```
1 import pandas as pd
2 import numpy as np
3 import matplotlib.pyplot as plt
4 import seaborn as sns
5 import  plotly.express as px
6 import sklearn
```

```
1 df=pd.read_csv("/content/Customer Data.csv")
```

```
1 df.head()
```

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOF
0	C10001	40.900749	0.818182	95.40	0.00	95.4	0.000000	0.166667	
1	C10002	3202.467416	0.909091	0.00	0.00	0.0	6442.945483	0.000000	
2	C10003	2495.148862	1.000000	773.17	773.17	0.0	0.000000	1.000000	
3	C10004	1666.670542	0.636364	1499.00	1499.00	0.0	205.788017	0.083333	
4	C10005	817.714335	1.000000	16.00	16.00	0.0	0.000000	0.083333	

```
1 df.max()
```

```
CUST_ID                             C19190
BALANCE                          19043.13856
BALANCE_FREQUENCY                        1.0
PURCHASES                            49039.57
ONEOFF_PURCHASES                     40761.25
INSTALLMENTS_PURCHASES                22500.0
CASH_ADVANCE                     47137.21176
PURCHASES_FREQUENCY                      1.0
ONEOFF_PURCHASES_FREQUENCY               1.0
PURCHASES_INSTALLMENTS_FREQUENCY         1.0
CASH_ADVANCE_FREQUENCY                   1.5
CASH_ADVANCE_TRX                         123
PURCHASES_TRX                            358
CREDIT_LIMIT                         30000.0
PAYMENTS                         50721.48336
MINIMUM_PAYMENTS                 76406.20752
PRC_FULL_PAYMENT                         1.0
TENURE                                    12
dtype: object
```

```
1 df.min()
```

```
CUST_ID                             C10001
BALANCE                                0.0
BALANCE_FREQUENCY                      0.0
PURCHASES                              0.0
ONEOFF_PURCHASES                       0.0
INSTALLMENTS_PURCHASES                 0.0
CASH_ADVANCE                           0.0
PURCHASES_FREQUENCY                    0.0
ONEOFF_PURCHASES_FREQUENCY             0.0
PURCHASES_INSTALLMENTS_FREQUENCY       0.0
CASH_ADVANCE_FREQUENCY                 0.0
CASH_ADVANCE_TRX                         0
PURCHASES_TRX                            0
CREDIT_LIMIT                          50.0
PAYMENTS                               0.0
MINIMUM_PAYMENTS                  0.019163
PRC_FULL_PAYMENT                       0.0
TENURE                                   6
dtype: object
```

```
1 df.shape
```

```
(8950, 18)
```

```
1 df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8950 entries, 0 to 8949
```

```
Data columns (total 18 columns):
 #   Column                            Non-Null Count  Dtype  
---  ------                            --------------  -----  
 0   CUST_ID                           8950 non-null   object 
 1   BALANCE                           8950 non-null   float64
 2   BALANCE_FREQUENCY                 8950 non-null   float64
 3   PURCHASES                         8950 non-null   float64
 4   ONEOFF_PURCHASES                  8950 non-null   float64
 5   INSTALLMENTS_PURCHASES            8950 non-null   float64
 6   CASH_ADVANCE                      8950 non-null   float64
 7   PURCHASES_FREQUENCY               8950 non-null   float64
 8   ONEOFF_PURCHASES_FREQUENCY        8950 non-null   float64
 9   PURCHASES_INSTALLMENTS_FREQUENCY  8950 non-null   float64
 10  CASH_ADVANCE_FREQUENCY            8950 non-null   float64
 11  CASH_ADVANCE_TRX                  8950 non-null   int64  
 12  PURCHASES_TRX                     8950 non-null   int64  
 13  CREDIT_LIMIT                      8949 non-null   float64
 14  PAYMENTS                          8950 non-null   float64
 15  MINIMUM_PAYMENTS                  8637 non-null   float64
 16  PRC_FULL_PAYMENT                  8950 non-null   float64
 17  TENURE                            8950 non-null   int64  
dtypes: float64(14), int64(3), object(1)
memory usage: 1.2+ MB
```

```
1 df.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_
count	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	8950.000000	
mean	1564.474828	0.877271	1003.204834	592.437371	411.067645	978.871112	0.490351	
std	2081.531879	0.236904	2136.634782	1659.887917	904.338115	2097.163877	0.401371	
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	128.281915	0.888889	39.635000	0.000000	0.000000	0.000000	0.083333	
50%	873.385231	1.000000	361.280000	38.000000	89.000000	0.000000	0.500000	
75%	2054.140036	1.000000	1110.130000	577.405000	468.637500	1113.821139	0.916667	
max	19043.138560	1.000000	49039.570000	40761.250000	22500.000000	47137.211760	1.000000	

```
1 df.isnull().sum()
```

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        1
PAYMENTS                            0
MINIMUM_PAYMENTS                  313
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

```
1 !pip install feature-engine
```

```
Collecting feature-engine
  Downloading feature_engine-1.6.2-py2.py3-none-any.whl (328 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 328.9/328.9 kB 3.5 MB/s eta 0:00:00
Requirement already satisfied: numpy>=1.18.2 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.23.5)
Requirement already satisfied: pandas>=1.0.3 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.5.3)
Requirement already satisfied: scikit-learn>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.2.2)
Requirement already satisfied: scipy>=1.4.1 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (1.11.4)
Requirement already satisfied: statsmodels>=0.11.1 in /usr/local/lib/python3.10/dist-packages (from feature-engine) (0.14.1)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature-engine) (2
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=1.0.3->feature-engine) (2023.3.post
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->feature-engine) (1.3.
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=1.0.0->feature-engine
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature-engine) (0.5.5
```

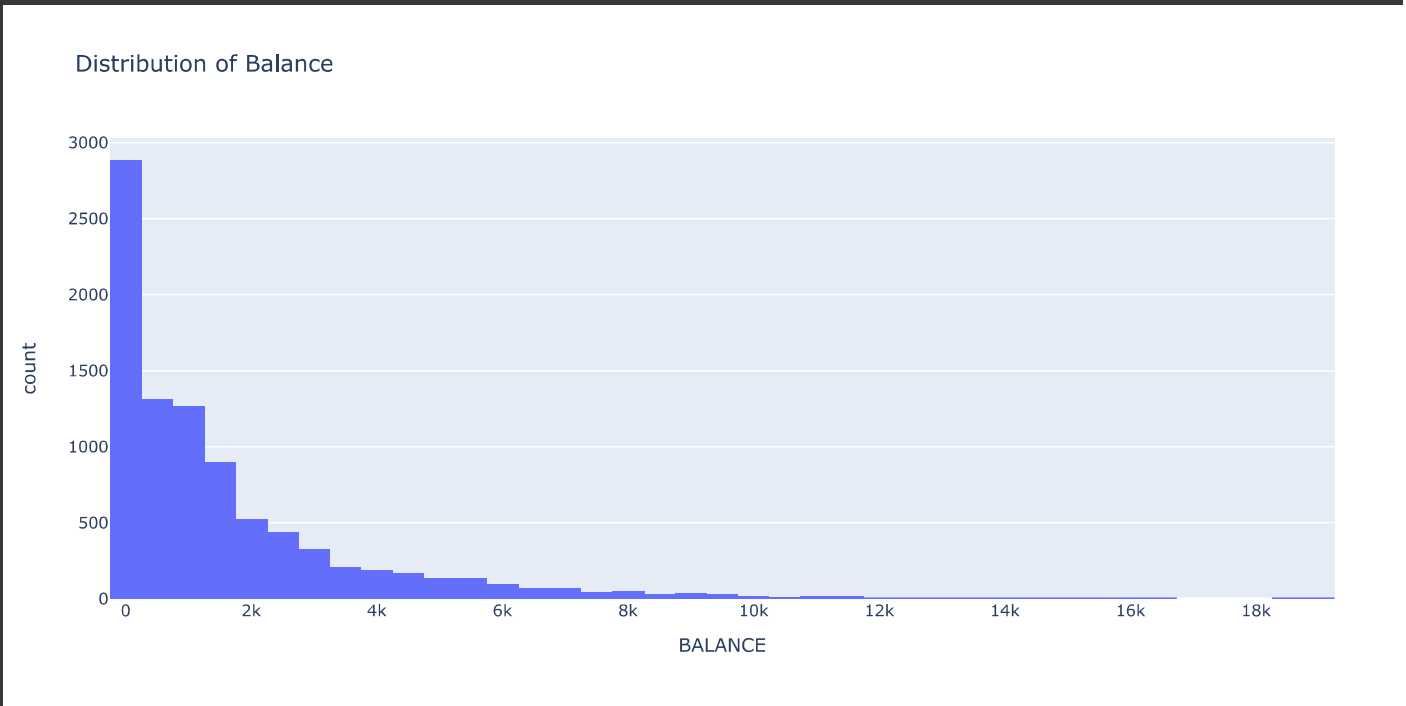
```
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.11.1->feature-engine) (23
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels>=0.11.1->feature-engine) (
Installing collected packages: feature-engine
Successfully installed feature-engine-1.6.2
```

```
from feature_engine.imputation import MeanMedianImputer
impute=MeanMedianImputer()
df=impute.fit_transform(df)
```

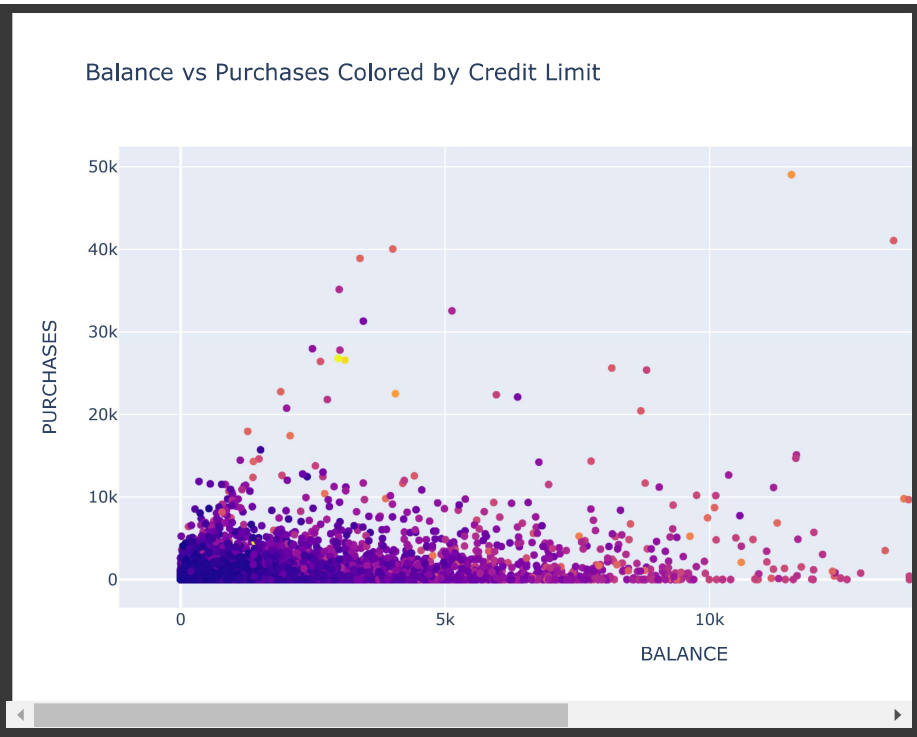
```
df.isnull().sum()
```

```
CUST_ID                             0
BALANCE                             0
BALANCE_FREQUENCY                   0
PURCHASES                           0
ONEOFF_PURCHASES                    0
INSTALLMENTS_PURCHASES              0
CASH_ADVANCE                        0
PURCHASES_FREQUENCY                 0
ONEOFF_PURCHASES_FREQUENCY          0
PURCHASES_INSTALLMENTS_FREQUENCY    0
CASH_ADVANCE_FREQUENCY              0
CASH_ADVANCE_TRX                    0
PURCHASES_TRX                       0
CREDIT_LIMIT                        0
PAYMENTS                            0
MINIMUM_PAYMENTS                    0
PRC_FULL_PAYMENT                    0
TENURE                              0
dtype: int64
```

```
import plotly.express as px

fig = px.histogram(df, x='BALANCE', nbins=50, title='Distribution of Balance')
fig.show()
```

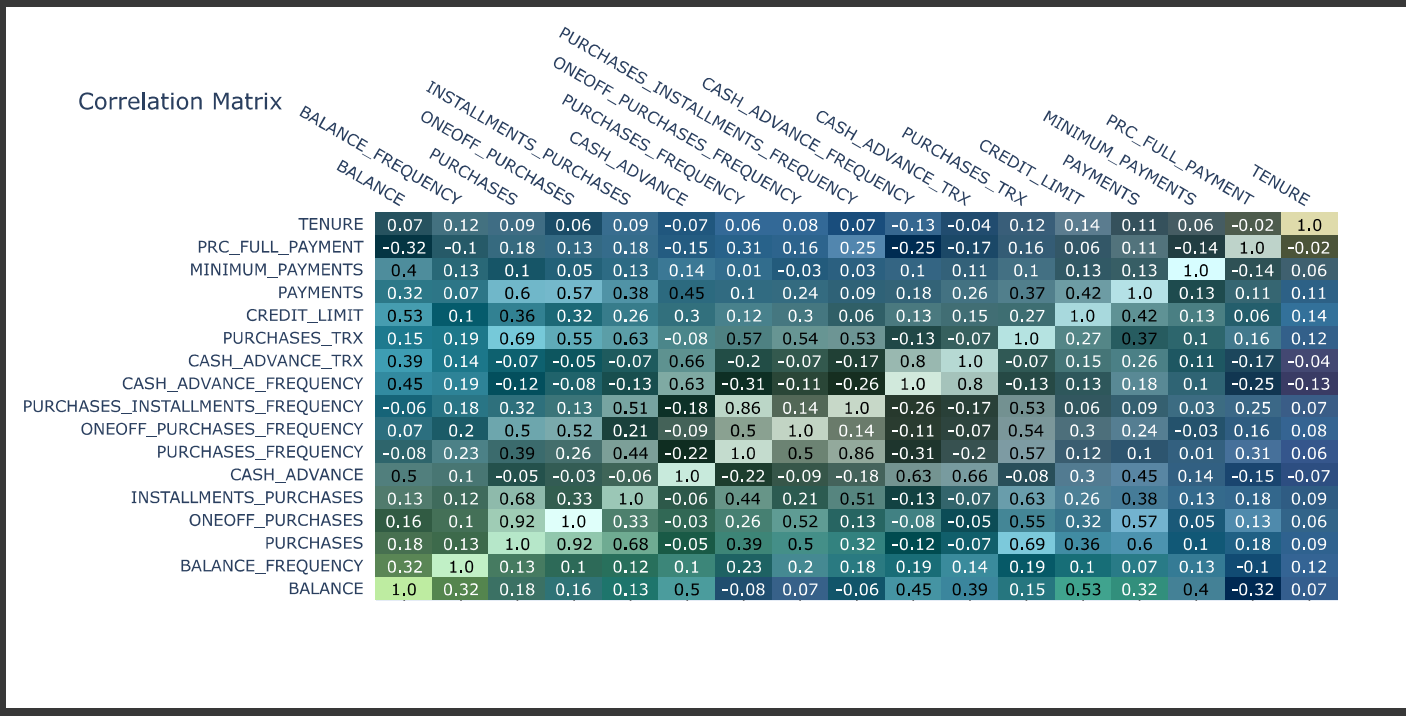



```
px.scatter(df, x='BALANCE', y='PURCHASES', color='CREDIT_LIMIT',
                  title='Balance vs Purchases Colored by Credit Limit')
```

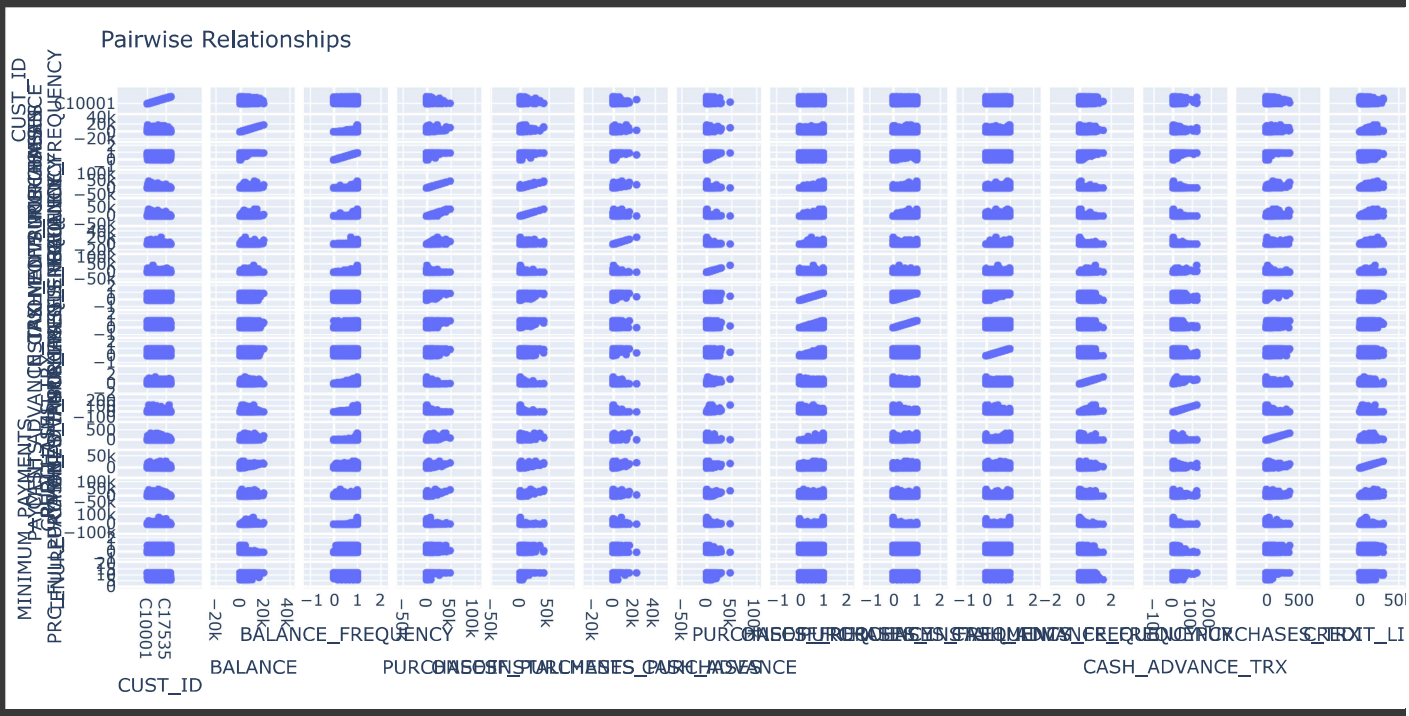



```
#Correlation Matrix
import plotly.figure_factory as ff

correlation_matrix = df.corr()
fig = ff.create_annotated_heatmap(
    z=correlation_matrix.to_numpy(),
    x=list(correlation_matrix.columns),
    y=list(correlation_matrix.index),
    annotation_text=correlation_matrix.round(2).values,
    colorscale='Viridis',
)
fig.update_layout(title='Correlation Matrix')
fig.show()

```

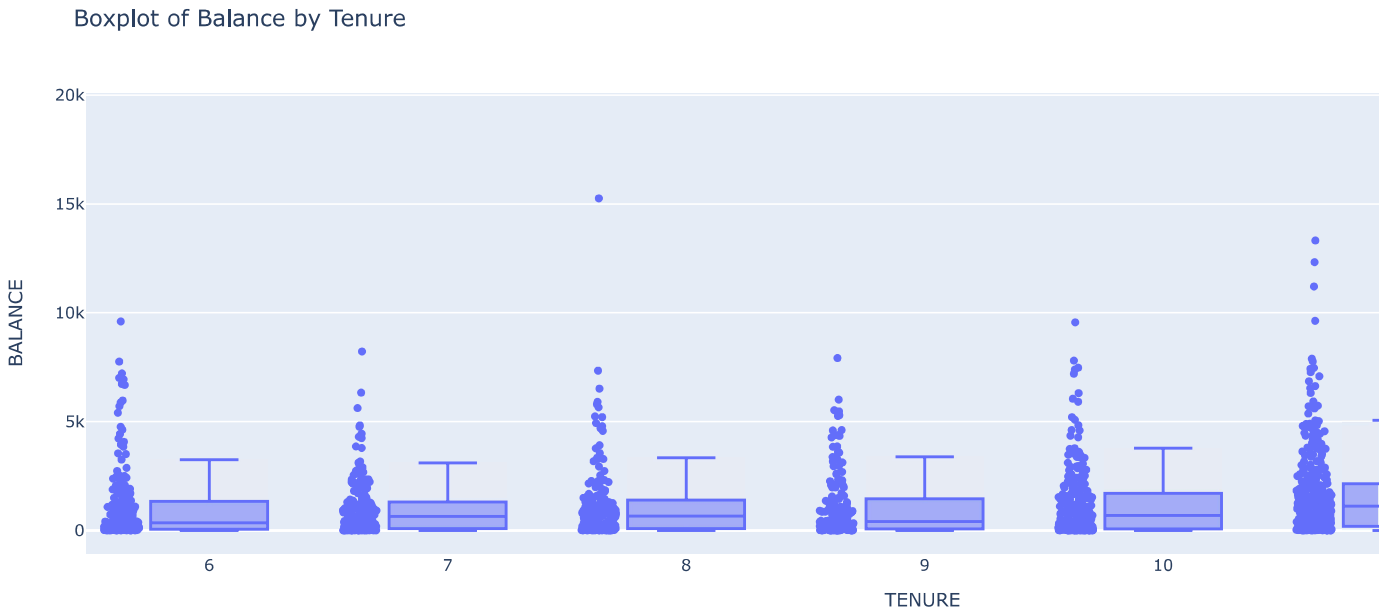
```
<ipython-input-15-6a6cf0a4cd83>:4: FutureWarning:

The default value of numeric_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid colu
```



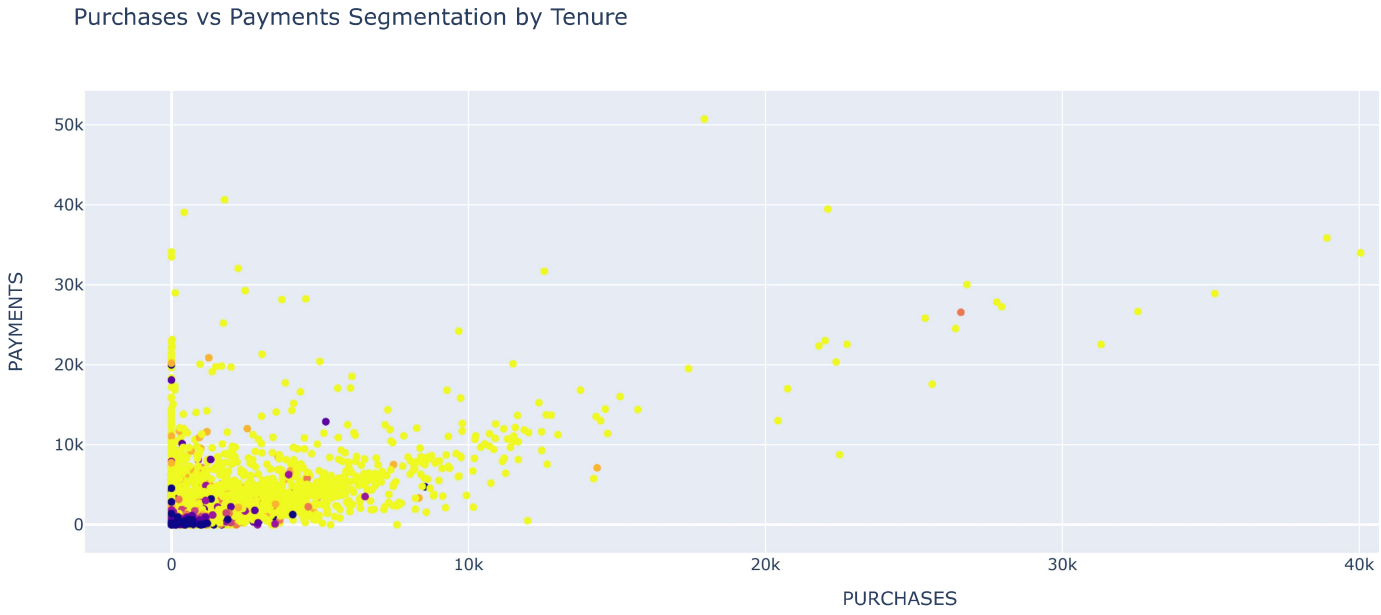

```
#pairwise relationships
fig = px.scatter_matrix(df)
fig.update_layout(title='Pairwise Relationships')
fig.show()
```



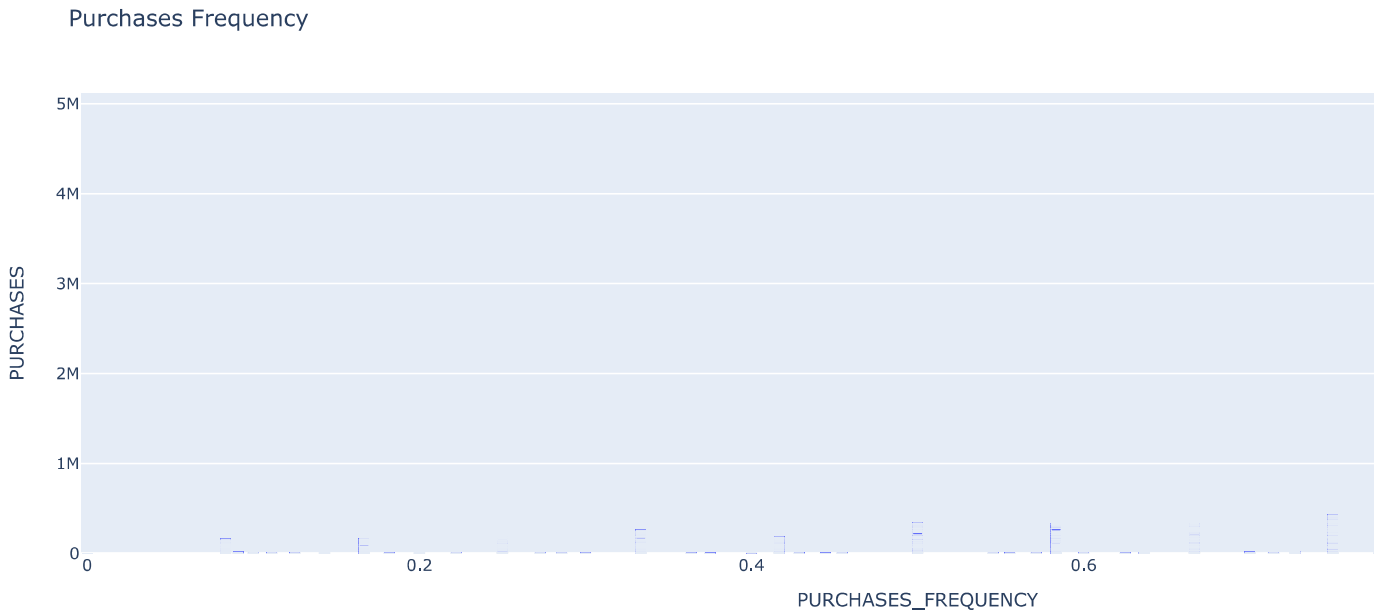

```
#category-based distributions
px.box(df, x='TENURE', y='BALANCE', points="all", title='Boxplot of Balance by Tenure')
```



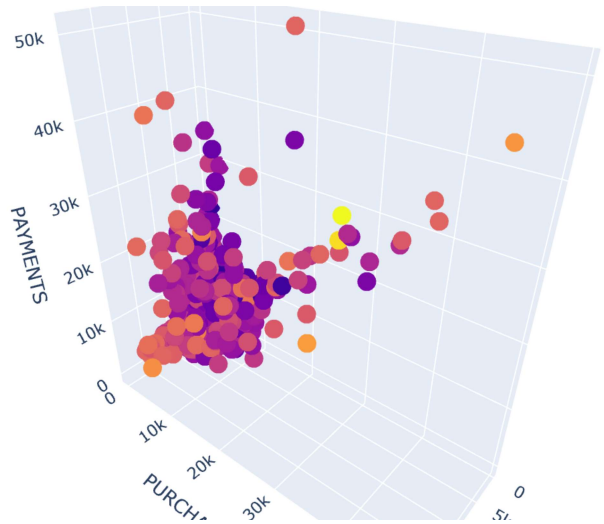

```
1 fig = px.scatter(df, x='PURCHASES', y='PAYMENTS', color='TENURE', title='Purchases vs Payments Segmentation by Tenure')
2 fig.show()
3
```



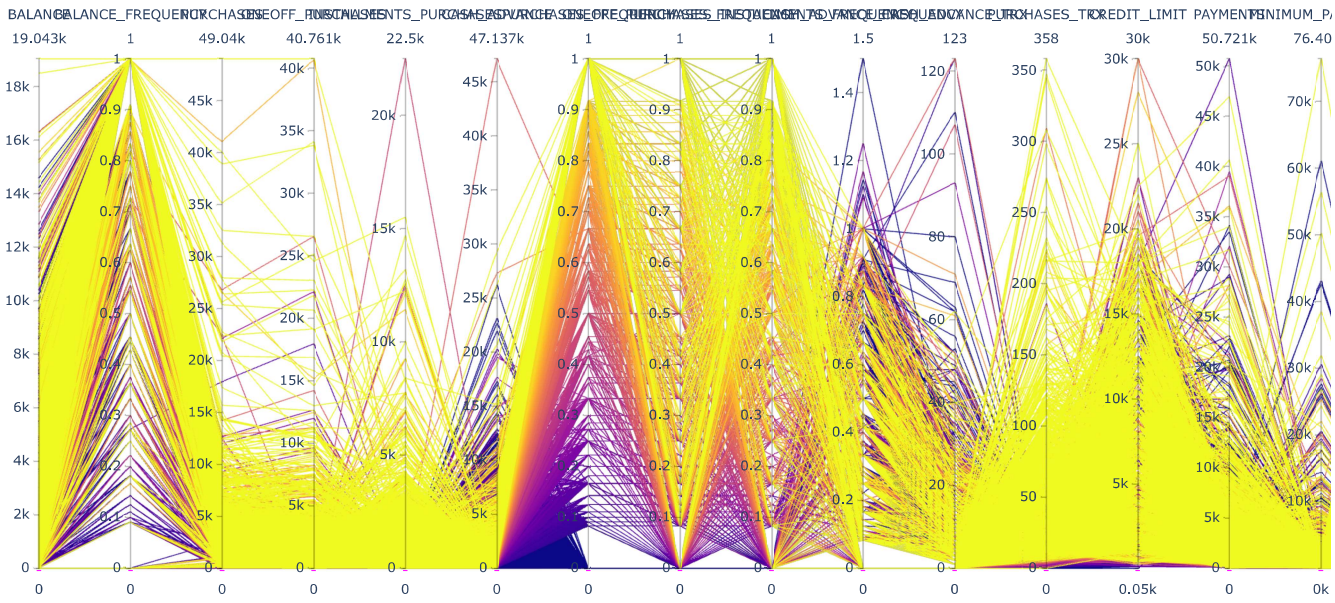

```
1 #frequency analysis of purchase
2 fig = px.bar(df, x='PURCHASES_FREQUENCY', y='PURCHASES', title='Purchases Frequency')
3 fig.show()
4
```




```
# 3d scatter plot
fig = px.scatter_3d(df, x='BALANCE', y='PURCHASES', z='PAYMENTS', color='CREDIT_LIMIT')
fig.show()
```

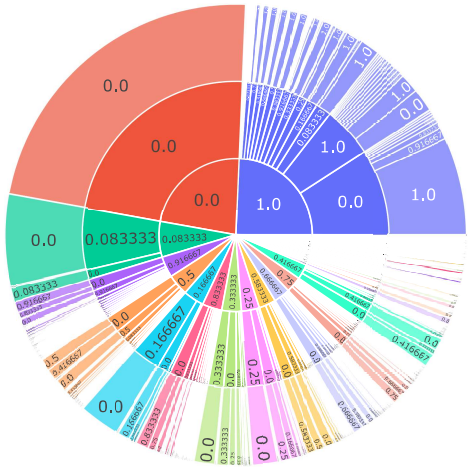



```
# Parallel Coordinates Plot
px.parallel_coordinates(df, color='PURCHASES_FREQUENCY')

```

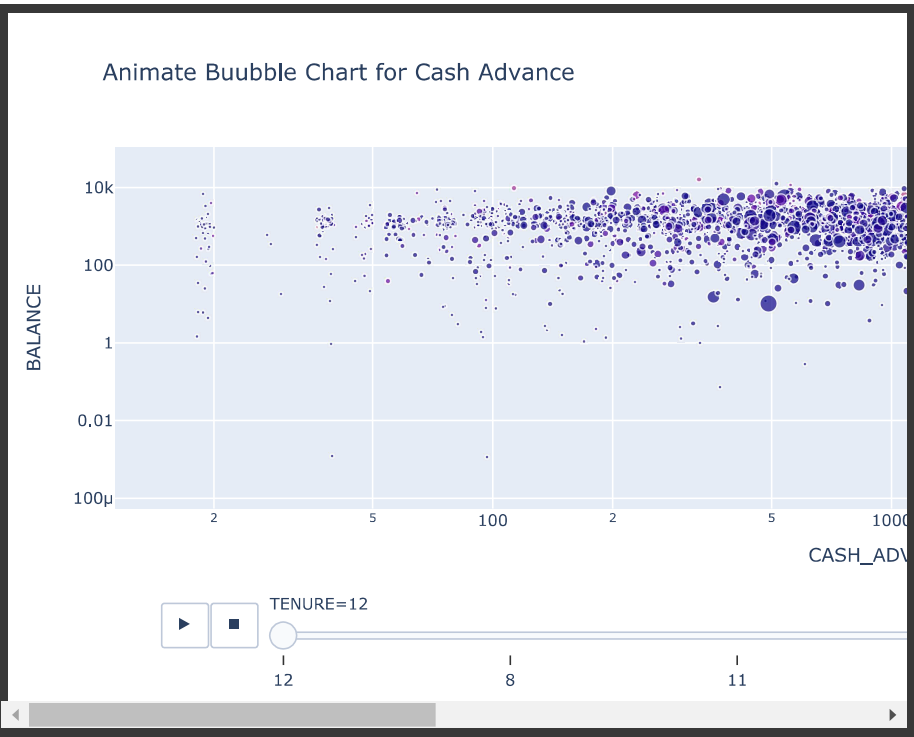



```
1 # Sunburst Chart for Purchase Categories
2 px.sunburst(df, path=['PURCHASES_FREQUENCY', 'ONEOFF_PURCHASES_FREQUENCY', 'PURCHASES_INSTALLMENTS_FREQUENCY'],title='Purchase Categories
3
```

Purchase Categories Sunburst Chart




```
1 px.scatter(df, x='CASH_ADVANCE', y='BALANCE', animation_frame='TENURE', size='CASH_ADVANCE_TRX',
2 color='PURCHASES_TRX', hover_name='CREDIT_LIMIT', log_x=True, log_y=True,
3 title='Animate Buubble Chart for Cash Advance')
```




```
#preproceesing data:
#scaling
#dimention reduction
#outlier reduction
```

```
df=df.drop(['CUST_ID'],axis=1) #dropping cust_id col
```

```
df.max()
```

```
BALANCE                          19043.13856
BALANCE_FREQUENCY                    1.00000
PURCHASES                        49039.57000
ONEOFF_PURCHASES                 40761.25000
INSTALLMENTS_PURCHASES           22500.00000
CASH_ADVANCE                     47137.21176
PURCHASES_FREQUENCY                  1.00000
ONEOFF_PURCHASES_FREQUENCY           1.00000
PURCHASES_INSTALLMENTS_FREQUENCY     1.00000
CASH_ADVANCE_FREQUENCY               1.50000
CASH_ADVANCE_TRX                   123.00000
PURCHASES_TRX                      358.00000
CREDIT_LIMIT                     30000.00000
PAYMENTS                         50721.48336
MINIMUM_PAYMENTS                 76406.20752
PRC_FULL_PAYMENT                     1.00000
TENURE                              12.00000
dtype: float64
```

```
#oulier handling
# from feature_engine.outliers import Winsorizer
# column = ['BALANCE', 'PURCHASES']
# out = Winsorizer(capping_method='gaussian', variables=column)
# df_out = out.fit_transform(df)
```

```
# px.box(df_out,x='TENURE',y='BALANCE')
# px.box(df_out,x='TENURE',y='PURCHASES')
```

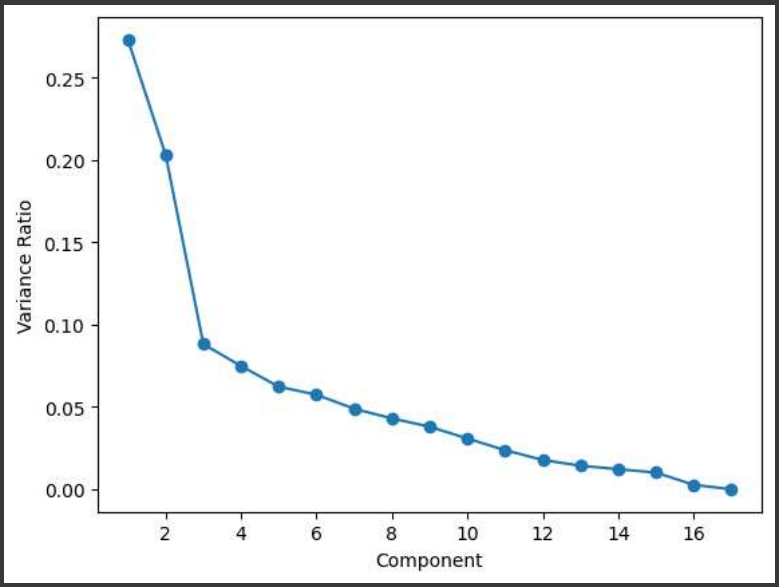
```
from sklearn.preprocessing import StandardScaler
Scaler=StandardScaler()
scaled_df=Scaler.fit_transform(df)
```

```
scaled_df
```

```
array([[-0.73198937, -0.24943448, -0.42489974, ..., -0.3024    ,
        -0.52555097,  0.36067954],
```

```
       [ 0.78696085,  0.13432467, -0.46955188, ...,  0.09749953,
         0.2342269 ,  0.36067954],
       [ 0.44713513,  0.51808382, -0.10766823, ..., -0.0932934 ,
        -0.52555097,  0.36067954],
       ...,
       [-0.7403981 , -0.18547673, -0.40196519, ..., -0.32687479,
         0.32919999, -4.12276757],
       [-0.74517423, -0.18547673, -0.46955188, ..., -0.33830497,
         0.32919999, -4.12276757],
       [-0.57257511, -0.88903307,  0.04214581, ..., -0.3243581 ,
        -0.52555097, -4.12276757]])
```

```
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(scaled_df)

plt.plot(range(1, len(pca.explained_variance_ratio_) + 1), pca.explained_variance_ratio_, marker='o')
plt.xlabel('Component')
plt.ylabel('Variance Ratio')
plt.show()
```



Graph showing aftet 2nd components its decesing aggresively so taking 2 compents or dimention

Double-click (or enter) to edit

```
from sklearn.decomposition import PCA
pca=PCA(n_components=2)
df_pca=pca.fit_transform(scaled_df)
df_pca=pd.DataFrame(data=df_pca,columns=['PCA1','PCA2']) #convrt to dataframe
```

```
df_pca
```

	PCA1	PCA2
0	-1.683648	-1.072245
1	-1.134083	2.509131
2	0.969400	-0.383621
3	-0.888221	0.004656
4	-1.600021	-0.683799
...	...	...
8945	-0.362573	-2.013428
8946	-0.580809	-1.675668
8947	-0.928986	-1.808043
8948	-2.337846	-0.653601
8949	-0.558025	-0.400659

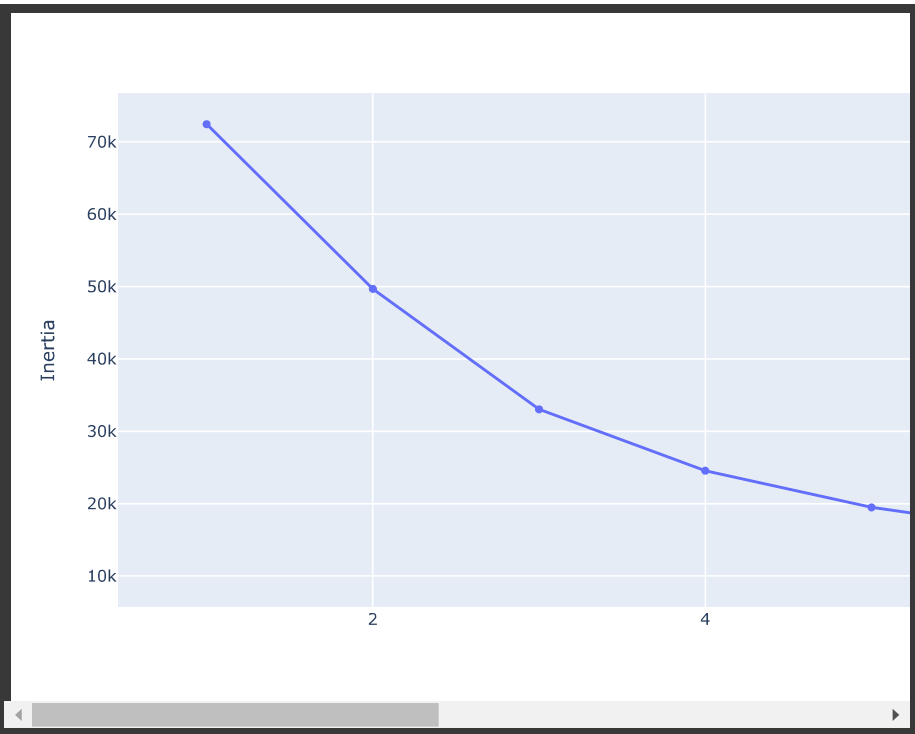
8950 rows × 2 columns

```
#elbow method
from sklearn.cluster import KMeans
```

```
inertia = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(df_pca)
    inertia.append(kmeans.inertia_)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning
```

```
px.line(x=range(1, 11), y=inertia,labels={'x': 'Clusters', 'y': 'Inertia'},markers=True)
```



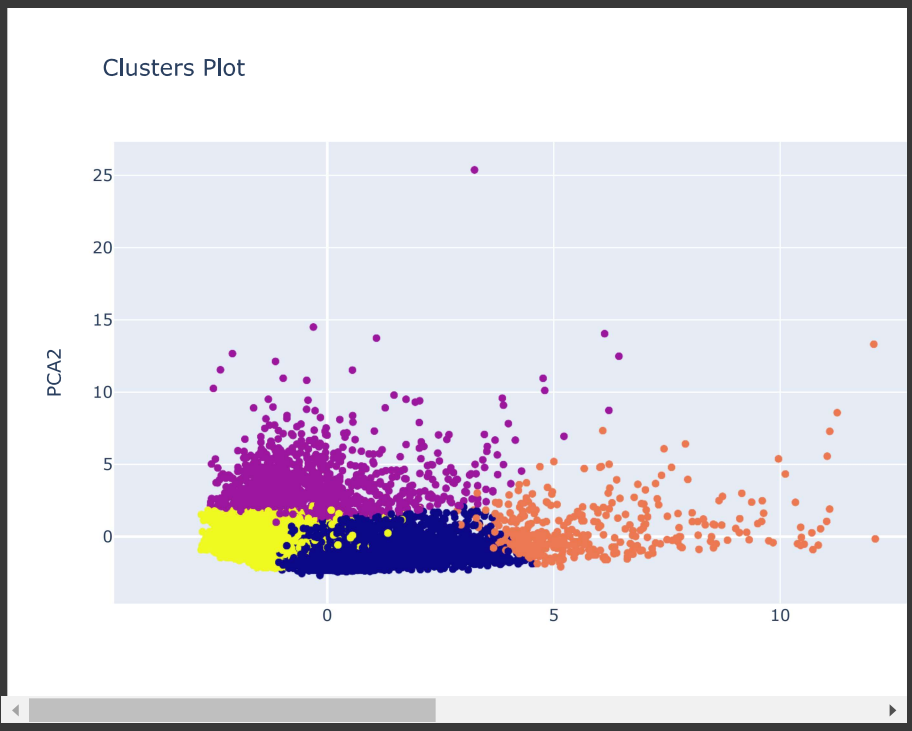

```
1 from sklearn.cluster import KMeans
2 model=KMeans(n_clusters=4,verbose=1)
3 model.fit_predict(scaled_df)
```

```
/usr/local/lib/python3.10/dist-packages/sklearn/cluster/_kmeans.py:870: FutureWarning:

The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init` explicitly to suppress the warning

Initialization complete
Iteration 0, inertia 118402.52236047386.
Iteration 1, inertia 104115.65223269447.
Iteration 2, inertia 102024.85528630813.
Iteration 3, inertia 100421.66210913235.
Iteration 4, inertia 99741.87728768904.
Iteration 5, inertia 99454.03940754272.
Iteration 6, inertia 99312.879860928.
Iteration 7, inertia 99233.76963906389.
Iteration 8, inertia 99185.98670782718.
Iteration 9, inertia 99141.26108193438.
Iteration 10, inertia 99100.50937657384.
Iteration 11, inertia 99078.83731901096.
Iteration 12, inertia 99071.40171378737.
Iteration 13, inertia 99068.63802274797.
Iteration 14, inertia 99066.35527977197.
Iteration 15, inertia 99063.9466335657.
Iteration 16, inertia 99063.174475735.
Iteration 17, inertia 99062.9700867256.
Iteration 18, inertia 99062.55007602685.
Iteration 19, inertia 99062.03598645121.
Iteration 20, inertia 99061.94356361014.
Converged at iteration 20: center shift 4.353272177708942e-06 within tolerance 0.00010000000000000038.
Initialization complete
Iteration 0, inertia 146934.0283840474.
Iteration 1, inertia 115686.9735536788.
Iteration 2, inertia 109148.93418374789.
Iteration 3, inertia 105025.92666941976.
Iteration 4, inertia 101104.25935989176.
Iteration 5, inertia 99776.76481651238.
Iteration 6, inertia 99364.37768258952.
Iteration 7, inertia 99242.90818276088.
Iteration 8, inertia 99174.02815565118.
Iteration 9, inertia 99144.56343742472.
Iteration 10, inertia 99136.49914560965.
Iteration 11, inertia 99128.18729982193.
Iteration 12, inertia 99122.17003194819.
Iteration 13, inertia 99116.86762681027.
Iteration 14, inertia 99113.02025706318.
Iteration 15, inertia 99112.05270328469.
Iteration 16, inertia 99111.55310090627.
Iteration 17, inertia 99110.8939512466.
Iteration 18, inertia 99109.25762375433.
```

```
Iteration 19, inertia 99107.28996368706.
Iteration 20, inertia 99103.97818817204.
Iteration 21, inertia 99101.14999846535.
Iteration 22, inertia 99100.47162182714.
Iteration 23, inertia 99100.24451329923.
Iteration 24, inertia 99100.08731111375.
Iteration 25, inertia 99099.7636830353.
Iteration 26, inertia 99099.17118148514.
Iteration 27, inertia 99098.65327006069.
Iteration 28, inertia 99097.47995238358.
```

```
df_cluster=pd.concat([df_pca,pd.DataFrame({'cluster':model.labels_})],axis=1)
```

```
px.scatter(df_cluster, x='PCA1', y='PCA2', color='cluster',
                title='Clusters Plot')
```




```
cluster_centers = pd.DataFrame(data=model.cluster_centers_,columns=[df.columns])
```

```
cluster_centers = Scaler.inverse_transform(cluster_centers)
cluster_centers = pd.DataFrame(data=cluster_centers,columns=[df.columns])
cluster_df = pd.concat([df,pd.DataFrame({'Cluster':model.labels_})],axis=1)
cluster_df.to_csv("cluster_customer_data.csv")
```

```
cluster_df
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	ONEOFF_PUR(
0	40.900749	0.818182	95.40	0.00	95.40	0.000000	0.166667	
1	3202.467416	0.909091	0.00	0.00	0.00	6442.945483	0.000000	
2	2495.148862	1.000000	773.17	773.17	0.00	0.000000	1.000000	
3	1666.670542	0.636364	1499.00	1499.00	0.00	205.788017	0.083333	
4	817.714335	1.000000	16.00	16.00	0.00	0.000000	0.083333	
...	...	...	...	...	...	...	...	
8945	28.493517	1.000000	291.12	0.00	291.12	0.000000	1.000000	
8946	19.183215	1.000000	300.00	0.00	300.00	0.000000	1.000000	
8947	23.398673	0.833333	144.40	0.00	144.40	0.000000	0.833333	
8948	13.457564	0.833333	0.00	0.00	0.00	36.558778	0.000000	
8949	372.708075	0.666667	1093.25	1093.25	0.00	127.040008	0.666667	

8950 rows × 18 columns

```
#now our data become supervised learning problem(classfication) having parameters and clusters as a target column

```

```
X=cluster_df.drop(['Cluster'],axis=1)
y=cluster_df['Cluster']
```

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=10)
```

```
pip install lazypredict
```

```
Collecting lazypredict
  Downloading lazypredict-0.2.12-py2.py3-none-any.whl (12 kB)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from lazypredict) (8.1.7)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.2.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.5.3)
Requirement already satisfied: tqdm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.66.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.10/dist-packages (from lazypredict) (1.3.2)
Requirement already satisfied: lightgbm in /usr/local/lib/python3.10/dist-packages (from lazypredict) (4.1.0)
Requirement already satisfied: xgboost in /usr/local/lib/python3.10/dist-packages (from lazypredict) (2.0.3)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.23.5)
Requirement already satisfied: scipy in /usr/local/lib/python3.10/dist-packages (from lightgbm->lazypredict) (1.11.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas->lazypredict) (2023.3.post1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn->lazypredict) (3.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.1->pandas->lazypredict) (1
Installing collected packages: lazypredict
Successfully installed lazypredict-0.2.12
```

```
from lazypredict.Supervised import LazyClassifier
clf = LazyClassifier(predictions=True)
models, predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

```
 97%|▉| 28/29 [00:49<00:04,  4.11s/it]
[LightGBM] [Info] Auto-choosing col-wise multi-threading, the overhead of testing was 0.009020 seconds.
You can set `force_col_wise=true` to remove the overhead.
[LightGBM] [Info] Total Bins 2359
[LightGBM] [Info] Number of data points in the train set: 7160, number of used features: 17
[LightGBM] [Info] Start training from score -0.975258
[LightGBM] [Info] Start training from score -2.012462
[LightGBM] [Info] Start training from score -3.074147
[LightGBM] [Info] Start training from score -0.814148
[LightGBM] [Warning] No further splits with positive gain, best gain: -inf
100%|▉| 29/29 [00:53<00:00,  1.86s/it]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score	Time Taken
NearestCentroid	1.00	1.00	None	1.00	0.09
LogisticRegression	0.99	0.98	None	0.99	0.82
SVC	0.98	0.97	None	0.98	3.17
XGBClassifier	0.97	0.96	None	0.97	15.98
LinearSVC	0.96	0.96	None	0.96	1.54
LGBMClassifier	0.97	0.95	None	0.97	4.02
SGDClassifier	0.96	0.95	None	0.96	0.38
Perceptron	0.93	0.94	None	0.93	0.11
RandomForestClassifier	0.96	0.94	None	0.96	5.83
ExtraTreesClassifier	0.96	0.94	None	0.96	1.44
GaussianNB	0.92	0.93	None	0.93	0.11
BaggingClassifier	0.95	0.93	None	0.95	0.60
AdaBoostClassifier	0.93	0.92	None	0.93	0.67
DecisionTreeClassifier	0.93	0.92	None	0.93	0.15
LabelPropagation	0.95	0.92	None	0.95	4.78
LabelSpreading	0.95	0.92	None	0.95	9.94
KNeighborsClassifier	0.95	0.91	None	0.95	0.75
CalibratedClassifierCV	0.95	0.90	None	0.95	2.60
QuadraticDiscriminantAnalysis	0.91	0.90	None	0.91	0.10
PassiveAggressiveClassifier	0.94	0.87	None	0.94	0.13
LinearDiscriminantAnalysis	0.94	0.87	None	0.94	0.19
ExtraTreeClassifier	0.91	0.85	None	0.91	0.04
BernoulliNB	0.84	0.84	None	0.85	0.05
RidgeClassifier	0.90	0.77	None	0.89	0.12
RidgeClassifierCV	0.90	0.77	None	0.89	0.06
DummyClassifier	0.45	0.25	None	0.28	0.03

```
from sklearn.metrics import accuracy_score,classification_report
from sklearn.linear_model import LogisticRegression
lr=LogisticRegression(multi_class='multinomial', solver='lbfgs')
lr.fit(X_train,y_train)
y_predict=lr.predict(X_test)
print(classification_report(y_test, y_predict))
```

```
              precision    recall  f1-score   support

           0       0.85      0.85      0.85       666
           1       0.83      0.76      0.79       240
           2       0.54      0.67      0.60        78
           3       0.88      0.87      0.87       806

    accuracy                           0.84      1790
   macro avg       0.77      0.79      0.78      1790
weighted avg       0.84      0.84      0.84      1790
```

```
from sklearn.svm import SVC
svc=SVC( decision_function_shape='ovr')
```