Customer Life time value by Probabilistic CLV Models

using Lifetimes package

- BG/NBD Model (Predicts future transaction frequency and churn rate)
- GammaGamma Model

In [ ]:
```
import pandas as pd
import lifetimes as lf
from lifetimes.plotting import plot_probability_alive_matrix
import plotly.express as px
```

In [ ]:
```
profit_margin=0.10 # 10% profit
df=pd.read_csv("./Dataset/Cleaned_Online_retail_dec2010-dec2011.csv")
df.head()
```

Out[ ]:

	Invoice	StockCode	Description	Quantity	InvoiceDate	Price	Customer ID	Country	Revenue
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34

In [ ]:
```
df["InvoiceDate"]=pd.to_datetime(df["InvoiceDate"])
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 397885 entries, 0 to 397884
Data columns (total 9 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   Invoice      397885 non-null  int64         
 1   StockCode    397885 non-null  object        
 2   Description  397885 non-null  object        
 3   Quantity     397885 non-null  int64         
 4   InvoiceDate  397885 non-null  datetime64[ns]
 5   Price        397885 non-null  float64       
 6   Customer ID  397885 non-null  int64         
 7   Country      397885 non-null  object        
 8   Revenue      397885 non-null  float64       
dtypes: datetime64[ns](1), float64(2), int64(3), object(3)
memory usage: 27.3+ MB
```

BetaGeo/Negative Binomial Distribution (BG/NBD) Model

Predicts the Buying and Churn Behaviour of customers using R,F,T

According to docs Penalizer is effective in range [0.0 ,0,1]

Feature Engineering

Lifetimes package can automatically produce Featues using its utils module

- Frequency = No of purchases (If package counts as Number of days Customer had a purchase (if F = 0 then only 1time))
- Recency = How Recent a customer visited (if R = 0 then Most Recent)
- T = Timespan of customer in days (from first purchase to last purchase)
- Monetary value = Avg amount spend per transaction

In [ ]:
```
Features=lf.utils.summary_data_from_transaction_data(
    df,
    customer_id_col="Customer ID",
    datetime_col="InvoiceDate",
    monetary_value_col="Revenue"
)
Features.head()
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value
12346	0.0	0.0	325.0	0.000000
12347	6.0	365.0	367.0	599.701667
12348	3.0	283.0	358.0	301.480000
12349	0.0	0.0	18.0	0.000000
12350	0.0	0.0	310.0	0.000000

In [ ]:
```
bgf=lf.BetaGeoFitter(penalizer_coef=0.0)
```

In [ ]:
```
bgf.fit(frequency=Features["frequency"],recency=Features["recency"],T=Features["T"])
```

Out[ ]:
```
<lifetimes.BetaGeoFitter: fitted with 4338 subjects, a: 0.00, alpha: 68.91, b: 6.75, r: 0.83>
```

In [ ]:
```
bgf.summary
```

Out[ ]:

	coef	se(coef)	lower 95% bound	upper 95% bound
r	0.826542	0.026785	0.774044	0.879040
alpha	68.905121	2.611786	63.786020	74.024221
a	0.003437	0.010339	-0.016828	0.023702
b	6.745116	22.414725	-37.187745	50.677978

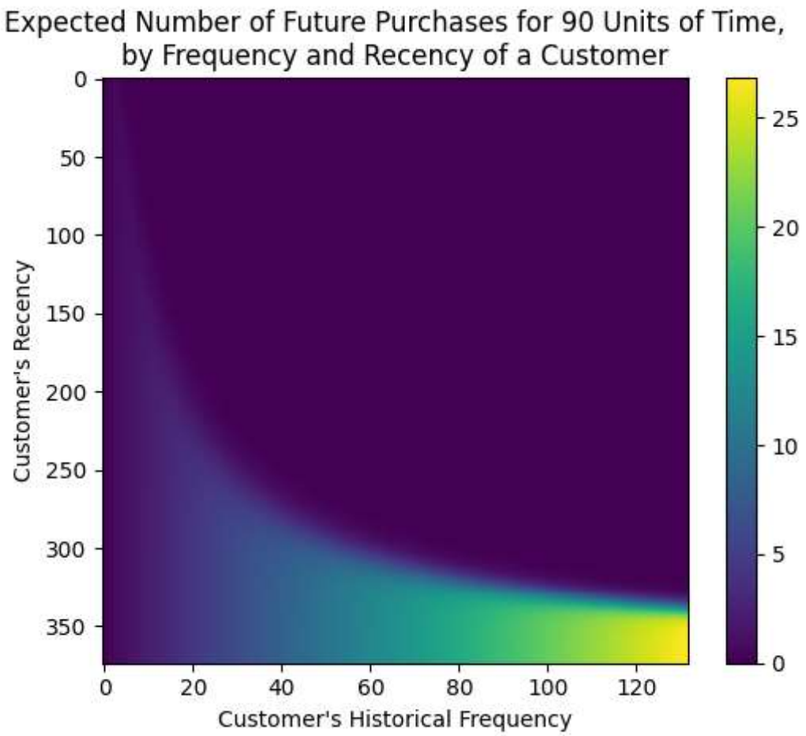
In [ ]:
```
Features["Probability_alive"]=bgf.conditional_probability_alive(frequency=Features["frequency"],recency=Features["recency"],T=Features["T"])
Features["expected_purchases_in_90days"]=bgf.conditional_expected_number_of_purchases_up_to_time(t=90,frequency=Features["frequency"],recency=Features["recency"],T=Features["T"])
Features.head()
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value	Probability_alive	expected_purchases_in_90days
12346	0.0	0.0	325.0	0.000000	1.000000	0.188830
12347	6.0	365.0	367.0	599.701667	0.999698	1.408736
12348	3.0	283.0	358.0	301.480000	0.999177	0.805907
12349	0.0	0.0	18.0	0.000000	1.000000	0.855607
12350	0.0	0.0	310.0	0.000000	1.000000	0.196304

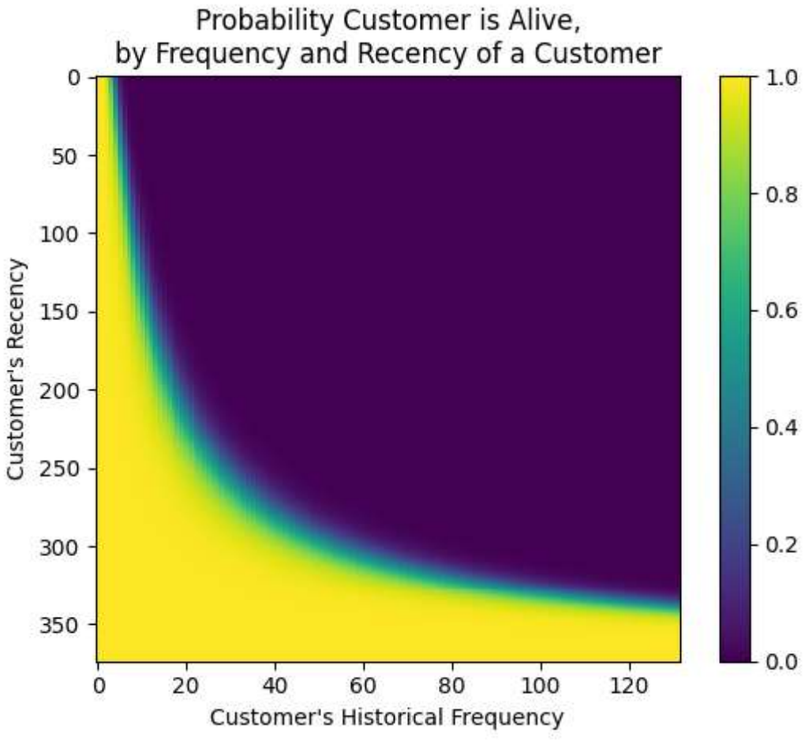
In [ ]:
```
from lifetimes.plotting import plot_frequency_recency_matrix, plot_probability_alive_matrix, plot_period_transactions
plot_frequency_recency_matrix(bgf,T=90)
```

Out[ ]: <Axes: title={'center': 'Expected Number of Future Purchases for 90 Units of Time,\nby Frequency and Recency of a Customer'}, xlabel="Customer's Historical Frequency", ylabel="Customer's Recency">



In [ ]:
```
plot_probability_alive_matrix(bgf)
```

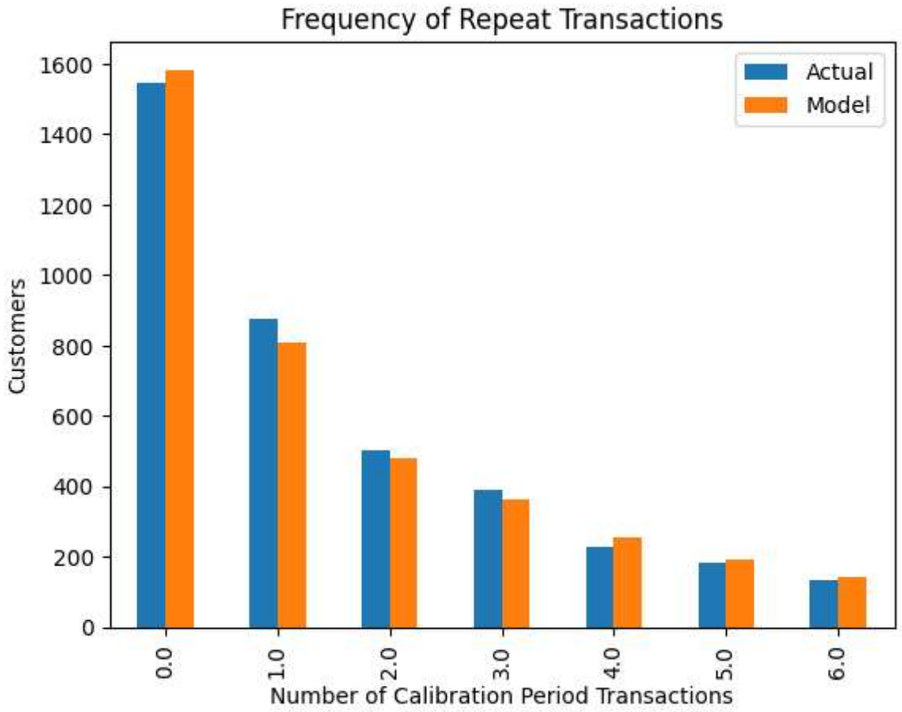
Out[ ]: <Axes: title={'center': 'Probability Customer is Alive,\nby Frequency and Recency of a Customer'}, xlabel="Customer's Historical Frequency", ylabel="Customer's Recency">



Assesing Model fitness

In [ ]:
```
plot_period_transactions(bgf)
```

Out[ ]:
```
<Axes: title={'center': 'Frequency of Repeat Transactions'}, xlabel='Number of Calibration Period Transactions', ylabel='Customers'>
```



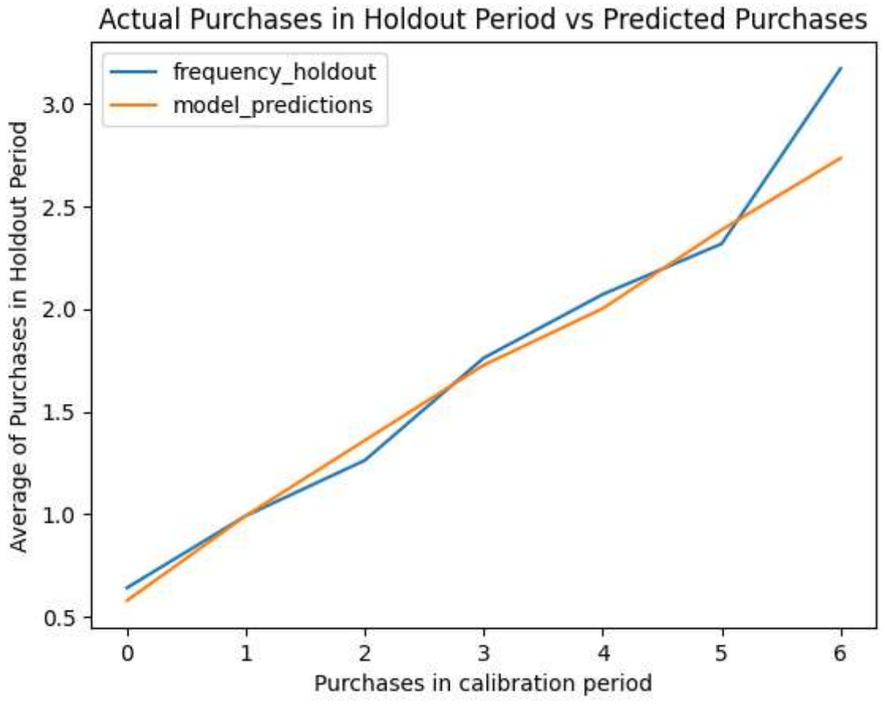
In [ ]:
```
# Splitting data for training(6months) and testing(4months)
from lifetimes.utils import calibration_and_holdout_data
from lifetimes.plotting import plot_calibration_purchases_vs_holdout_purchases
df_Cal_hold=calibration_and_holdout_data(df,
                                         customer_id_col="Customer ID",
                                         datetime_col="InvoiceDate",
                                         calibration_period_end="2011-09-01",
                                         observation_period_end="2011-12-31")
df_Cal_hold.head()
```

Out[ ]:

Customer ID	frequency_cal	recency_cal	T_cal	frequency_holdout	duration_holdout
12346	0.0	0.0	226.0	0.0	121.0
12347	4.0	238.0	268.0	2.0	121.0
12348	2.0	110.0	259.0	1.0	121.0
12350	0.0	0.0	211.0	0.0	121.0
12352	3.0	34.0	197.0	3.0	121.0

In [ ]:
```
plot_calibration_purchases_vs_holdout_purchases(bgf,df_Cal_hold)
```

Out[ ]:
```
<Axes: title={'center': 'Actual Purchases in Holdout Period vs Predicted Purchases'}, xlabel='Purchases in calibration period', ylabel='Average of Purchases in Holdout Period'>
```



GammaGamma Model

- Predicts the CLV
- Assumes that Frequency & Monetary has Pearson Correaltion near to 0

In [ ]:
```
Features=Features[Features["frequency"]>0]
Features.head()
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value	Probability_alive	expected_purchases_in_90days
12347	6.0	365.0	367.0	599.701667	0.999698	1.408736
12348	3.0	283.0	358.0	301.480000	0.999177	0.805907
12352	6.0	260.0	296.0	368.256667	0.999406	1.682277
12356	2.0	303.0	325.0	269.905000	0.999478	0.645368
12358	1.0	149.0	150.0	683.200000	0.999486	0.750390

As Frquency = 0 means customers bought only once

In [ ]:
```
Features[Features["monetary_value"]<=0]
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value	Probability_alive	expected_purchases_in_90days

In [ ]:
```
Features[["frequency","monetary_value"]].corr()
```

Out[ ]:

	frequency	monetary_value
frequency	1.000000	0.015905
monetary_value	0.015905	1.000000

In [ ]:
```
from lifetimes import GammaGammaFitter
ggf=GammaGammaFitter()
```

In [ ]:
```
ggf.fit(frequency=Features["frequency"],monetary_value=Features["monetary_value"])
ggf
```

Out[ ]:
```
<lifetimes.GammaGammaFitter: fitted with 2790 subjects, p: 2.10, q: 3.45, v: 485.92>
```

In [ ]:
```
Features["Expected_conditional_avg_revenue"]=ggf.conditional_expected_average_profit(
                                                frequency=Features["frequency"],
                                                monetary_value=Features["monetary_value"])
```

In [ ]:
```
Features["Predict_CLV_3Months"]=ggf.customer_lifetime_value(bgf,
                                Features["frequency"],
                                Features["recency"],
                                Features["T"],
                                Features["monetary_value"],
                                time=3 ,  #in months
                                freq='D'  #frquency is in days
                                )
Features.head()
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value	Probability_alive	expected_purchases_in_90days	Expected_conditional_avg_revenue	Predict_CLV_3Months
12347	6.0	365.0	367.0	599.701667	0.999698	1.408736	569.977820	787.153453
12348	3.0	283.0	358.0	301.480000	0.999177	0.805907	333.785887	263.709035
12352	6.0	260.0	296.0	368.256667	0.999406	1.682277	376.175965	620.384166
12356	2.0	303.0	325.0	269.905000	0.999478	0.645368	324.041778	205.012619
12358	1.0	149.0	150.0	683.200000	0.999486	0.750390	539.904594	397.169912

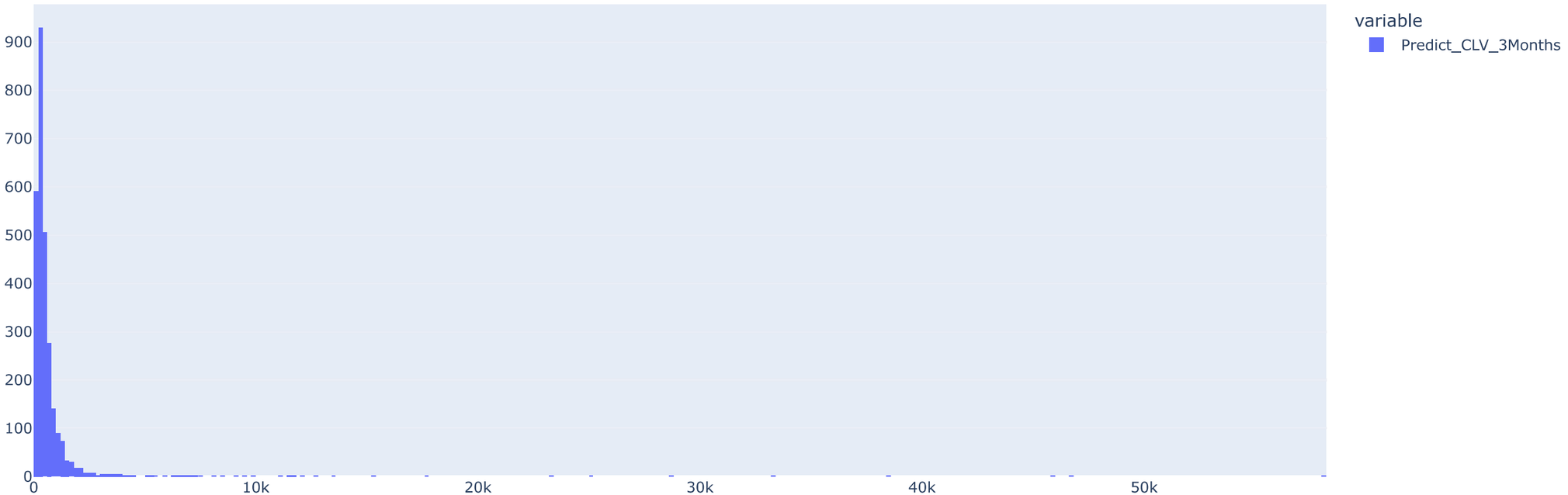
In [ ]:
```
print("Expected Conditional Avg Revenue : %s \n  Avg Profit : %s"%(
    Features["Expected_conditional_avg_revenue"].mean(),
    Features["monetary_value"].mean()
))
```

```
Expected Conditional Avg Revenue : 454.5955413501255
  Avg Profit : 477.3824890515858
```

In [ ]:
```
# Scatter plot for avg monetary value from data vs gamma model
fig=px.scatter(Features,x=Features["monetary_value"],y=Features["Expected_conditional_avg_revenue"])
fig.update_layout(xaxis_title="Actual",yaxis_title=" Gamma Model",title={
            'text' : 'Avg Monetary for customers ',
            'x':0.5,
            'xanchor': 'center'
        })
fig.show("notebook")
```

In [ ]:
```
fig=px.histogram(Features["expected_purchases_in_90days"])
fig.update_layout(xaxis_title="",yaxis_title="",title={
            'text' : 'Histogram of expected_purchases in next 3months',
            'x':0.5,
            'xanchor': 'center'
        })
fig.show("notebook")
```

In [ ]:
```
fig=px.histogram(Features["Predict_CLV_3Months"])
fig.update_layout(xaxis_title="",yaxis_title="",title={
            'text' : 'Histogram of Predicted CLV for next 3Months',
            'x':0.5,
            'xanchor': 'center'
        })
fig.show("notebook")
```



In [ ]:
```
#bgf.save_model("./Models/BgNbdModel-Purchase-Churn-behaviour.pickle")
#ggf.save_model("./Models/gammagammaModel-CLV-prediction.pickle")
```

Customer Segmentation

In [ ]:
```
x=Features[["frequency","recency","T","monetary_value","Predict_CLV_3Months"]]
x.head()
```

Out[ ]:

Customer ID	frequency	recency	T	monetary_value	Predict_CLV_3Months
12347	6.0	365.0	367.0	599.701667	787.153453
12348	3.0	283.0	358.0	301.480000	263.709035
12352	6.0	260.0	296.0	368.256667	620.384166
12356	2.0	303.0	325.0	269.905000	205.012619
12358	1.0	149.0	150.0	683.200000	397.169912

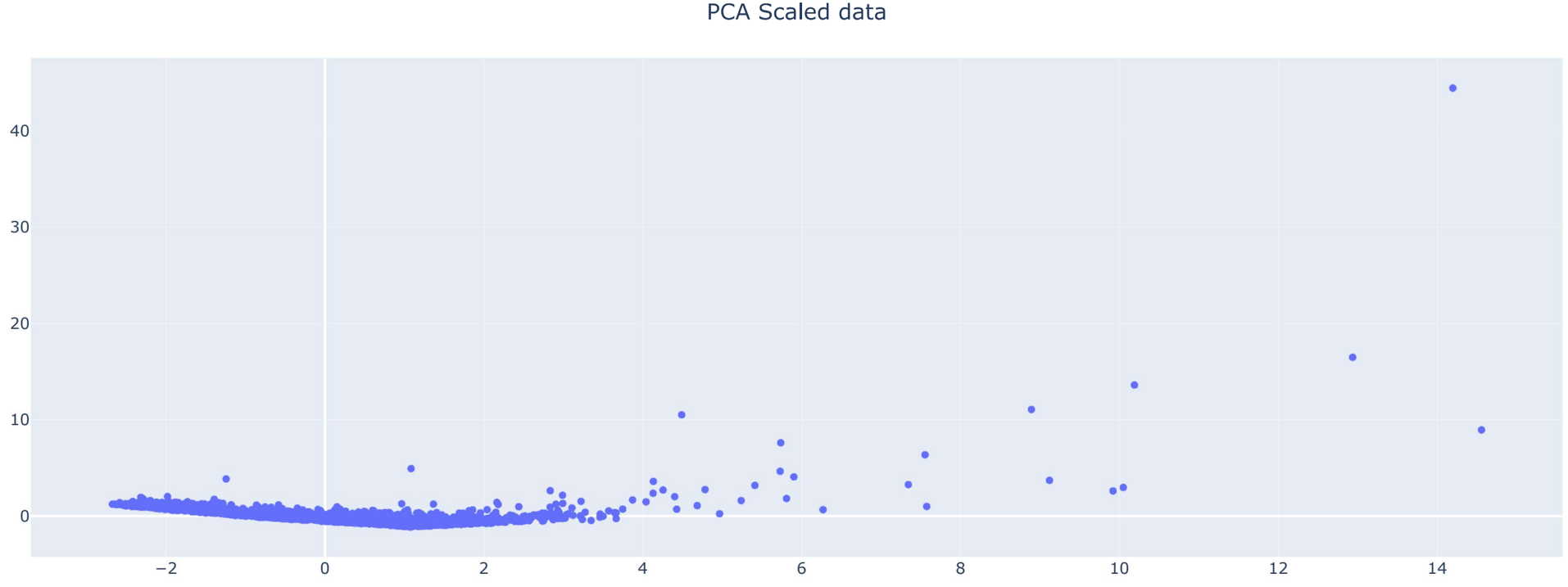
In [ ]:
```
#Scalling
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()
x_scaled=scaler.fit_transform(x)
x_scaled.shape
```

Out[ ]:
```
(2790, 5)
```

In [ ]:
```
#Dimensionality reduction
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
pca_scaled=pca.fit_transform(x_scaled)
pca_scaled.shape
```

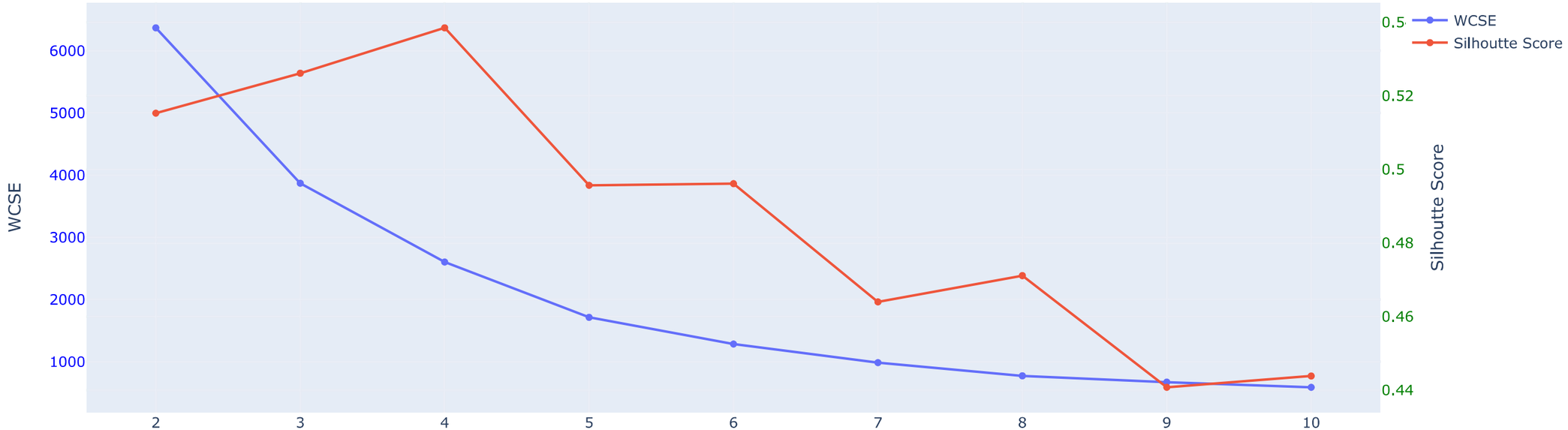
Out[ ]:
```
(2790, 2)
```

In [ ]:
```
fig=px.scatter(x=pca_scaled[:,0],y=pca_scaled[:,1])
fig.update_layout(xaxis_title="",yaxis_title="",title={
            'text' : 'PCA Scaled data',
            'x':0.5,
            'xanchor': 'center'
        })
fig.show("notebook")
```



In [ ]:
```
# elbow method
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
wcse=[]
s_score=[]
for i in range(2,11):
    model=KMeans(n_clusters=i,init="k-means++")
    model.fit(pca_scaled)
    wcse.append(model.inertia_)
    s_score.append(silhouette_score(pca_scaled,model.labels_))
```

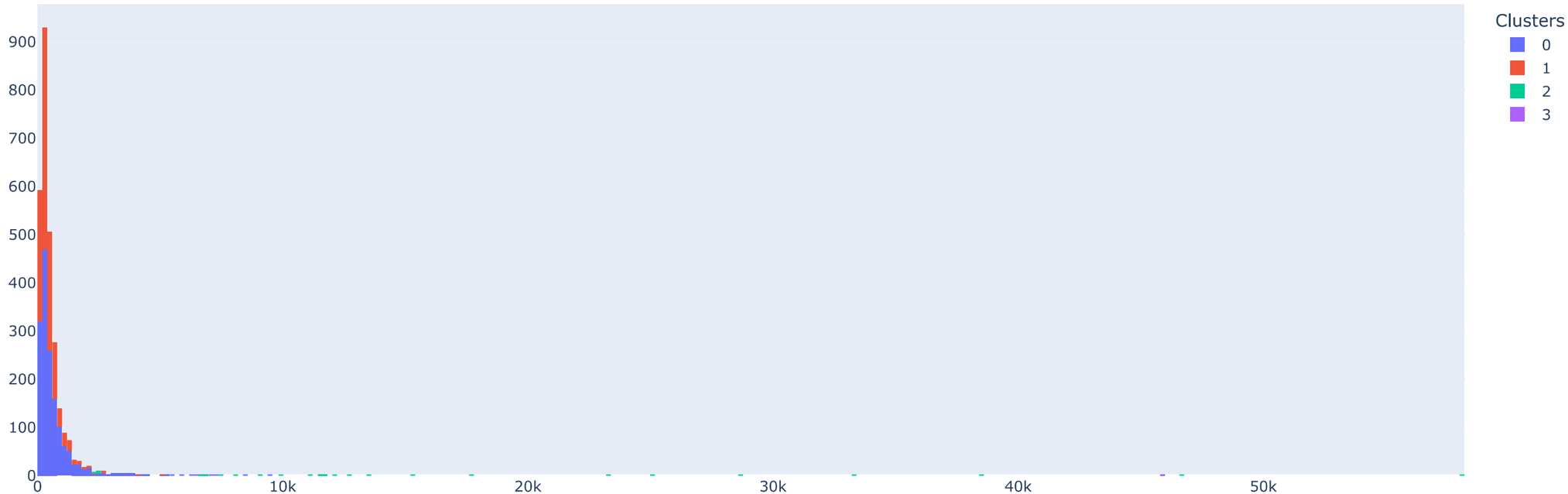
In [ ]:
```
import plotly.graph_objects as go
t1=go.Scatter(x=list(range(2,11)),y=wcse,name="WCSE",yaxis="y1")
t2=go.Scatter(x=list(range(2,11)),y=s_score,name="Silhoutte Score",yaxis="y2")
layout=go.Layout(title="ELbow Method For Kmeans",
                 yaxis=dict(
                     title="WCSE",
                     tickfont=dict(color="blue")
                 ),
                 yaxis2=dict(
                     title="Silhoutte Score",
                     tickfont=dict(color="green"),
                     overlaying="y",
                     side="right"
                 )
                 )
fig=go.Figure(data=[t1,t2],layout=layout)
fig.show("notebook")
```



for k=4 has elbow and high silhoutte score

In [ ]:
```
kmodel=KMeans(n_clusters=4,init="k-means++")
kmodel.fit(pca_scaled)
px.scatter(x=pca_scaled[:,0],y=pca_scaled[:,1],color=kmodel.labels_)
```

In [ ]:
```
Features["Clusters"]=kmodel.labels_
fig=px.histogram(Features,x=["Predict_CLV_3Months"],color="Clusters")
fig.update_layout(xaxis_title="",yaxis_title="",title={
            'text' : 'Histogram of Predicted CLV for next 3Monthsn by CLusters',
            'x':0.5,
            'xanchor': 'center'
        })
fig.show("notebook")
```



In [ ]:
```
#Features.to_csv("./Outputs/Customers-BG-Gamama-Predictions-3months.csv")
```

Summary

In [ ]:
```
Cluster_summary=Features.groupby("Clusters").agg(
    Avg_CLV_Predicted=("Predict_CLV_3Months","mean"),
   MinCLV_Predicted=("Predict_CLV_3Months","min"),
   MaxCLV_predicted=("Predict_CLV_3Months","max"),
   Customers=("Predict_CLV_3Months","count"),
   Predictive_Future_Avg_Purchases_3mon=("expected_purchases_in_90days","mean"),
   past_Avg_monetary=("monetary_value","mean")
)
Cluster_summary
```

Out[ ]:

Clusters	Avg_CLV_Predicted	MinCLV_Predicted	MaxCLV_predicted	Customers	Predictive_Future_Avg_Purchases_3mon	past_Avg_monetary
0	639.746344	84.496959	9524.745693	1552	1.488705	415.995642
1	434.963946	85.761323	5117.442038	1214	1.175910	374.755979
2	17933.864012	2315.455405	58083.494336	23	10.631349	2732.545650
3	45874.271033	45874.271033	45874.271033	1	0.599743	168469.600000