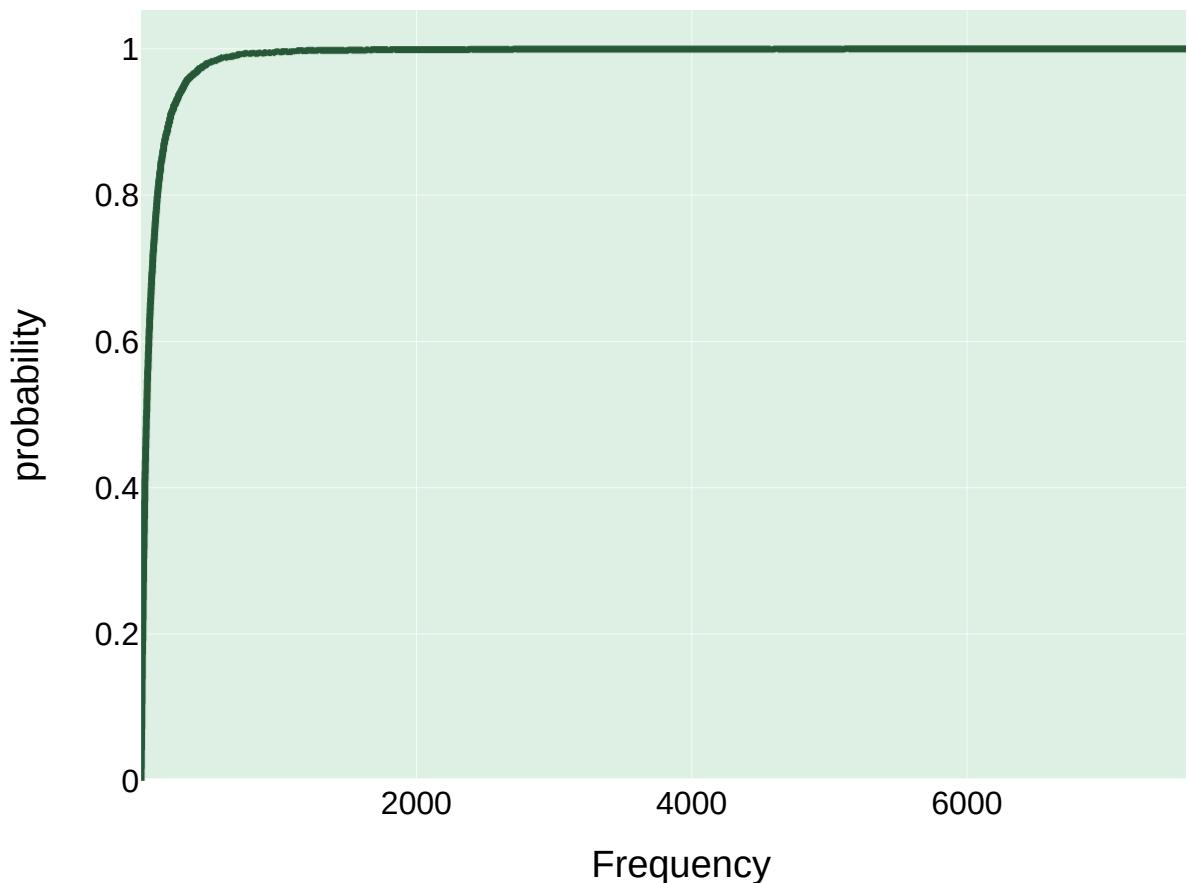


بیشترین توزیع برای بسامد تراکنش ها در مقادیر کمتر از ۵۰۰ دیده میشه. منحنی خطی روی داده ها یک

توزیع تخمین زده شده به کمک یک کرنل گوسی است که از واریانس و انحراف از معیار داده ها بدست میآید.

```
In [14]: df = RFM
fig = px.ecdf(df, x="Frequency")
fig.update_traces(line_color='#265736', line_width=3.5)
fig.update_layout(width = 700,title="Cumulative Distribution Plots",font
                  family="Arial",
                  size=16,
                  color='#000000'
                ), paper_bgcolor='rgba(0,0,0,0)',
                plot_bgcolor='#ddf0e3', legend=dict(
                orientation="h",
                yanchor="bottom",
                y=1.02,
                xanchor="right",
                x=1
              )
)
fig.show()
```

Cumulative Distribution Plots

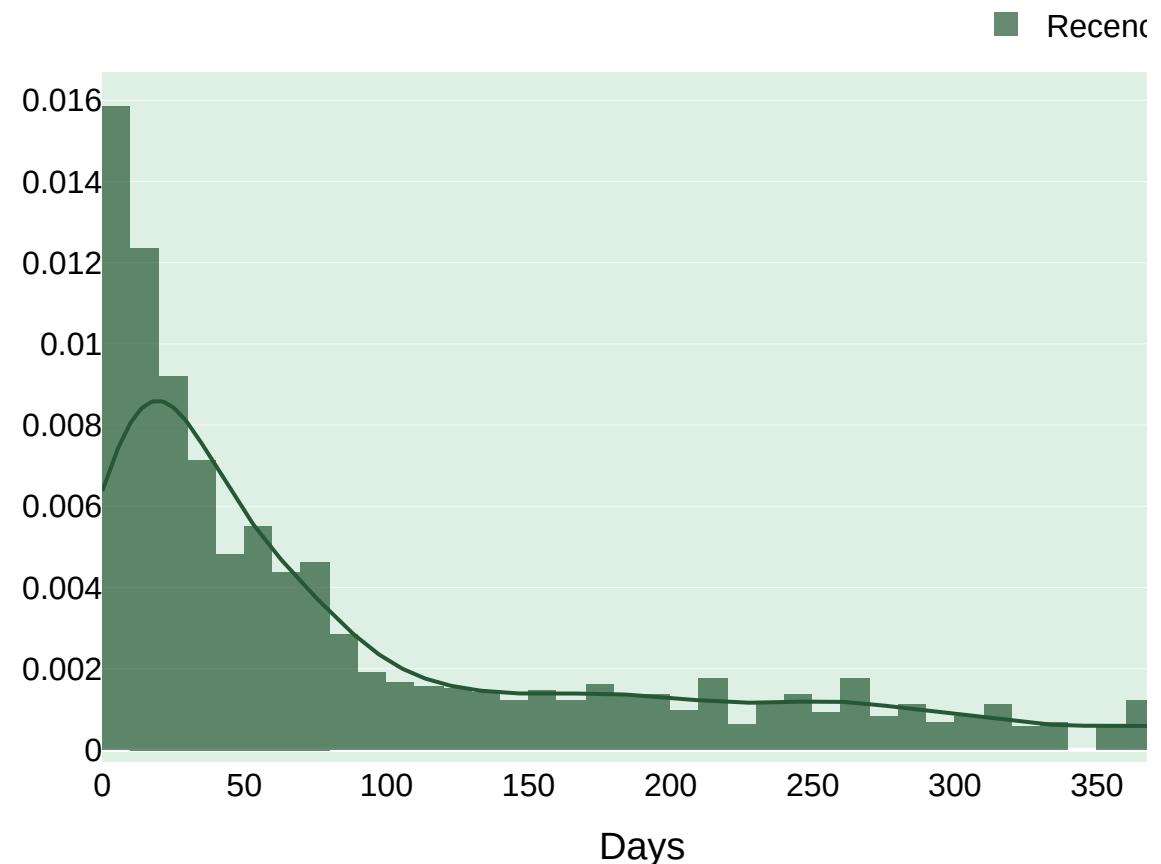


همانطور که انتظار میرفت در این قسمت هم نمودار تجمعی رشد زیاد تا بسامد ۳۰۰ را برای ۹۵ درصد از مشتری ها نشان میدهد. و درصد بسیار کمی از مشتری ها بسامد های بالا داشته اند (با توجه به یک ساله

(بودن داده ها و کوتاه بودن زمان به طور نسبی همین انتظار را داریم)

```
In [15]: x = RFM['Recency']
hist_data = [x]
group_labels = ['Recency'] # name of the dataset
# colors = ['red', 'blue']
fig = ff.create_distplot(hist_data, group_labels, bin_size=10, show_rug=False)
colors = ['#265736']
fig.update_layout(width = 700, title="Distribution Plot", font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.02,
    xanchor="right",
    x=1
))
fig.update_xaxes(title_text='Days')
fig.show()
```

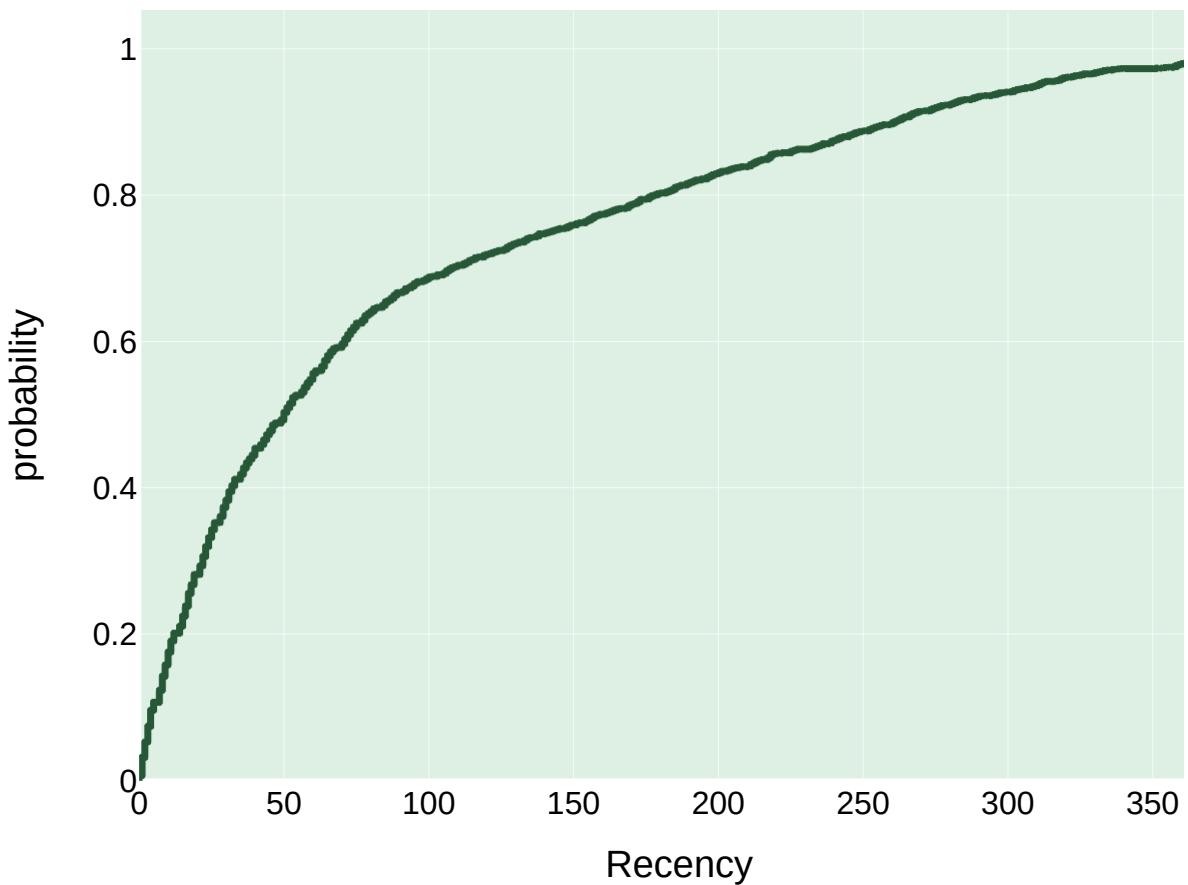
Distribution Plot



نمودار بالا نشان میده حدود نیمی مشتری ها در کمتر از ۵۰ روز پیش از تاریخ نهایی تراکنش داشته اند.
برای بررسی بیشتر این نمودار رو تجمعی نیز رسم میکنیم.

```
In [16]: df = RFM
fig = px.ecdf(df, x="Recency")
fig.update_traces(line_color='#265736', line_width=3.5)
fig.update_layout(width = 700,title="Cumulative Distribution Plots",font
                  family="Arial",
                  size=16,
                  color='#000000'
                ), paper_bgcolor='rgba(0,0,0,0)',
                plot_bgcolor='#ddf0e3', legend=dict(
                orientation="h",
                yanchor="bottom",
                y=1.02,
                xanchor="right",
                x=1
              )
)
fig.show()
```

Cumulative Distribution Plots



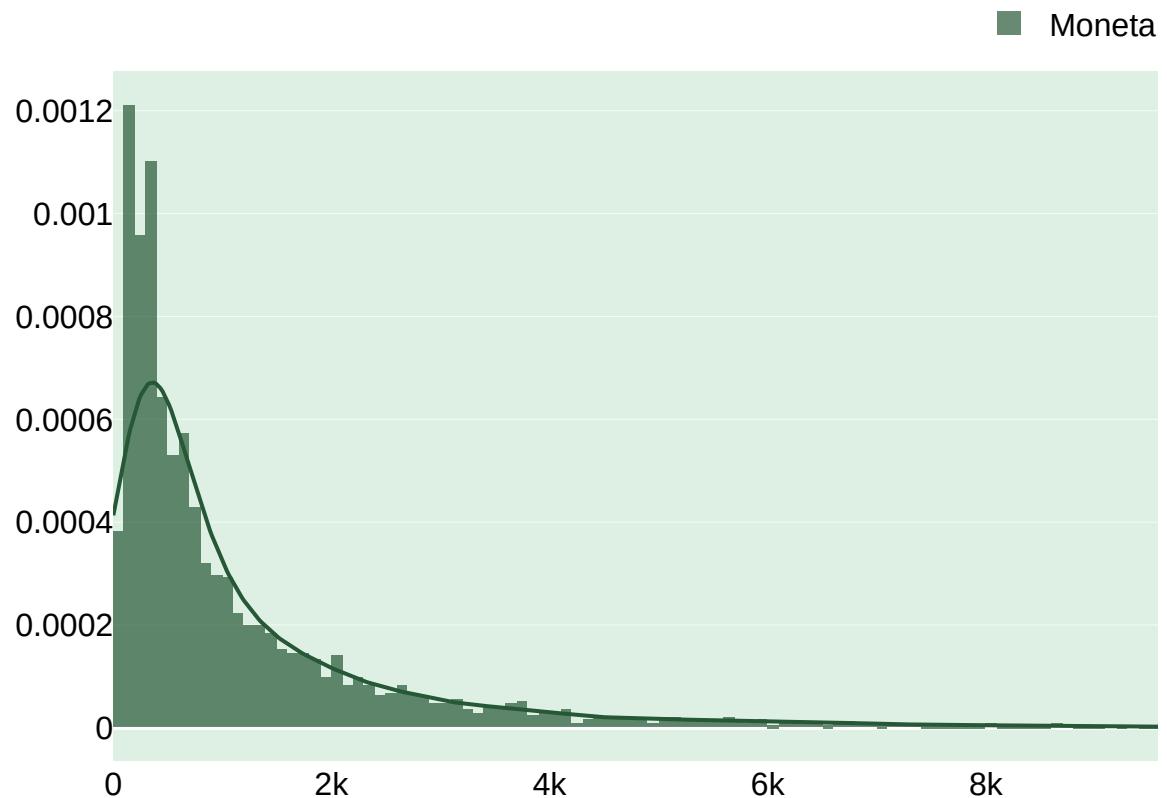
در مورد آخرین تراکنش ها اما این رشد با شبیه کمتری اتفاق افتاده به این صورت که کمتر از ۱۸۰ روز از آخرین تراکنش حدود ۸۰ درصد مشتری ها میگذرد.

In [17]:

```
x = RFM.query('Monetary < 10000')['Monetary']
hist_data = [x]
group_labels = ['Monetary'] # name of the dataset

fig = ff.create_distplot(hist_data, group_labels, bin_size=100, show_rug=False)
colors = ['#265736']
fig.update_layout(width = 700, title="Distribution Plot", font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
    orientation="h",
    yanchor="bottom",
    y=1.02,
    xanchor="right",
    x=1
))
fig.show()
```

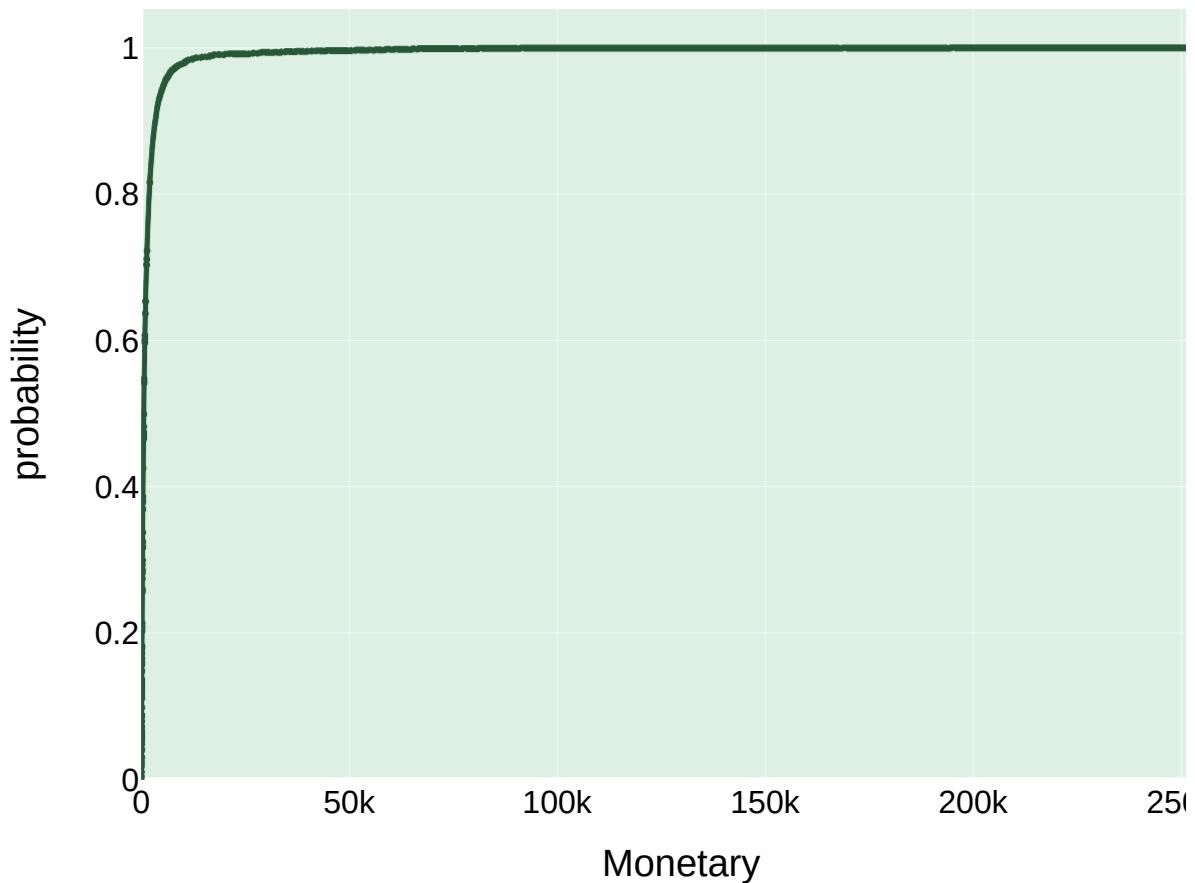
Distribution Plot



در اینجا نیز توزیع در مبالغ کمتر بیشتر است

```
In [18]: df = RFM
fig = px.ecdf(df, x="Monetary",)
fig.update_traces(line_color='#265736', line_width=3.5)
fig.update_layout(width = 700,title="Cumulative Distribution Plots",font
                  family="Arial",
                  size=16,
                  color='#000000'
                 ), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
fig.show()
```

Cumulative Distribution Plots



نمودار تجمعی بالا نشان میدهد که حدود ۹۰ درصد از مشتری های هزینه پرداختی

3.5k

و کمتر داشته اند.

In [19]: #Descriptive Statistics (Frequency)
RFM.Frequency.describe()

Out[19]: count 3921.000000
mean 90.371079
std 217.796155
min 1.000000
25% 17.000000
50% 41.000000
75% 99.000000
max 7847.000000
Name: Frequency, dtype: float64

In [20]: #Descriptive Statistics (Monetary)
RFM.Monetary.describe()

Out[20]: count 3921.000000
mean 1863.910113
std 7481.922217
min 0.000000
25% 300.040000
50% 651.820000
75% 1575.890000
max 259657.300000
Name: Monetary, dtype: float64

در این بخش برای دسته بندی مشتری ها از دو روش استفاده میکنیم: روش اول روش چارک بندی داده ها: به این صورت که مقادیر دیناتی هر ویژگی را برای یک چهارم، نیمه و سه چهارم بدست میآوریم و بر اساس این لیمیت های بازه بقیه داده ها رو در چهار دسته قرار میدهم به صورتی که:

کوچکترین مقدار ویژگی سر بازه و مقدار آن در یک چهارم انتهای بازه ای اول باشه.

.... مقدار یک چهارم سر بازه و مقدار نیم انتهای بازه ای دوم باشه و

In [21]: #Split into four segments using quantiles
quantiles = RFM[['Recency', 'Frequency', 'Monetary']].quantile(q=[0.25, 0.5, 0.75])
quantiles = quantiles.to_dict()

RFM['R_score']=RFM['Recency'].apply(lambda x: 1 if x<quantiles['Recency'][0] else 2 if x<quantiles['Recency'][1] else 3 if x<quantiles['Recency'][2] else 4)
RFM['F_score']=RFM['Frequency'].apply(lambda x: 1 if x<quantiles['Frequency'][0] else 2 if x<quantiles['Frequency'][1] else 3 if x<quantiles['Frequency'][2] else 4)
RFM['M_score']=RFM['Monetary'].apply(lambda x: 1 if x<quantiles['Monetary'][0] else 2 if x<quantiles['Monetary'][1] else 3 if x<quantiles['Monetary'][2] else 4)
RFM['RFM_combined'] = RFM.R_score.map(str) + RFM.F_score.map(str) + RFM.M_score.map(str)
Calculate and Add RFM value column showing total sum of RFM_score values
RFM['RFM_score'] = RFM[['R_score', 'F_score', 'M_score']].sum(axis = 1)
You can change the contribution of R/F/M_score on the total RFM_score
sr, sf, sm = 1, 1, 1
RFM['RFM_weighted']=RFM.apply(lambda x: sr*x['R_score'] + sf*x['F_score'] + sm*x['M_score'])

RFM.sort_values(by=['Recency', 'Frequency', 'Monetary'], ascending=False)

در دو حالت دسته بندی انجام دادیم : ۱) مجموع هر سه نمره را برای هر مشتری در نظر گرفتیم

به نمره هر ویژگی ضریبی اضافه کردیم که بتوان با وزن دادن به ضرایب تاثیر هرکدام از ویژگی ها را (۲) بیشتر یا کمتر کرد

In [22]: *#Assign Level to each customer*

```
Levels = ['Great', 'Good', 'Bad', 'Ugly']
Score_cuts = pd.qcut(RFM.RFM_score, q = 4, labels = Levels)
RFM['RFM_level'] = Score_cuts.values
RFM.reset_index()
```

Out[22]:

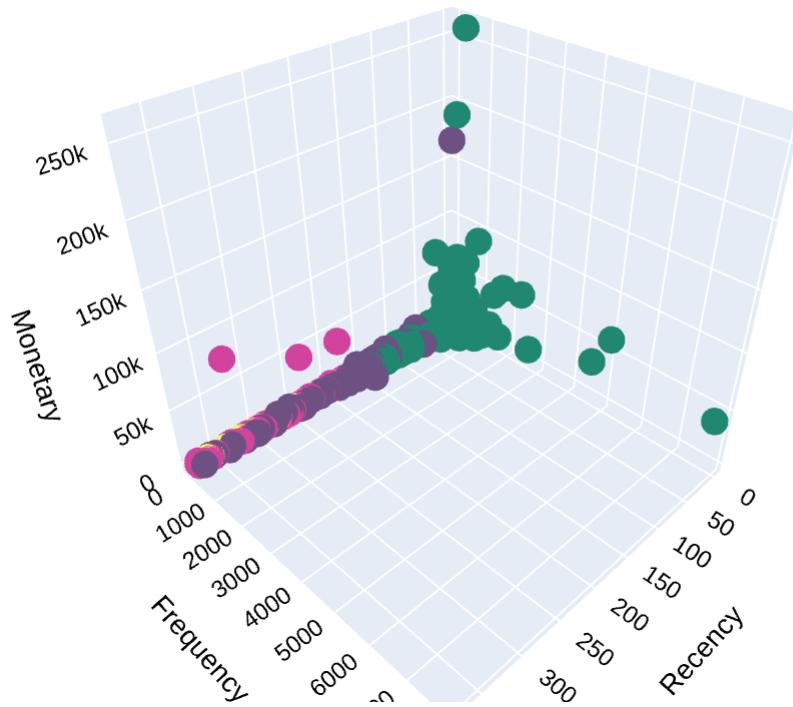
	index	CustomerID	Recency	Frequency	Monetary	R_score	F_score	M_score	RFM_cor
0	0	12346.0	325	1	77183.60	4	4	1	
1	1	12747.0	2	103	4196.01	1	1	1	
2	2	12748.0	0	4596	33719.73	1	1	1	
3	3	12749.0	3	199	4090.88	1	1	1	
4	4	12820.0	3	59	942.34	1	2	2	
...
3916	3916	18280.0	277	10	180.60	4	4	4	
3917	3917	18281.0	180	7	80.82	4	4	4	
3918	3918	18282.0	7	12	178.05	1	4	4	
3919	3919	18283.0	3	756	2094.88	1	1	1	
3920	3920	18287.0	42	70	1837.28	2	2	1	

3921 rows × 12 columns

بر اساس مجموع نمره های ویژگی ها، به روش قبل یه دسته بندی جدید به وجود می آوریم تا در نهایت بتوان کل مشتری ها رو با فاکتور یکسانی به دسته های مختلف تقسیم کرد

```
In [24]: df = RFM
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color=df['Segment'])
fig.update_layout(width = 700,title="3D scatter Plots",font=dict(
    family="Arial",
    size=12,
    color="#000000"
), paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor="#ddf0e3")
)
fig.show()
```

3D scatter Plots



در پلات بالا نمایش سه بعدی از سه ویژگی همه‌ی مشتری‌ها را مشاهده می‌کنیم. رنگ سبزآبی نمایانگر بهترین مشتری‌ها طبق دسته بندی بالا می‌باشد و رنگ زرد مشتری‌هایی را نشان میدهد که مدنظر ما (:). نیستند. (نام‌گذاری صرفا برای تلطیف فضاست)

برای نمایش بهتر محورها این ویژگی‌ها به صورت پلات‌های دو بعدی در زیر آمده است.

```
In [25]: #Recency Vs Frequency
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Recency'],
        y=graph.query("RFM_level == 'Ugly'")['Frequency'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Recency'],
        y=graph.query("RFM_level == 'Bad'")['Frequency'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Recency'],
        y=graph.query("RFM_level == 'Good'")['Frequency'],
        mode='markers',
        name='Good',
        marker= dict(size= 11,
                    line= dict(width=1),
                    color= '#6c5182',
                    opacity= 0.9
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Great'")['Recency'],
        y=graph.query("RFM_level == 'Great'")['Frequency'],
        mode='markers',
        name='Great',
        marker= dict(size= 13,
                    line= dict(width=1),
                    color= '#208771',
                    opacity= 0.9
                )
    ),
]
plot_layout = gobj.Layout(
    yaxis= {'title': "Frequency"},
    xaxis= {'title': "Recency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
```

```
fig.update_layout(width = 700,title="Frequency vs. Recency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Frequency Vs Monetary
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Frequency'],
        y=graph.query("RFM_level == 'Ugly'")['Monetary'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Frequency'],
        y=graph.query("RFM_level == 'Bad'")['Monetary'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Frequency'],
        y=graph.query("RFM_level == 'Good'")['Monetary'],
        mode='markers',
        name='Good',
        marker= dict(size= 11,
                    line= dict(width=1),
                    color= '#6c5182',
                    opacity= 0.9
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Great'")['Frequency'],
        y=graph.query("RFM_level == 'Great'")['Monetary'],
        mode='markers',
        name='Great',
        marker= dict(size= 13,
                    line= dict(width=1),
                    color= '#2ca02c',
                    opacity= 0.9
                )
    )
]
```

```
marker= dict(size= 13,
             line= dict(width=1),
             color= '#208771',
             opacity= 0.9
           )
        ),
    ]

plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Frequency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Frequency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

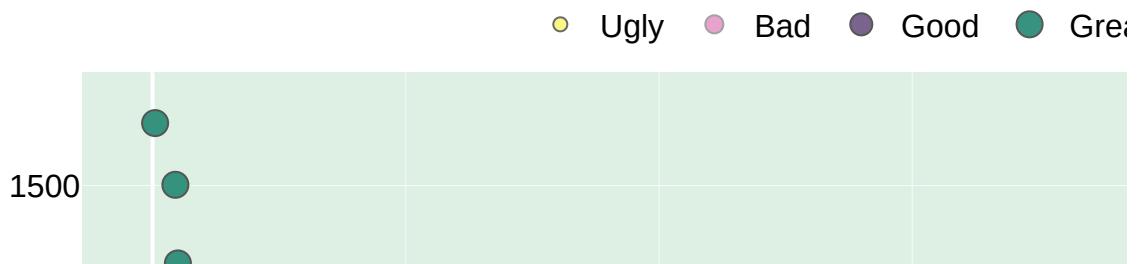
#Recency Vs Monetary
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

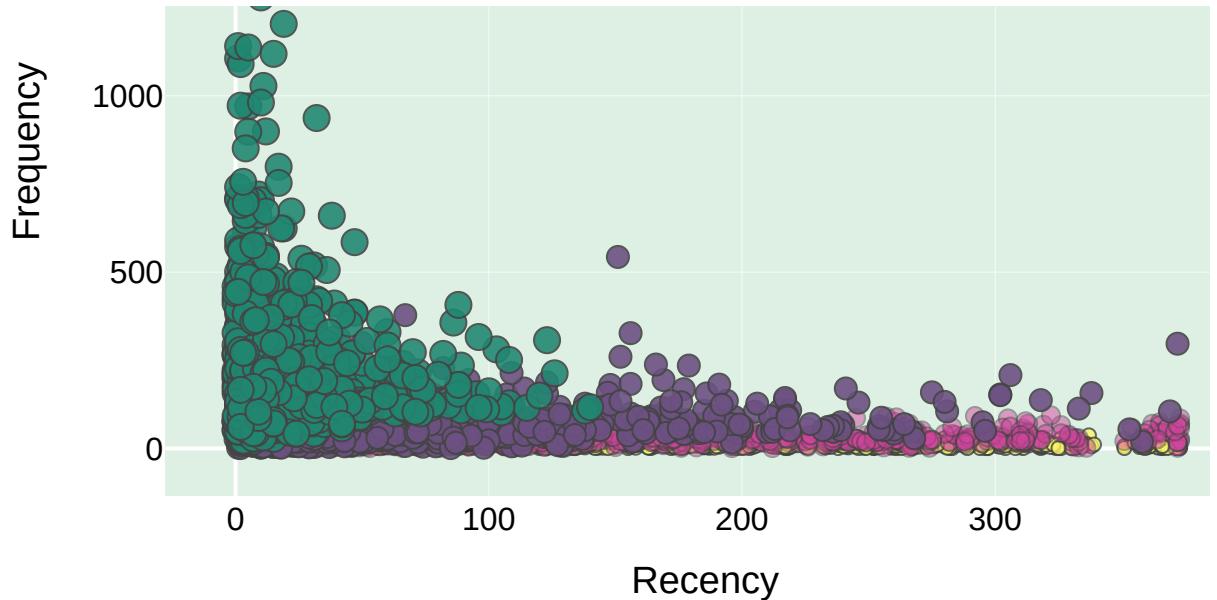
plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Recency'],
        y=graph.query("RFM_level == 'Ugly'")['Monetary'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                  )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Recency'],
        y=graph.query("RFM_level == 'Bad'")['Monetary'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                  )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Recency'],

```

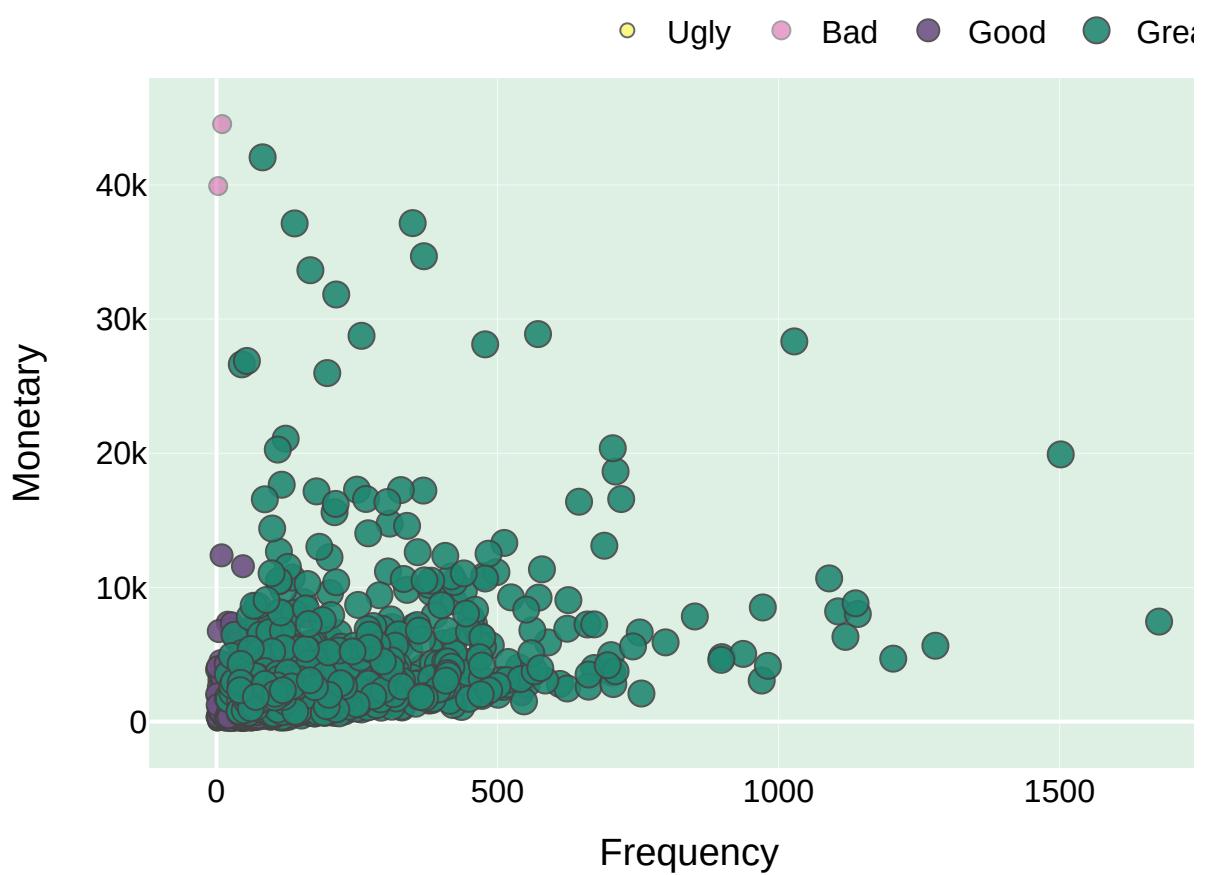
```
y=graph.query("RFM_level == 'Good'")['Monetary'],
mode='markers',
name='Good',
marker= dict(size= 11,
             line= dict(width=1),
             color= '#6c5182',
             opacity= 0.9
           )
),
gobj.Scatter(
    x=graph.query("RFM_level == 'Great'")['Recency'],
    y=graph.query("RFM_level == 'Great'")['Monetary'],
    mode='markers',
    name='Great',
    marker= dict(size= 13,
                 line= dict(width=1),
                 color= '#208771',
                 opacity= 0.9
               )
),
]
plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Recency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Recency", font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
))
)
po.iplot(fig)
```

Frequency vs. Recency

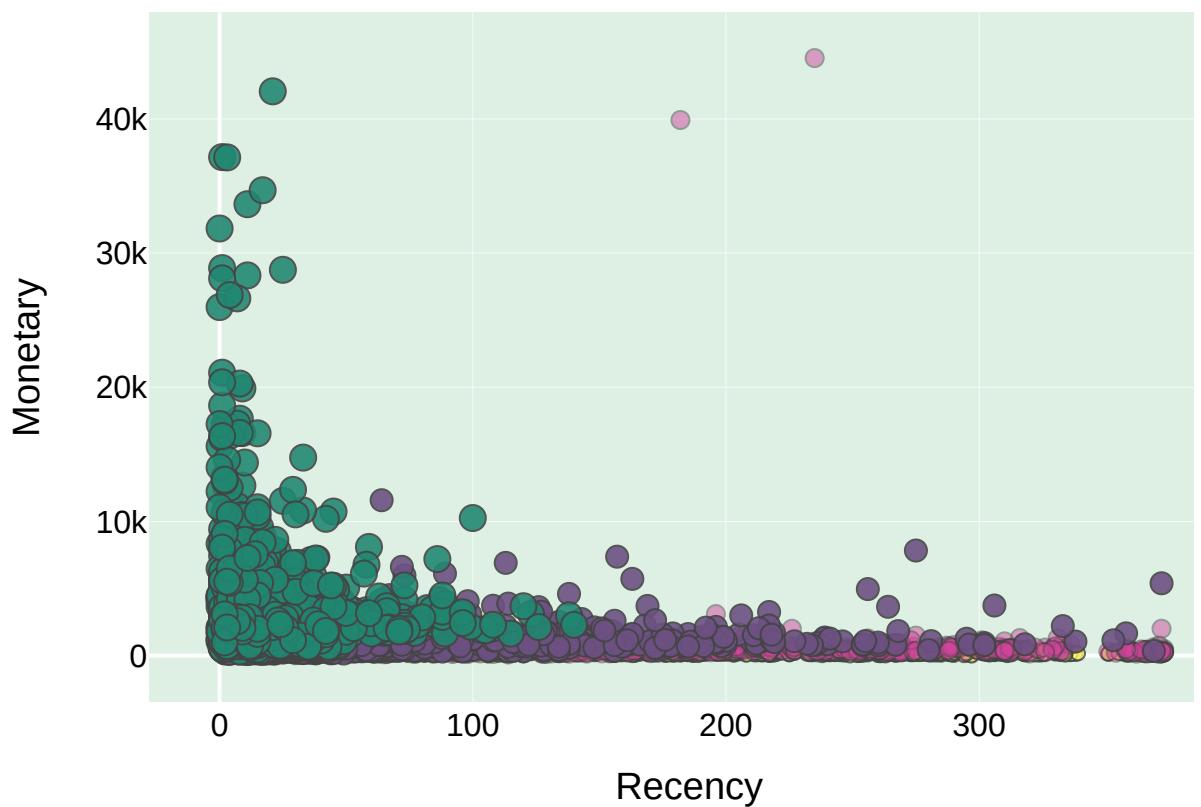




Monetary vs. Frequency



Monetary vs. Recency



را مشاهده میکنیم RFM در گراف های بالا نمایش دو بعدی و دو به دوی ویژگی های متده است.

در نمودار اول آن دسته از مشتریانی که مدت زمان کوتاه تری از آخرین تراکنششان گذشته و از آن طرف بسامد بیشتری در تراکنش ها داشتند را میتوان مشتری های وفاداری دانست. و احتمالا مشتریانی با آخرین تراکنش نزدیک ولی بسامد کم را بتوان مشتری های جدید اضافه شده به مجموعه در نظر گرفت(نیاز به بررسی بیشتر دارد و احتمال هست)

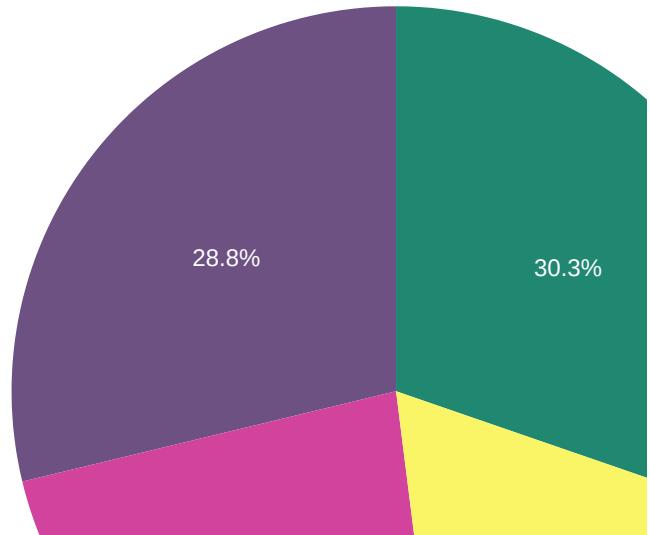
نمودار دوم اما جالبتر به نظر میاید با کمی زوم روی نواحی ابتدایی همانطور که انتظار داریم مشتری هایی که هم بسامد کم و هم هزینه ای کمی پرداخت کردن مشتری های مورد نظر ما نیستند. اما مشتریانی که از ما نمره عالی رو هم دریافت کردند در بخش هایی از آنواحی دیده میشوند که به نظر میاد بیشتر نمره ای خود را از آخرین تراکنش بدست آورده اند. (تئوری مشتری های جدید)

در نمودار سوم هم باز به خوبی دسته ای "عالی" توجیه میشود. مشتریانی با پرداخت بالا و زمان های نزدیک.

بدیهی است ایده آل مشتری ها آنهایی است که تراکنش های زیاد و در زمان های نزدیک و پرداخت های بالا(متنااسب با بسامد) دارند

In [26]: RFM=df

```
fig = px.pie(df, values='CustomerID', names='RFM_level',color_discrete_s  
fig.show()
```



روش دوم) حالت دیگری در نظر گرفتیم که درصد مشتری هایی که بر جسب عالی دریافت کردند رو کم کنیم. بازه بندی را سختگیرانه تر کردیم و به جای در نظر گرفتن چارک ها بازه ها را چندک ۹۰/۷۰ و چندک ۹۰/۶۰ در نظر گرفتیم

```
In [27]: #Split into four segments using quantiles_III
quantiles_III = RFM[['Recency', 'Frequency', 'Monetary']].quantile(q=[0.45])
quantiles_III = quantiles_III.to_dict()
RFMIII=RFM
RFMIII['R_score']=RFMIII['Recency'].apply(lambda x: 1 if x<quantiles_III['Recency'][0] else 2 if x<quantiles_III['Recency'][1] else 3 if x<quantiles_III['Recency'][2] else 4)
RFMIII['F_score']=RFMIII['Frequency'].apply(lambda x: 1 if x<quantiles_III['Frequency'][0] else 2 if x<quantiles_III['Frequency'][1] else 3 if x<quantiles_III['Frequency'][2] else 4)
RFMIII['M_score']=RFMIII['Monetary'].apply(lambda x: 1 if x<quantiles_III['Monetary'][0] else 2 if x<quantiles_III['Monetary'][1] else 3 if x<quantiles_III['Monetary'][2] else 4)
RFMIII['RFM_combined'] = RFM.R_score.map(str) + RFM.F_score.map(str) + RFM.M_score.map(str)
# Calculate and Add RFM value column showing total sum of RFM_score values
RFMIII['RFM_score'] = RFMIII[['R_score', 'F_score', 'M_score']].sum(axis=1)
# You can change the contribution of R/F/M_score on the total RFM_score
sr, sf, sm = 1, 1, 1
RFMIII['RFM_weighted']=RFM.apply(lambda x: sr*x['R_score'] + sf*x['F_score'] + sm*x['M_score'])

# RFM.sort_values(by=['Recency', 'Frequency', 'Monetary'], ascending=False)

Levels = ['Great', 'Good', 'Bad', 'Ugly']
Score_cuts = pd.qcut(RFMIII.RFM_score, q = 4, labels = Levels)
RFMIII['RFM_level'] = Score_cuts.values
RFMIII.reset_index()
```

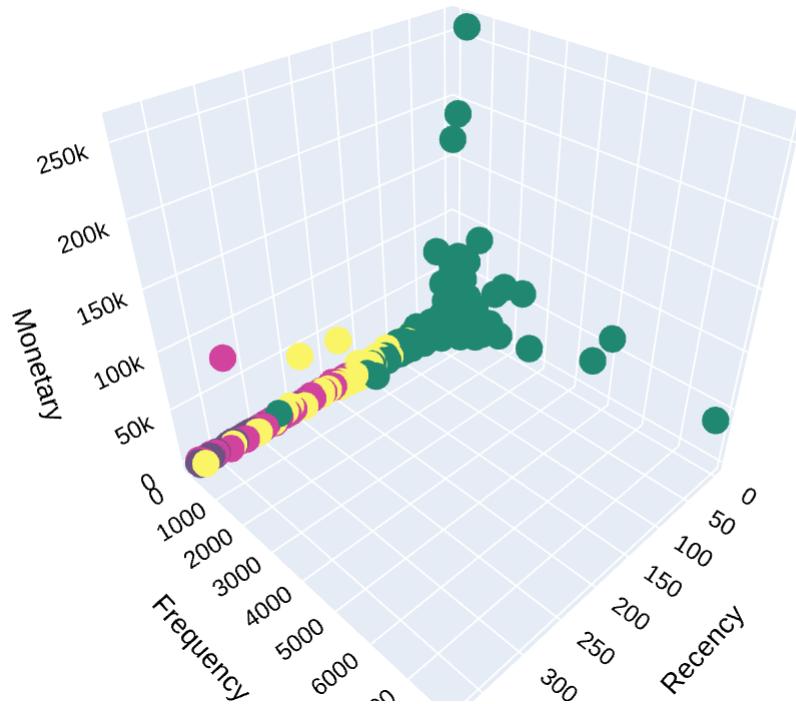
Out[27]:

	index	CustomerID	Recency	Frequency	Monetary	R_score	F_score	M_score	RFM_cor
0	0	12346.0	325	1	77183.60	4	4	1	
1	1	12747.0	2	103	4196.01	1	2	2	
2	2	12748.0	0	4596	33719.73	1	1	1	
3	3	12749.0	3	199	4090.88	1	2	2	
4	4	12820.0	3	59	942.34	1	3	3	
...
3916	3916	18280.0	277	10	180.60	3	4	4	
3917	3917	18281.0	180	7	80.82	3	4	4	
3918	3918	18282.0	7	12	178.05	1	4	4	
3919	3919	18283.0	3	756	2094.88	1	1	2	
3920	3920	18287.0	42	70	1837.28	2	3	2	

3921 rows × 12 columns

```
In [28]: df = RFMIII  
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color  
fig.update_layout(width = 700,title="3D scatter Plots",font=dict(  
    family="Arial",  
    size=12,  
    color='#000000'  
    ),  
    paper_bgcolor='rgba(0,0,0,0)',  
    plot_bgcolor="#ddf0e3"  
)  
fig.show()  
  
# روش دو مرحله
```

3D scatter Plots



```
In [29]: #Recency Vs Frequency
graph = RFMIII.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Recency'],
        y=graph.query("RFM_level == 'Ugly'")['Frequency'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                     line= dict(width=1),
                     color= '#faf566',
                     opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Recency'],
        y=graph.query("RFM_level == 'Bad'")['Frequency'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                     line= dict(width=1),
                     color= '#d1439d',
                     opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Recency'],
        y=graph.query("RFM_level == 'Good'")['Frequency'],
        mode='markers',
        name='Good',
        marker= dict(size= 11,
                     line= dict(width=1),
                     color= '#6c5182',
                     opacity= 0.9
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Great'")['Recency'],
        y=graph.query("RFM_level == 'Great'")['Frequency'],
        mode='markers',
        name='Great',
        marker= dict(size= 13,
                     line= dict(width=1),
                     color= '#208771',
                     opacity= 0.9
                )
    ),
]
plot_layout = gobj.Layout(
    yaxis= {'title': "Frequency"},
    xaxis= {'title': "Recency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
```

```
fig.update_layout(width = 700,title="Frequency vs. Recency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Frequency Vs Monetary
graph = RFMIII.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Frequency'],
        y=graph.query("RFM_level == 'Ugly'")['Monetary'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Frequency'],
        y=graph.query("RFM_level == 'Bad'")['Monetary'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Frequency'],
        y=graph.query("RFM_level == 'Good'")['Monetary'],
        mode='markers',
        name='Good',
        marker= dict(size= 11,
                    line= dict(width=1),
                    color= '#6c5182',
                    opacity= 0.9
                )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Great'")['Frequency'],
        y=graph.query("RFM_level == 'Great'")['Monetary'],
        mode='markers',
        name='Great',
        marker= dict(size= 13,
                    line= dict(width=1),
                    color= '#2ca02c',
                    opacity= 0.9
                )
    )
]
```

```
marker= dict(size= 13,
             line= dict(width=1),
             color= '#208771',
             opacity= 0.9
           )
        ),
    ]

plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Frequency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Frequency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

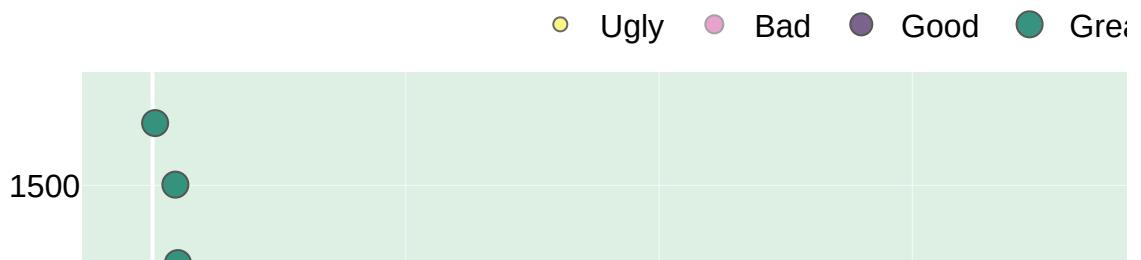
#Recency Vs Monetary
graph = RFMIII.query("Monetary < 50000 and Frequency < 2000")

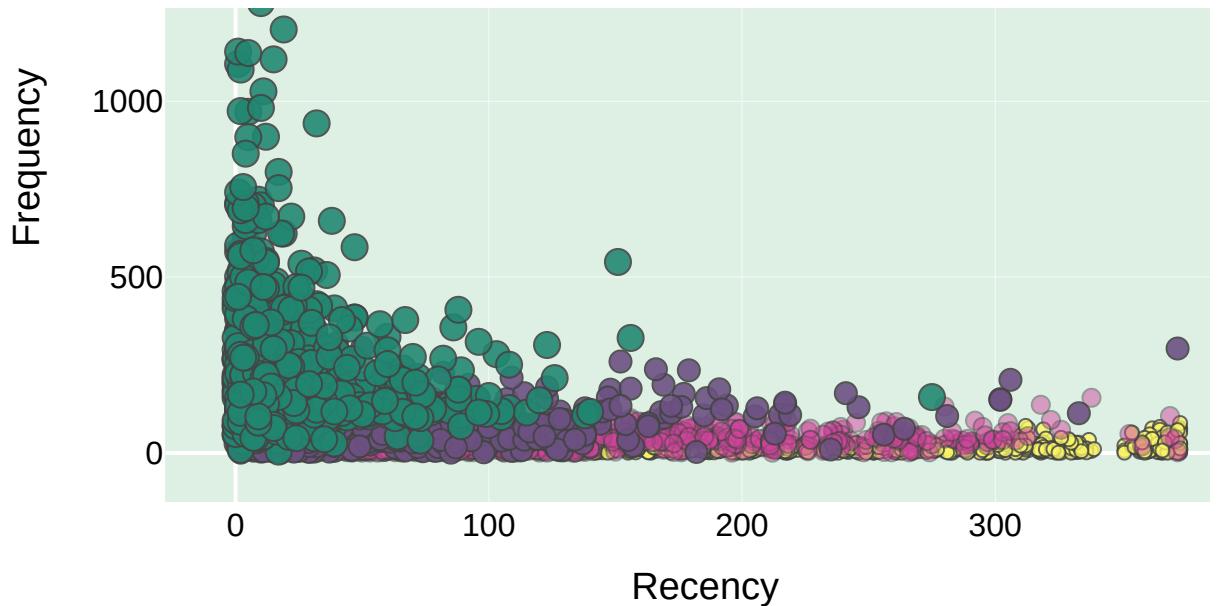
plot_data = [
    gobj.Scatter(
        x=graph.query("RFM_level == 'Ugly'")['Recency'],
        y=graph.query("RFM_level == 'Ugly'")['Monetary'],
        mode='markers',
        name='Ugly',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                  )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Bad'")['Recency'],
        y=graph.query("RFM_level == 'Bad'")['Monetary'],
        mode='markers',
        name='Bad',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                  )
    ),
    gobj.Scatter(
        x=graph.query("RFM_level == 'Good'")['Recency'],

```

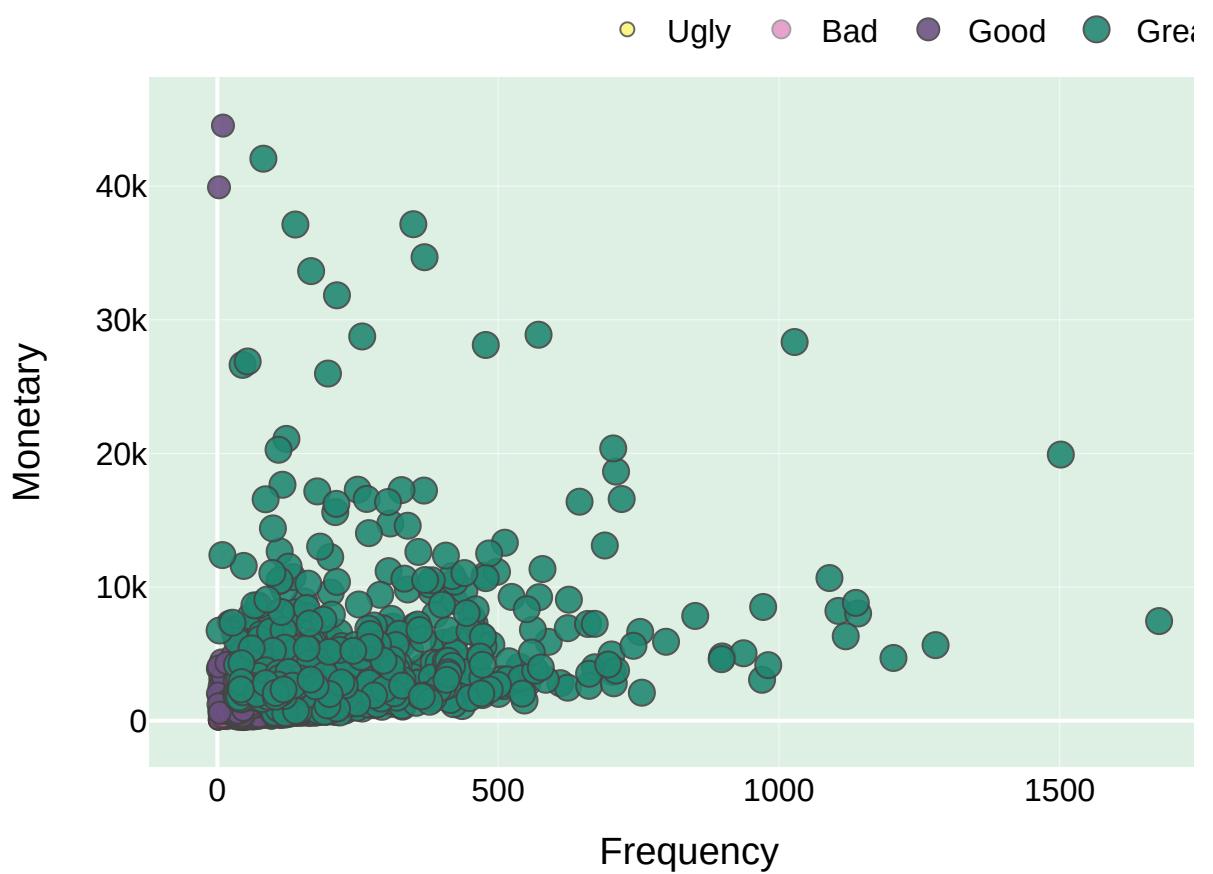
```
y=graph.query("RFM_level == 'Good'")['Monetary'],
mode='markers',
name='Good',
marker= dict(size= 11,
             line= dict(width=1),
             color= '#6c5182',
             opacity= 0.9
         )
),
gobj.Scatter(
    x=graph.query("RFM_level == 'Great'")['Recency'],
    y=graph.query("RFM_level == 'Great'")['Monetary'],
    mode='markers',
    name='Great',
    marker= dict(size= 13,
                 line= dict(width=1),
                 color= '#208771',
                 opacity= 0.9
             )
),
]
plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Recency"},
    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Recency", font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
))
)
po.iplot(fig)
```

Frequency vs. Recency



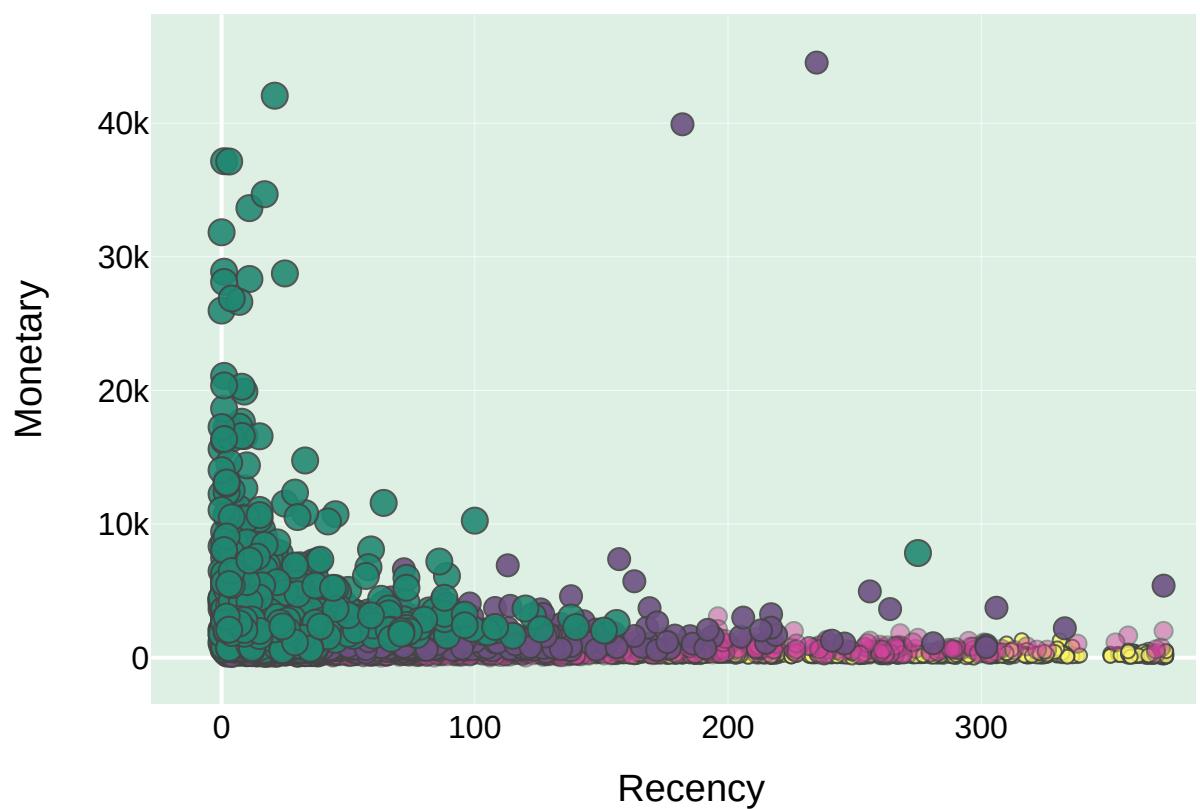


Monetary vs. Frequency

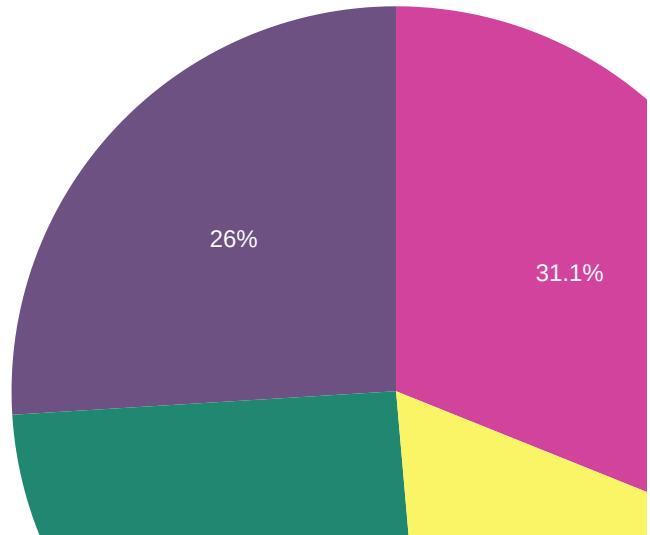


Monetary vs. Recency





```
In [30]: RFMIII=df  
fig = px.pie(df, values='CustomerID', names='RFM_level',color_discrete_s  
fig.show()
```



همانطور که در چارت بالا هم مشخص است این تغییر در مرزها باعث کاهش برچسب های "عالی" شد

روش دوم

در این روش از کتابخانه ای به نام سای کیت استفاده میکنیم

این روش دیتای ویژگی های مختلف را میخواند و سپس آنها را هم مقیاس(هم اسکیل) میکند و سپس با روش های آماری و در نظر گرفتن جهت ماقریزم پراکندگی و جهت عمود بر آن دسته بندی میکند . این کتابخانه از روش های متفاوتی برای این کار استفاده میکند که هر کدام مزایای مربوط به خود را دارند. در اینجا من از سه روش استفاده کردم.

معروف است. این روش به طور خود کار دیتا ها رو هم مقیاس میکند "K-means" روش اول: این روش به و سپس با تبدیل یک ماتریس به دو تا ماتریس که در هم ضرب میشوند یک ماتریس ضراایب بدست می آورد و ماتریسی دیگر از مقادیر ویژگی ها و در نهایت با تبدیل این ماتریس ها داده ها را بر اساس ویژگی هایشان دسته بندی میکند.

ابتدا برای کمک به بدست آوردن داده های هم مقیاس برای کمتر کردن فاصله ها از همه های مقادیر لگاریتم میگیریم و اقدامات بعدی را بر روی داده های لگاریتمی انجام میدهیم.

```
In [31]: #Handle negative and zero values so as to handle infinite numbers during
def handle_neg_n_zero(num):
    if num <= 0:
        return 1
    else:
        return num
#Apply handle_neg_n_zero function to Recency and Monetary columns
RFM['Recency'] = [handle_neg_n_zero(x) for x in RFM.Recency]
RFM['Monetary'] = [handle_neg_n_zero(x) for x in RFM.Monetary]

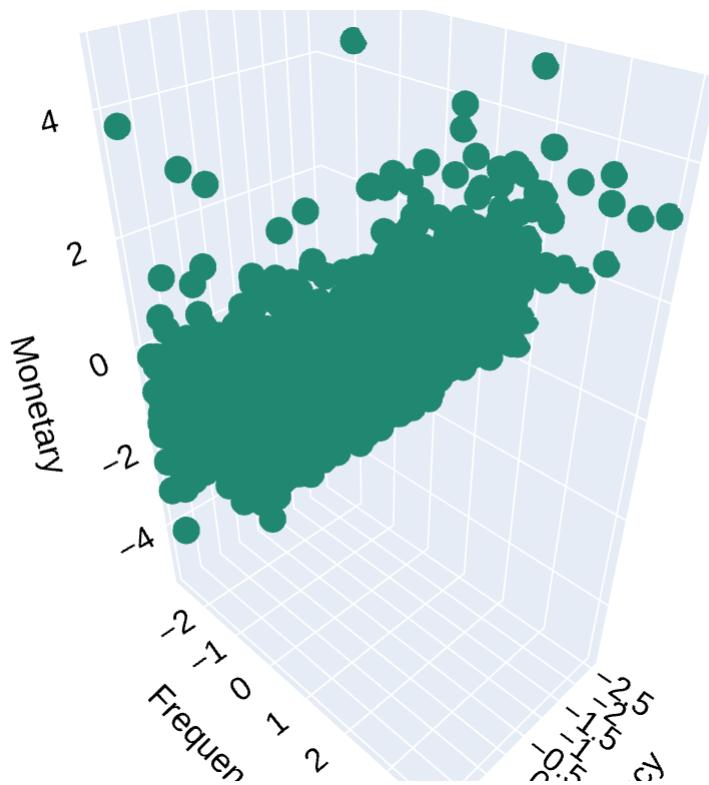
#Perform Log transformation to bring data into normal or near normal distribution
Log_Data = RFM[['Recency', 'Frequency', 'Monetary']].apply(np.log, axis=0)
RFM_new=RFM[['Recency', 'Frequency', 'Monetary']]

#Bring the data on same scale
scaleobj = StandardScaler()
Scaled_Data = scaleobj.fit_transform(Log_Data)

#Transform it back to dataframe
Scaled_Data = pd.DataFrame(Scaled_Data, index = RFM.index, columns = Log_Data.columns)
```

```
In [32]: import plotly.express as px
df = Scaled_Data
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color=df['Cluster'])
fig.update_layout(width = 700,title="3D Scatter plot of unclassified data",
                    family="Arial",
                    size=14,
                    color="#000000"
                ),    paper_bgcolor='rgba(0,0,0,0)',
                plot_bgcolor='#ddf0e3',
                legend=dict(
                    orientation="h",
                    yanchor="bottom",
                    y=1.02,
                    xanchor="right",
                    x=1
                )
)
fig.show()
```

3D Scatter plot of unclassified data



نمودار بالا تجمعی از همه داده ها در حالتی که هم مقیاس شده اند نشان میدهد. آنچه مورد انتظار است "K-means" است دسته بندی این داده ها به کمک روش

در ادامه مناسب ترین تعداد گروه برای داده ها ۳ دسته میباشد(با آزمون و خطا هم بدست میآید اما روش آماری دقیقی هم مثل الوبلاط برای تشخیص این تعداد وجود دارد)

```
In [33]: from sklearn.cluster import KMeans
num_cluster = 3
KMean_clust = KMeans(n_clusters= num_cluster, init= 'k-means++', max_iter= 300)
KMean_clust.fit(Scaled_Data)
#Find the clusters for the observation given in the dataset
RFM['Cluster'] = KMean_clust.labels_
RFM
```

Out[33]:

	CustomerID	Recency	Frequency	Monetary	R_score	F_score	M_score	RFM_combined
0	12346.0	325	1	77183.60	4	4	1	441
1	12747.0	2	103	4196.01	1	2	2	122
2	12748.0	1	4596	33719.73	1	1	1	111
3	12749.0	3	199	4090.88	1	2	2	122
4	12820.0	3	59	942.34	1	3	3	133
...
3916	18280.0	277	10	180.60	3	4	4	344
3917	18281.0	180	7	80.82	3	4	4	344
3918	18282.0	7	12	178.05	1	4	4	144
3919	18283.0	3	756	2094.88	1	1	2	112
3920	18287.0	42	70	1837.28	2	3	2	232

3921 rows × 12 columns

```
In [34]: RFM.groupby(['RFM_level','Cluster']).count()
```

Out[34]:

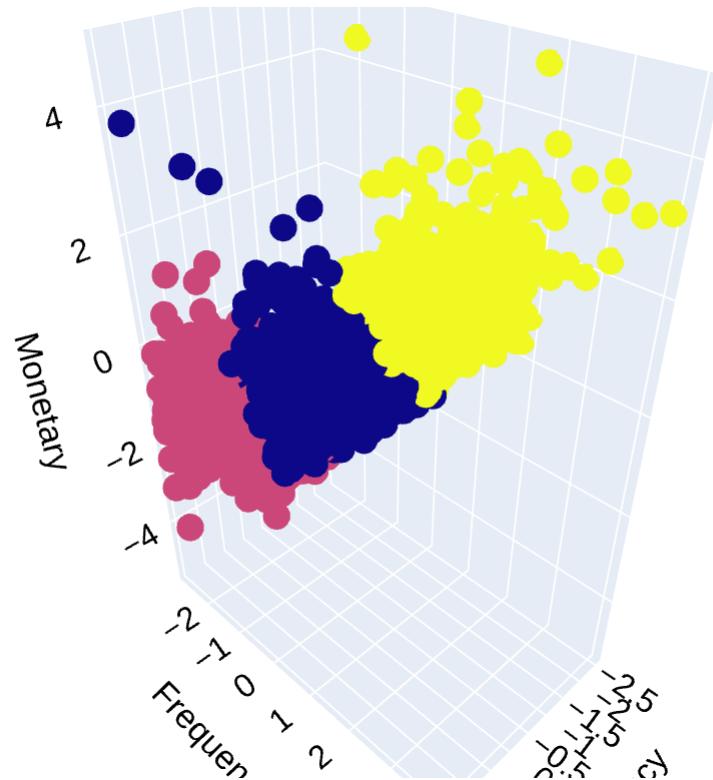
RFM_level	Cluster	CustomerID	Recency	Frequency	Monetary	R_score	F_score	M_score	R
Great	0	268	268	268	268	268	268	268	268
	1	0	0	0	0	0	0	0	0
	2	731	731	731	731	731	731	731	731
Good	0	891	891	891	891	891	891	891	891
	1	20	20	20	20	20	20	20	20
	2	107	107	107	107	107	107	107	107
Bad	0	486	486	486	486	486	486	486	486
	1	727	727	727	727	727	727	727	727
	2	5	5	5	5	5	5	5	5
Ugly	0	8	8	8	8	8	8	8	8
	1	678	678	678	678	678	678	678	678
	2	0	0	0	0	0	0	0	0



```
In [35]: Scaled_Data['Cluster']=RFM['Cluster']
```

```
import plotly.express as px
df = Scaled_Data
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color=df['Cluster'])
fig.update_layout(width = 700,title="3D Scatter plot of unclassified data",
family="Arial",
size=14,
color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3',
legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
fig.show()
```

3D Scatter plot of unclassified data



همانطور که در پلاٹ سه بعدی بالا قابل مشاهده است همه دیتای هم مقیاس شده مابه سه دسته تقسیم شده اند و به هر کدام لبیل ای مربوط به آن دسته اختصاص داده شد که در نمودار با رنگ این لبیل

ها نشان داده شده است. انتخاب بهترین دسته به نگاه بیزینسی هم وابسته است ممکن است در مواردی تنها مشتریانی که این اواخر تراکنش داده اند مشتریان مهم باشند فارغ از میزان بسامد و هزینه. و به همین شکل برای ویژگی های بعدی.

```
In [36]: #Recency Vs Frequency
graph = Scaled_Data.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Cluster == 2")['Recency'],
        y=graph.query("Cluster == 2")['Frequency'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                     line= dict(width=1),
                     color= '#faf566',
                     opacity= 0.8
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 1")['Recency'],
        y=graph.query("Cluster == 1")['Frequency'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                     line= dict(width=1),
                     color= '#d1439d',
                     opacity= 0.5
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 0")['Recency'],
        y=graph.query("Cluster == 0")['Frequency'],
        mode='markers',
        name='1st',
        marker= dict(size= 11,
                     line= dict(width=1),
                     color= '#6c5182',
                     opacity= 0.9
                    )
    ),
    # gobj.Scatter(
    #     x=graph.query("Cluster == 'Great')")['Recency'],
    #     y=graph.query("Cluster == 'Great')")['Frequency'],
    #     mode='markers',
    #     name='Great',
    #     marker= dict(size= 13,
    #                  line= dict(width=1),
    #                  color= '#208771',
    #                  opacity= 0.9
    #                 )
    # ),
]

plot_layout = gobj.Layout(
    yaxis= {'title': "Frequency"},
    xaxis= {'title': "Recency"},
    # title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
```

```
fig.update_layout(width = 700,title="Frequency vs. Recency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Frequency Vs Monetary
graph = Scaled_Data.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Cluster == 2")['Frequency'],
        y=graph.query("Cluster == 2")['Monetary'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 1")['Frequency'],
        y=graph.query("Cluster == 1")['Monetary'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 0")['Frequency'],
        y=graph.query("Cluster == 0")['Monetary'],
        mode='markers',
        name='1st',
        marker= dict(size= 11,
                    line= dict(width=1),
                    color= '#6c5182',
                    opacity= 0.9
                    )
    ),
    #    gobj.Scatter(
    #        x=graph.query("Cluster == 'Great'")['Frequency'],
    #        y=graph.query("Cluster == 'Great'")['Monetary'],
    #        mode='markers',
    #        name='Great',
    #    )
]
```

```
#         marker= dict(size= 13,
#                         line= dict(width=1),
#                         color= '#208771',
#                         opacity= 0.9
#                         )
#                 ),
#             ]
#         )

plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Frequency"},
#        title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Frequency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Recency Vs Monetary
graph = Scaled_Data.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Cluster == 2")['Recency'],
        y=graph.query("Cluster == 2")['Monetary'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                    line= dict(width=1),
                    color= '#faf566',
                    opacity= 0.8
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 1")['Recency'],
        y=graph.query("Cluster == 1")['Monetary'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                    line= dict(width=1),
                    color= '#d1439d',
                    opacity= 0.5
                    )
    ),
    gobj.Scatter(
        x=graph.query("Cluster == 0")['Recency'],
```

```

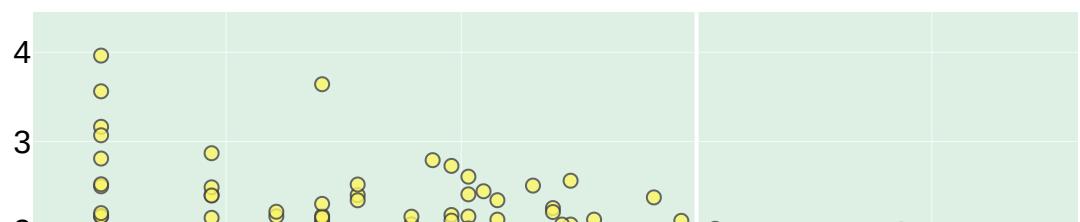
y=graph.query("Cluster == 0")['Monetary'],
mode='markers',
name='1st',
marker= dict(size= 11,
             line= dict(width=1),
             color= '#6c5182',
             opacity= 0.9
         )
),
#      gobj.Scatter(
#          x=graph.query("Cluster == 'Great'")['Recency'],
#          y=graph.query("Cluster == 'Great'")['Monetary'],
#          mode='markers',
#          name='Great',
#          marker= dict(size= 13,
#                      line= dict(width=1),
#                      color= '#208771',
#                      opacity= 0.9
#                  )
#      ),
]
]

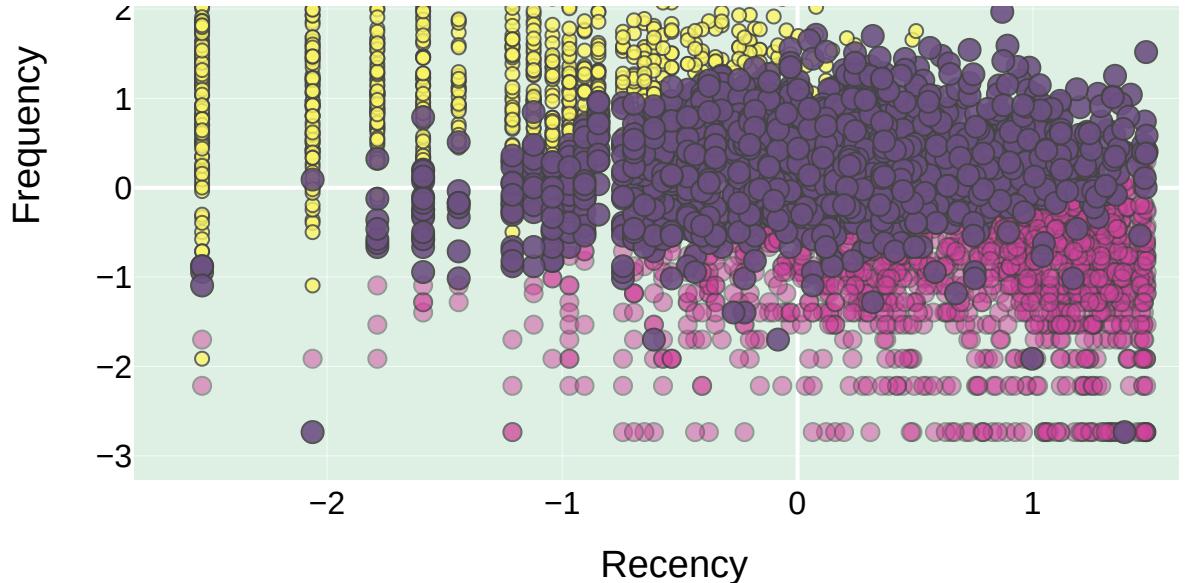
plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Recency"},
#    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Recency", font=dict(
    family="Arial",
    size=16,
    color="#000000"
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
))
)
)
po.iplot(fig)

```

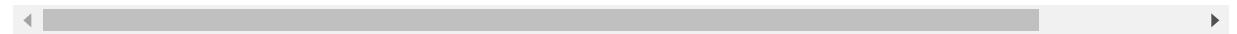
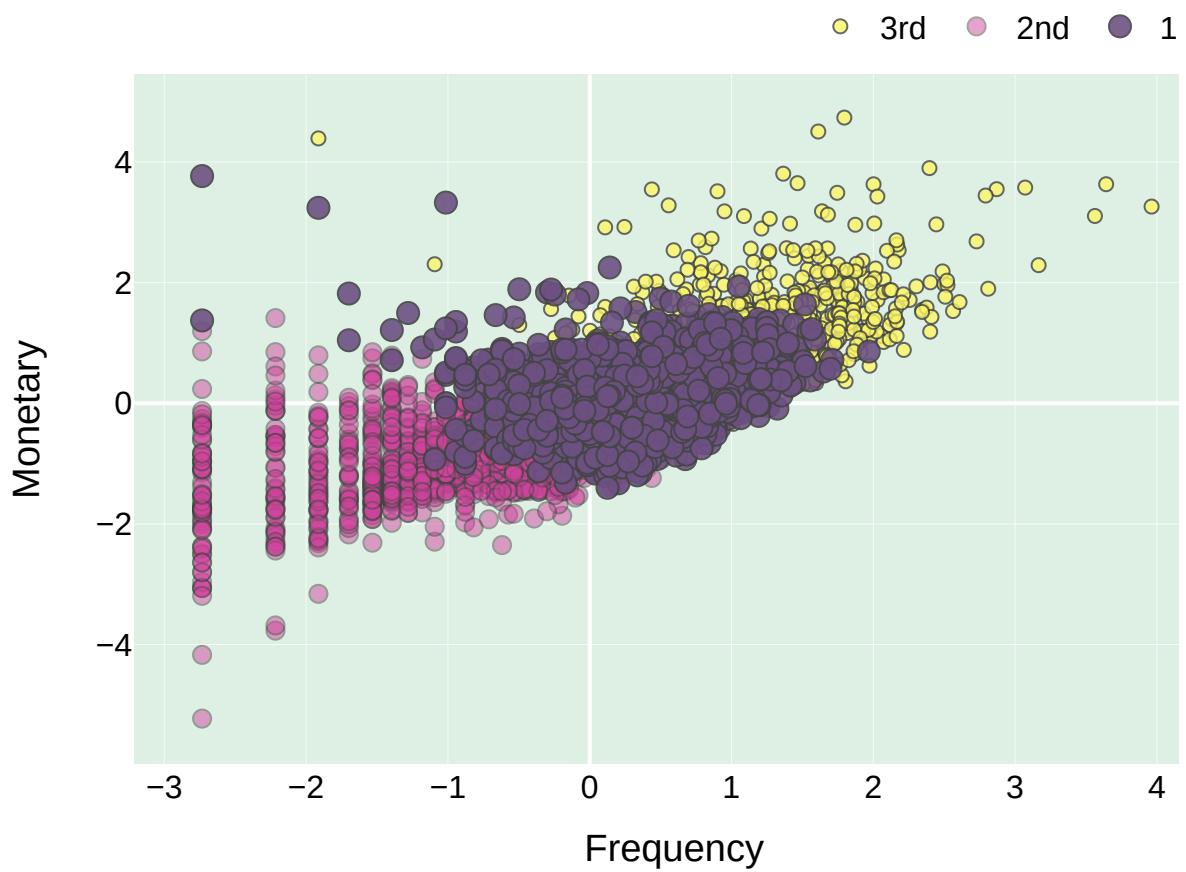
Frequency vs. Recency

● 3rd ● 2nd ● 1

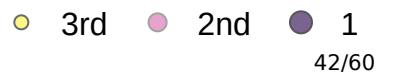


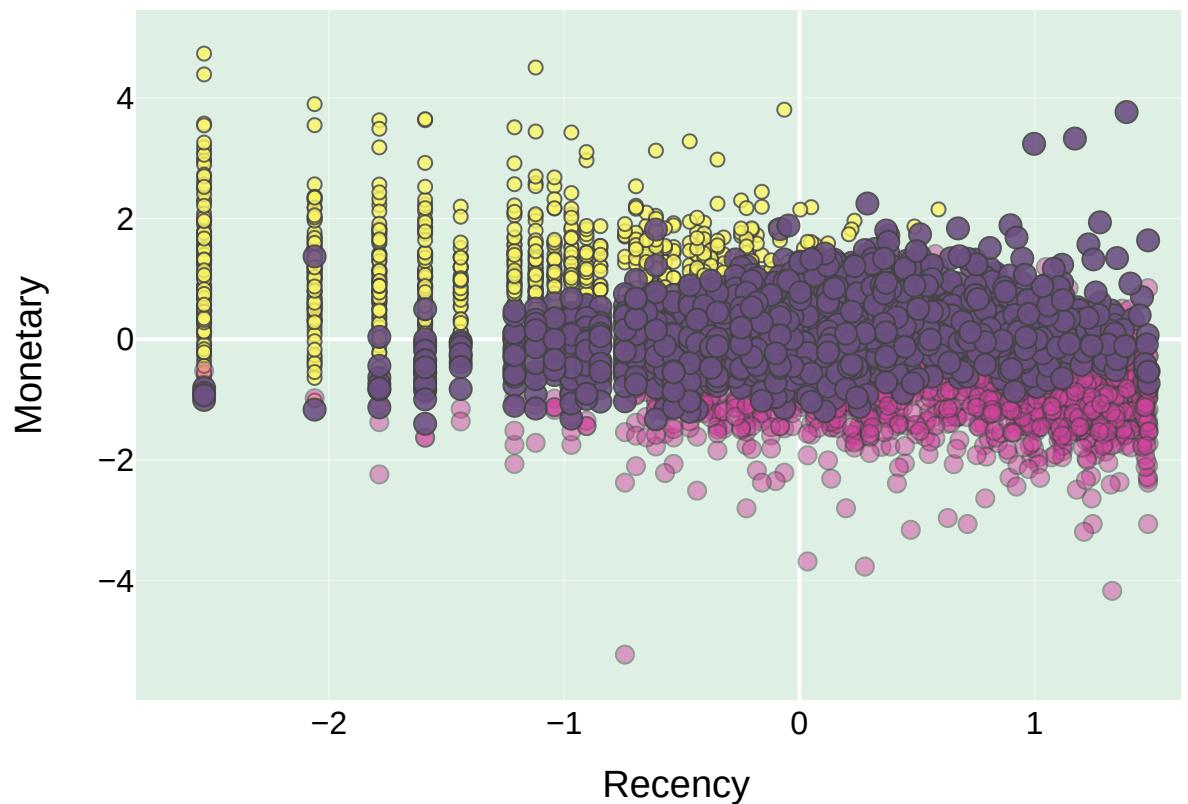


Monetary vs. Frequency



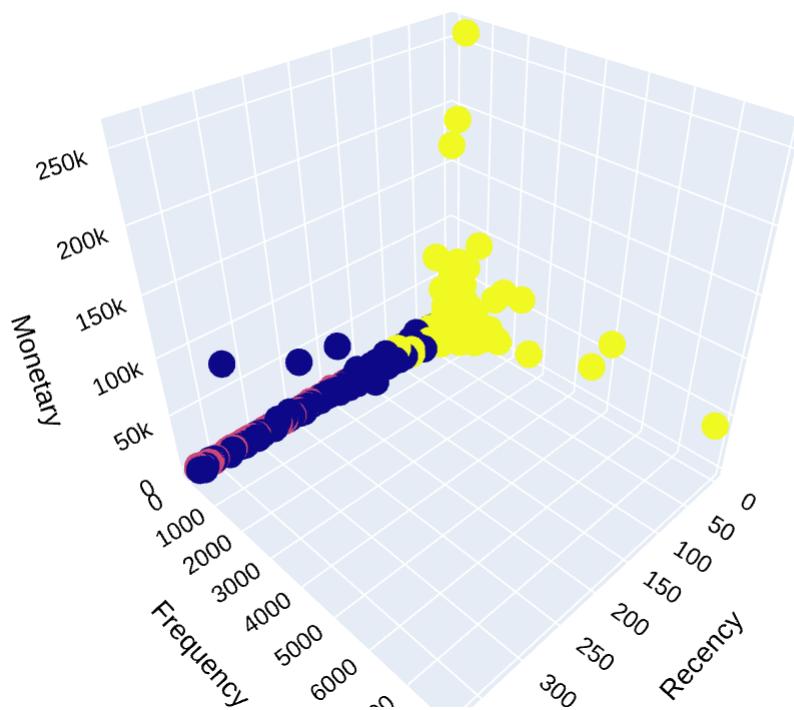
Monetary vs. Recency





```
In [37]: import plotly.express as px
df = RFM
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color
fig.update_layout(width = 700,title="3D scatter Plots",font=dict(
    family="Arial",
    size=12,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor="#ddf0e3"
)
fig.show()
```

3D scatter Plots



نمودار بالا همان دسته بندی بر اساس روش

"k-means"

اما بر روی داده های اصلی را نشان میدهد .

GaussianMixture model

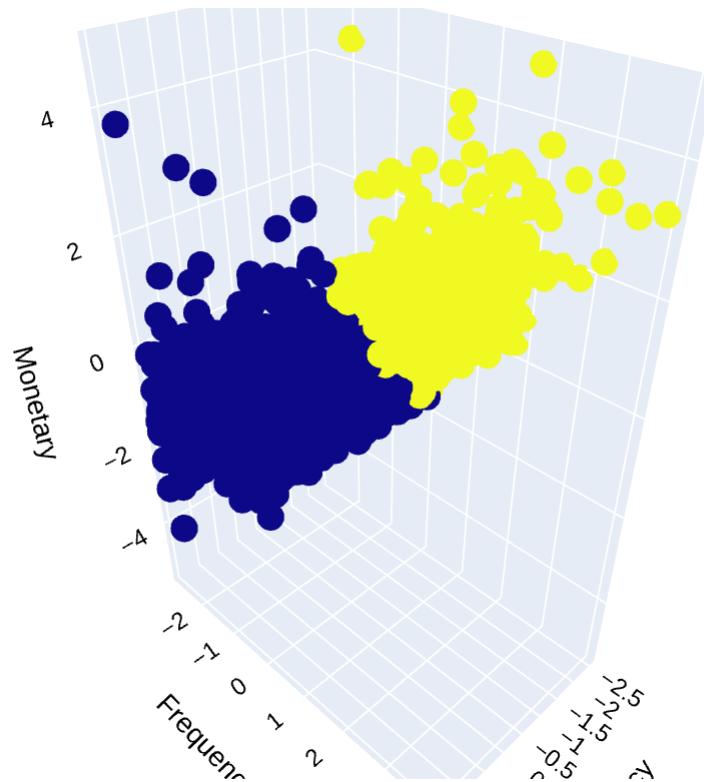
این روش نیز از همان کتابخانه انتخاب شده است و میتوان تعداد دسته را مشخص کرد و خروجی را بنا بر پلن بیزینسی تحلیل کرد.

همچون روش های قبل برای این مدل نیز گراف های سه بعدی و دو بعدی برای سهولت در بررسی رسم شده است.

In [38]: `df=RFM[['Frequency', 'Monetary', 'Recency']]`

```
from sklearn import mixture
gmm = mixture.GaussianMixture(n_components=2).fit(Scaled_Data)
labelsg = gmm.predict(Scaled_Data)
fig = px.scatter_3d(Scaled_Data, x='Recency', y='Frequency', z='Monetary')
fig.update_layout(width = 700,title="3D scatter Plots",font=dict(
    family="Arial",
    size=12,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor='#ddf0e3')
)
fig.show()
```

3D scatter Plots



In [39]: `RFM['Labels_g']=labelsg`

```
In [40]: RFM['Labels_g'].value_counts()
```

```
Out[40]: 0    3078  
1    843  
Name: Labels_g, dtype: int64
```

```
In [41]: import chart_studio as cs
import plotly.offline as po
import plotly.graph_objs as gobj

#Recency Vs Frequency
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Labels_g == 2")['Recency'],
        y=graph.query("Labels_g == 2")['Frequency'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                     line= dict(width=1),
                     color= '#faf566',
                     opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("Labels_g == 1")['Recency'],
        y=graph.query("Labels_g == 1")['Frequency'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                     line= dict(width=1),
                     color= '#d1439d',
                     opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("Labels_g == 0")['Recency'],
        y=graph.query("Labels_g == 0")['Frequency'],
        mode='markers',
        name='1st',
        marker= dict(size= 11,
                     line= dict(width=1),
                     color= '#6c5182',
                     opacity= 0.9
                )
    ),
    #    gobj.Scatter(
    #        x=graph.query("Labels_g == 'Great'")['Recency'],
    #        y=graph.query("Labels_g == 'Great'")['Frequency'],
    #        mode='markers',
    #        name='Great',
    #        marker= dict(size= 13,
    #                     line= dict(width=1),
    #                     color= '#208771',
    #                     opacity= 0.9
    #                )
    #    ),
    ]

plot_layout = gobj.Layout(
    yaxis= {'title': "Frequency"},
```

```
        xaxis= {'title': "Recency"},  
#           title='Segments'  
        )  
fig = gobj.Figure(data=plot_data, layout=plot_layout)  
fig.update_layout(width = 700,title="Frequency vs. Recency",font=dict(  
                           family="Arial",  
                           size=16,  
                           color='#000000'  
                         ), paper_bgcolor='rgba(0,0,0,0)',  
                         plot_bgcolor='#ddf0e3', legend=dict(  
                           orientation="h",  
                           yanchor="bottom",  
                           y=1.02,  
                           xanchor="right",  
                           x=1  
                         )  
                         )  
                         )  
po.iplot(fig)  
  
#Frequency Vs Monetary  
graph = RFM.query("Monetary < 50000 and Frequency < 2000")  
  
plot_data = [  
    gobj.Scatter(  
        x=graph.query("Labels_g == 2")['Frequency'],  
        y=graph.query("Labels_g == 2")['Monetary'],  
        mode='markers',  
        name='3rd',  
        marker= dict(size= 7,  
                     line= dict(width=1),  
                     color= '#faf566',  
                     opacity= 0.8  
                   )  
    ),  
    gobj.Scatter(  
        x=graph.query("Labels_g == 1")['Frequency'],  
        y=graph.query("Labels_g == 1")['Monetary'],  
        mode='markers',  
        name='2nd',  
        marker= dict(size= 9,  
                     line= dict(width=1),  
                     color= '#d1439d',  
                     opacity= 0.5  
                   )  
    ),  
    gobj.Scatter(  
        x=graph.query("Labels_g == 0")['Frequency'],  
        y=graph.query("Labels_g == 0")['Monetary'],  
        mode='markers',  
        name='1st',  
        marker= dict(size= 11,  
                     line= dict(width=1),  
                     color= '#6c5182',  
                     opacity= 0.9  
                   )  
    ),  
#    gobj.Scatter(  
#        x=graph.query("Labels_g == 0")['Frequency'],  
#        y=graph.query("Labels_g == 0")['Monetary'],  
#        mode='markers',  
#        name='0th',  
#        marker= dict(size= 13,  
#                     line= dict(width=1),  
#                     color= '#333399',  
#                     opacity= 0.7  
#                   )  
#    )
```

```

#           x=graph.query("Labels_g == 'Great'")['Frequency'],
#           y=graph.query("Labels_g == 'Great'")['Monetary'],
#           mode='markers',
#           name='Great',
#           marker= dict(size= 13,
#                         line= dict(width=1),
#                         color= '#208771',
#                         opacity= 0.9
#                         )
#       ),
#   ]
#           )

plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Frequency"},
#    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Frequency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Recency Vs Monetary
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Labels_g == 2")['Recency'],
        y=graph.query("Labels_g == 2")['Monetary'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                      line= dict(width=1),
                      color= '#faf566',
                      opacity= 0.8
                      )
    ),
    gobj.Scatter(
        x=graph.query("Labels_g == 1")['Recency'],
        y=graph.query("Labels_g == 1")['Monetary'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                      line= dict(width=1),
                      color= '#d1439d',
                      opacity= 0.5
                      )
]

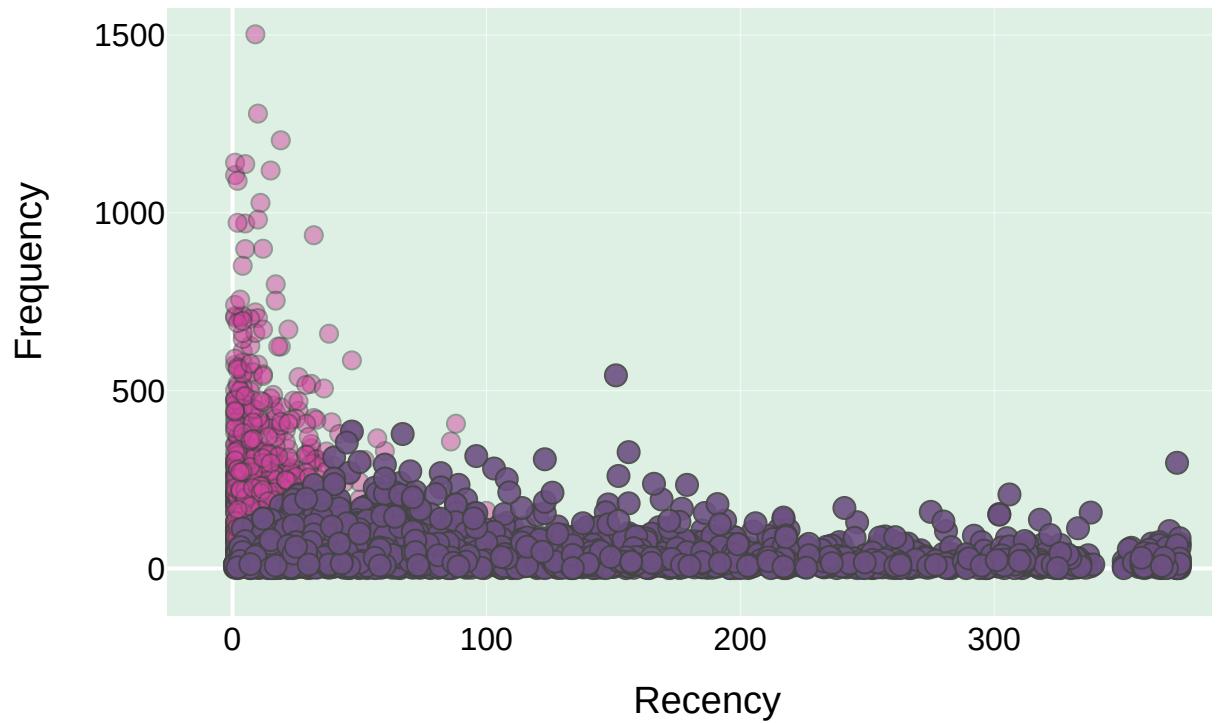
```

```
)  
,  
    gobj.Scatter(  
        x=graph.query("Labels_g == 0")['Recency'],  
        y=graph.query("Labels_g == 0")['Monetary'],  
        mode='markers',  
        name='1st',  
        marker= dict(size= 11,  
                     line= dict(width=1),  
                     color= '#6c5182',  
                     opacity= 0.9  
                )  
,  
#    gobj.Scatter(  
#        x=graph.query("Labels_g == 'Great'")['Recency'],  
#        y=graph.query("Labels_g == 'Great'")['Monetary'],  
#        mode='markers',  
#        name='Great',  
#        marker= dict(size= 13,  
#                     line= dict(width=1),  
#                     color= '#208771',  
#                     opacity= 0.9  
#                )  
#,  
]  
  
plot_layout = gobj.Layout(  
    yaxis= {'title': "Monetary"},  
    xaxis= {'title': "Recency"},  
#    title='Segments'  
)  
fig = gobj.Figure(data=plot_data, layout=plot_layout)  
fig.update_layout(width = 700,title="Monetary vs. Recency",font=dict(  
    family="Arial",  
    size=16,  
    color='#000000'  
, paper_bgcolor='rgba(0,0,0,0)',  
    plot_bgcolor='#ddf0e3', legend=dict(  
        orientation="h",  
        yanchor="bottom",  
        y=1.02,  
        xanchor="right",  
        x=1  
)  
)  
po.iplot(fig)
```

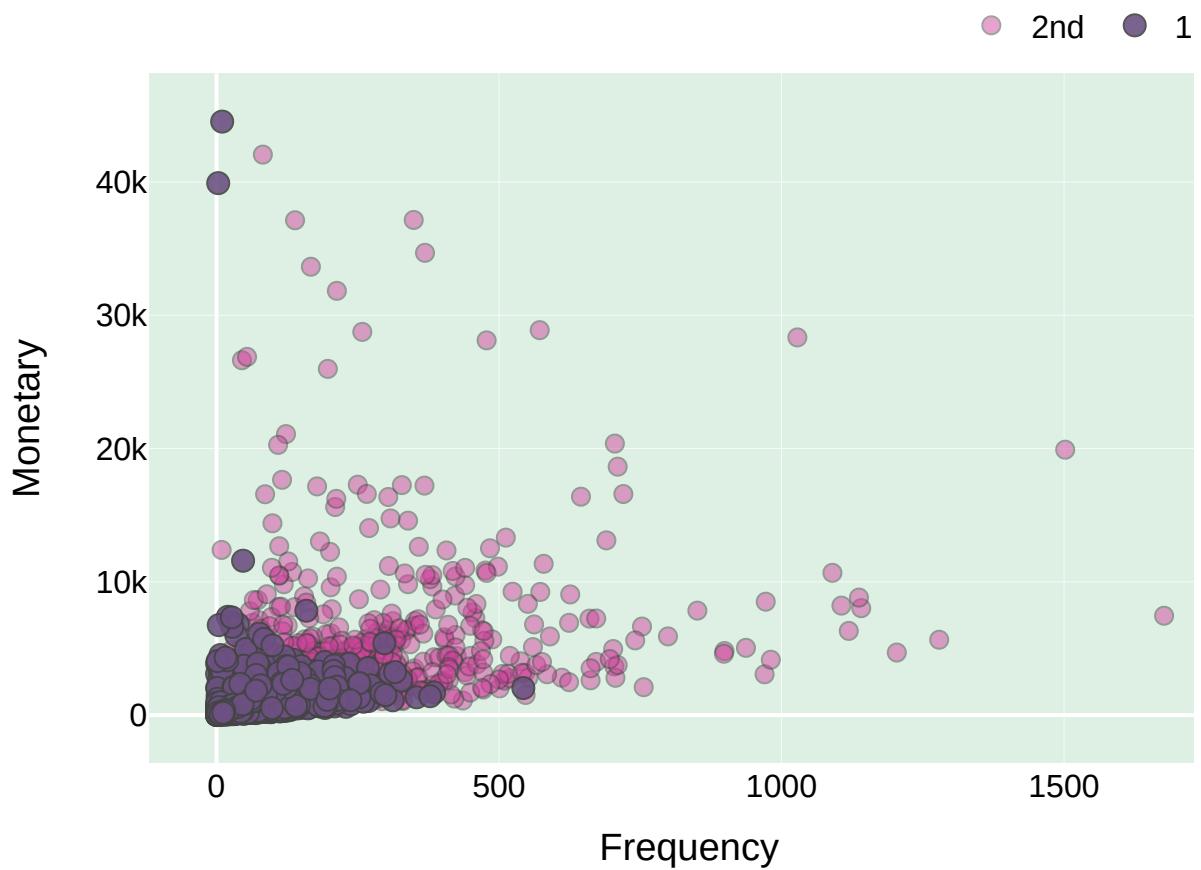
Frequency vs. Recency

● 2nd ● 1

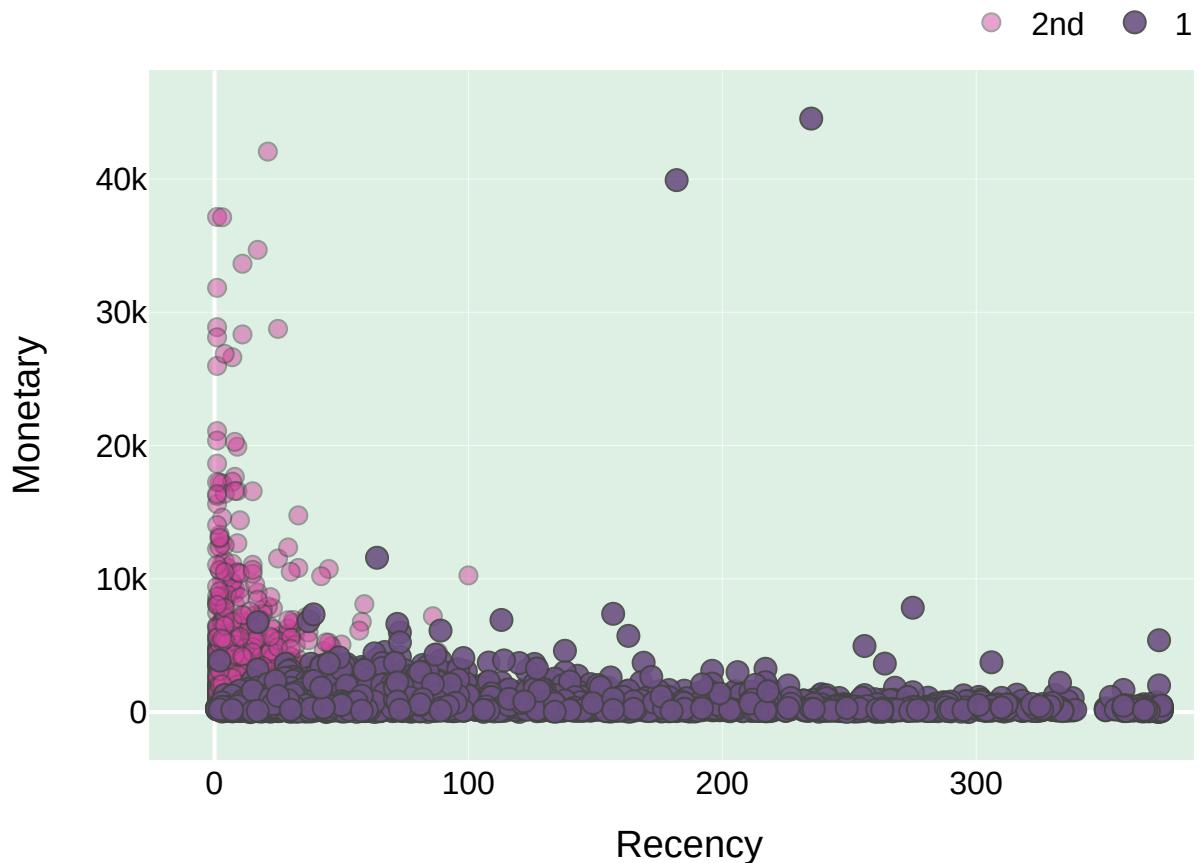




Monetary vs. Frequency



Monetary vs. Recency



در نمودار اول این مجموعه هم مشخص است که دسته ی دوم در زمان نزدیکتری تراکنش داشته اند و هم‌زمان فرکانس های بیشتری هم دارند. همچنین در آخرین گراف نیز هزینه بیشتر و زمان نزدیکتر متعلق به دسته دوم است.

شاید بتوان از این بخش این نتیجه را گرفت که دو دسته کردن مشتری ها را بهتر و بیشتر نمایان می‌کند.

DBSCAN Model

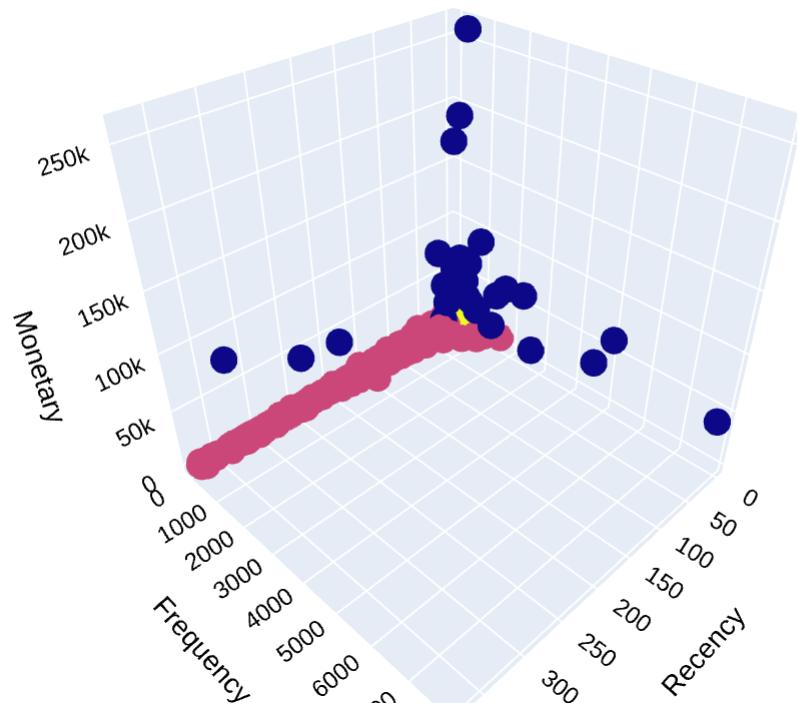
روش آخر که از این کتابخانه انتخاب شد روش

"DBSCAN"

هست. که با توجه به خروجی آن به نظر میرسد دیتای مناسب برای این روش در دسترس نبوده و تفسیر زیادی نمیتوان روی آن داشت.

```
In [42]: db = DBSCAN(eps=1000, min_samples=8).fit(df)
labels = db.labels_
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
fig = px.scatter_3d(df, x='Recency', y='Frequency', z='Monetary', color=labels)
fig.update_layout(width = 700,title="3D scatter Plots",font=dict(
    family="Arial",
    size=12,
    color="#000000"
), paper_bgcolor='rgba(0,0,0,0)', plot_bgcolor='#ddf0e3')
)
fig.show()
```

3D scatter Plots



```
In [43]: RFM['Labels_db']=labels
```

```
In [44]: import chart_studio as cs
import plotly.offline as po
import plotly.graph_objs as gobj

#Recency Vs Frequency
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

plot_data = [
    gobj.Scatter(
        x=graph.query("Labels_db == 2")['Recency'],
        y=graph.query("Labels_db == 2")['Frequency'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                     line= dict(width=1),
                     color= '#faf566',
                     opacity= 0.8
                )
    ),
    gobj.Scatter(
        x=graph.query("Labels_db == 1")['Recency'],
        y=graph.query("Labels_db == 1")['Frequency'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                     line= dict(width=1),
                     color= '#d1439d',
                     opacity= 0.5
                )
    ),
    gobj.Scatter(
        x=graph.query("Labels_db == 0")['Recency'],
        y=graph.query("Labels_db == 0")['Frequency'],
        mode='markers',
        name='1st',
        marker= dict(size= 11,
                     line= dict(width=1),
                     color= '#6c5182',
                     opacity= 0.9
                )
    ),
    #    gobj.Scatter(
    #        x=graph.query("Labels_db == 'Great'")['Recency'],
    #        y=graph.query("Labels_db == 'Great'")['Frequency'],
    #        mode='markers',
    #        name='Great',
    #        marker= dict(size= 13,
    #                     line= dict(width=1),
    #                     color= '#208771',
    #                     opacity= 0.9
    #                )
    #    ),
    ]

plot_layout = gobj.Layout(
    yaxis= {'title': "Frequency"},
```

```
        xaxis= {'title': "Recency"},  
#           title='Segments'  
        )  
fig = gobj.Figure(data=plot_data, layout=plot_layout)  
fig.update_layout(width = 700,title="Frequency vs. Recency",font=dict(  
                           family="Arial",  
                           size=16,  
                           color='#000000'  
                         ), paper_bgcolor='rgba(0,0,0,0)',  
                         plot_bgcolor='#ddf0e3', legend=dict(  
                           orientation="h",  
                           yanchor="bottom",  
                           y=1.02,  
                           xanchor="right",  
                           x=1  
                         )  
                         )  
                         )  
po.iplot(fig)  
  
#Frequency Vs Monetary  
graph = RFM.query("Monetary < 50000 and Frequency < 2000")  
  
plot_data = [  
    gobj.Scatter(  
        x=graph.query("Labels_db == 2")['Frequency'],  
        y=graph.query("Labels_db == 2")['Monetary'],  
        mode='markers',  
        name='3rd',  
        marker= dict(size= 7,  
                     line= dict(width=1),  
                     color= '#faf566',  
                     opacity= 0.8  
                   )  
    ),  
    gobj.Scatter(  
        x=graph.query("Labels_db == 1")['Frequency'],  
        y=graph.query("Labels_db == 1")['Monetary'],  
        mode='markers',  
        name='2nd',  
        marker= dict(size= 9,  
                     line= dict(width=1),  
                     color= '#d1439d',  
                     opacity= 0.5  
                   )  
    ),  
    gobj.Scatter(  
        x=graph.query("Labels_db == 0")['Frequency'],  
        y=graph.query("Labels_db == 0")['Monetary'],  
        mode='markers',  
        name='1st',  
        marker= dict(size= 11,  
                     line= dict(width=1),  
                     color= '#6c5182',  
                     opacity= 0.9  
                   )  
    ),  
#    gobj.Scatter(  
#        x=graph.query("Labels_db == 0")['Frequency'],  
#        y=graph.query("Labels_db == 0")['Monetary'],  
#        mode='markers',  
#        name='0th',  
#        marker= dict(size= 13,  
#                     line= dict(width=1),  
#                     color= '#333399',  
#                     opacity= 0.7  
#                   )  
#    )
```

```

#           x=graph.query("Labels_db == 'Great'")['Frequency'],
#           y=graph.query("Labels_db == 'Great'")['Monetary'],
#           mode='markers',
#           name='Great',
#           marker= dict(size= 13,
#                         line= dict(width=1),
#                         color= '#208771',
#                         opacity= 0.9
#                         )
#       ),
#   ]
#       )

plot_layout = gobj.Layout(
    yaxis= {'title': "Monetary"},
    xaxis= {'title': "Frequency"},
#    title='Segments'
)
fig = gobj.Figure(data=plot_data, layout=plot_layout)
fig.update_layout(width = 700,title="Monetary vs. Frequency",font=dict(
    family="Arial",
    size=16,
    color='#000000'
), paper_bgcolor='rgba(0,0,0,0)',
plot_bgcolor='#ddf0e3', legend=dict(
orientation="h",
yanchor="bottom",
y=1.02,
xanchor="right",
x=1
)
)
po.iplot(fig)

#Recency Vs Monetary
graph = RFM.query("Monetary < 50000 and Frequency < 2000")

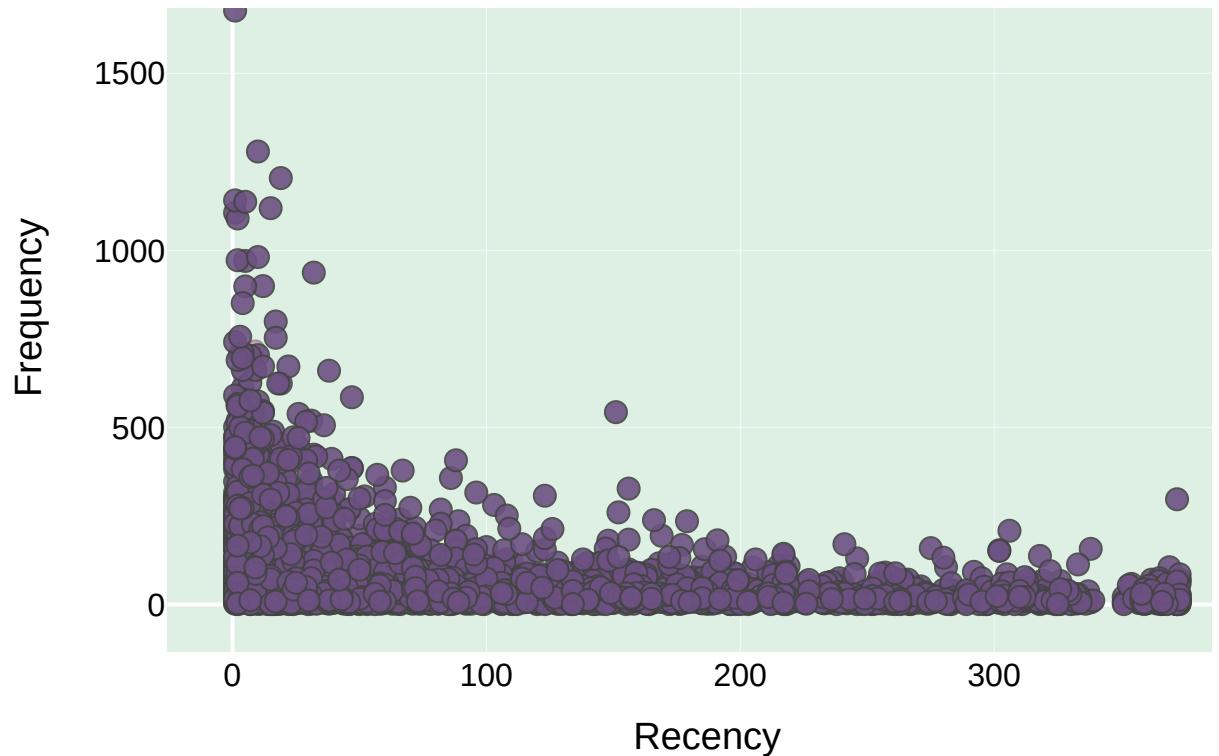
plot_data = [
    gobj.Scatter(
        x=graph.query("Labels_db == 2")['Recency'],
        y=graph.query("Labels_db == 2")['Monetary'],
        mode='markers',
        name='3rd',
        marker= dict(size= 7,
                      line= dict(width=1),
                      color= '#faf566',
                      opacity= 0.8
                      )
    ),
    gobj.Scatter(
        x=graph.query("Labels_db == 1")['Recency'],
        y=graph.query("Labels_db == 1")['Monetary'],
        mode='markers',
        name='2nd',
        marker= dict(size= 9,
                      line= dict(width=1),
                      color= '#d1439d',
                      opacity= 0.5
                      )
]

```

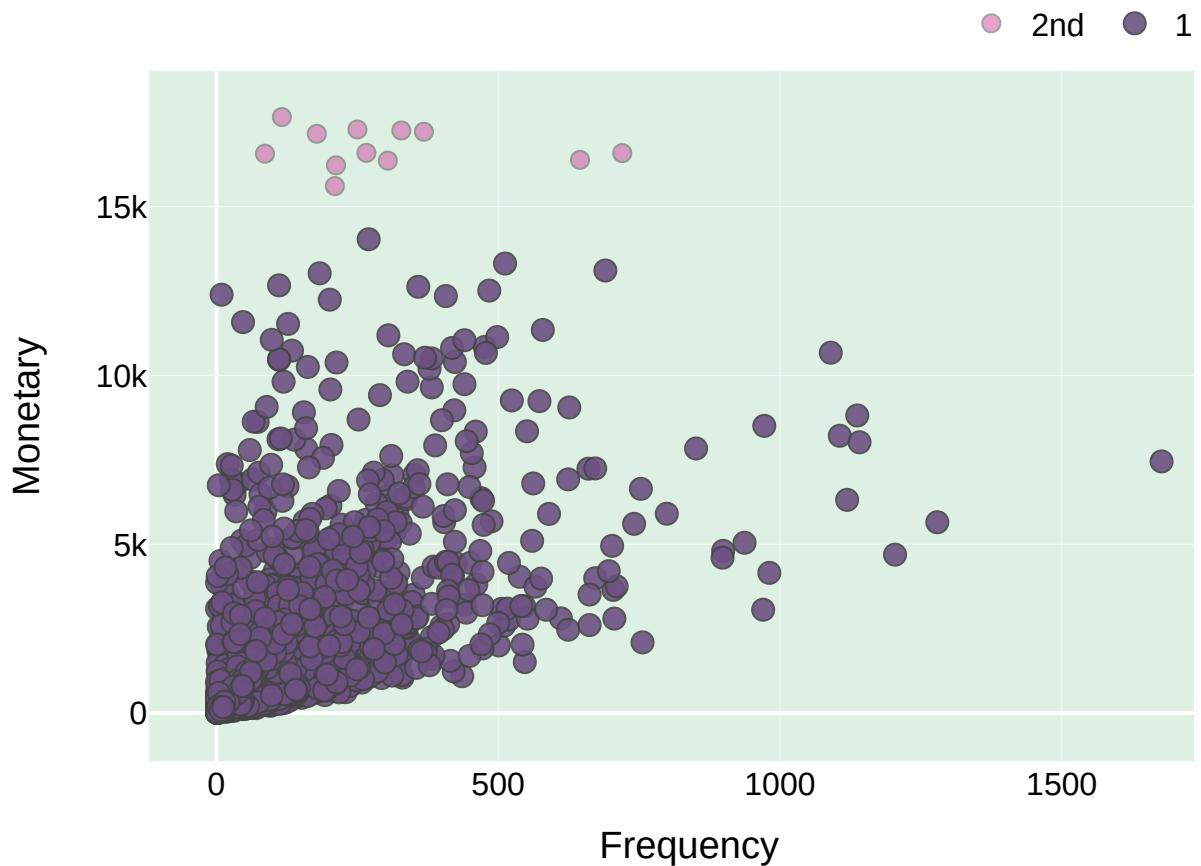
```
)  
,  
    gobj.Scatter(  
        x=graph.query("Labels_db == 0")['Recency'],  
        y=graph.query("Labels_db == 0")['Monetary'],  
        mode='markers',  
        name='1st',  
        marker= dict(size= 11,  
                     line= dict(width=1),  
                     color= '#6c5182',  
                     opacity= 0.9  
                )  
,  
#    gobj.Scatter(  
#        x=graph.query("Labels_db == 'Great'")['Recency'],  
#        y=graph.query("Labels_db == 'Great'")['Monetary'],  
#        mode='markers',  
#        name='Great',  
#        marker= dict(size= 13,  
#                     line= dict(width=1),  
#                     color= '#208771',  
#                     opacity= 0.9  
#                )  
#,  
]  
  
plot_layout = gobj.Layout(  
    yaxis= {'title': "Monetary"},  
    xaxis= {'title': "Recency"},  
#    title='Segments'  
)  
fig = gobj.Figure(data=plot_data, layout=plot_layout)  
fig.update_layout(width = 700,title="Monetary vs. Recency",font=dict(  
    family="Arial",  
    size=16,  
    color='#000000'  
, paper_bgcolor='rgba(0,0,0,0)',  
    plot_bgcolor='#ddf0e3', legend=dict(  
        orientation="h",  
        yanchor="bottom",  
        y=1.02,  
        xanchor="right",  
        x=1  
)  
)  
po.iplot(fig)
```

Frequency vs. Recency

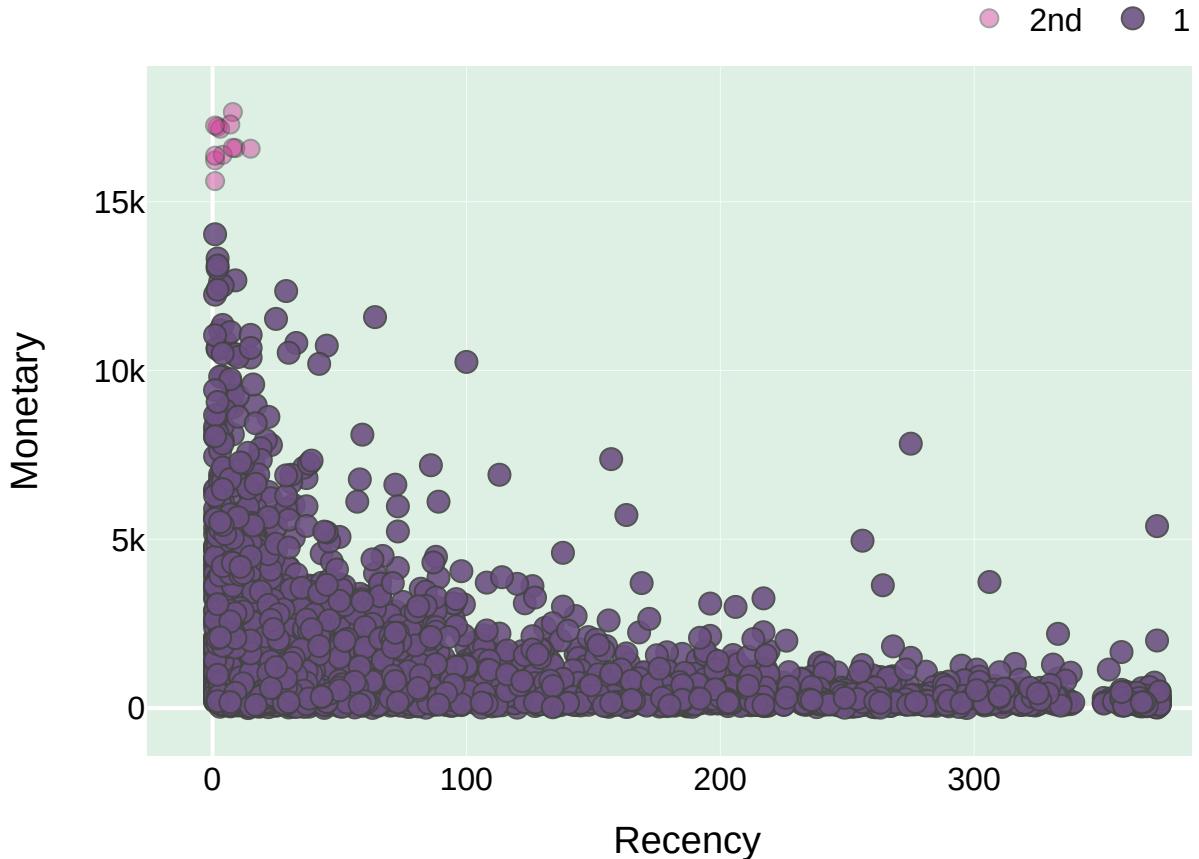
● 2nd ● 1



Monetary vs. Frequency



Monetary vs. Recency



گروه بندی به کمک روش

GaussianMixture

به ما یک ماتریس احتمال هم میدهد. درایه های این ماتریس احتمال عضویت هر نقطه در هر گروه را نشان میدهد اگر نقطه میانی داشته باشیم یعنی نقطه ای که بستگی محکمی به هیچ گروهی نداشته باشد میتوان آن ها را بر اساس این احتمال پیدا کنیم. اهمیت این نقطه های میانی این است که به راحتی میتوان آنها را ... جا به جا کرد. به لحاظ فعالیت و ...

برای مثال کمپینی برای تخفیف به گروهی از مشتریان را اندازی میشود. این دسته از مشتریان میانی در این کمپین شرکت داده میشوند و با خرید پکیج ها میتوان آنها را به دسته های دیگر راه داد

```
In [45]: # probs = gmm.predict_proba(Scaled_Data)
# print(probs[:5].round(3))
```

```
In [ ]:
```

