**DATA MINING**

Logo, company name

Description automatically generated **PROJECT**

*Telco Customer Churn Using K-Modes Clustering*

*Ambika Chundru (*[*ajc7832@psu.edu*](mailto:ajc7832@psu.edu)*)*

*Ashwath Ramesh (*[*akr6021@psu.edu*](mailto:akr6021@psu.edu)*)*

*Rudraksh Mishra (*[*rjm7016@psu.edu*](mailto:rjm7016@psu.edu)*)*

**School of Graduate Professional Studies**

Software Engineering Department

SWENG 545, Section 301: Data Mining

December 2021

**The necessary libraries and modules are imported.**

**import os, sys** – The os module is used to use the Operating system dependent functionality. The sys module is used to operate and handle the python system.

**import numpy as np** – Used to import numpy which is used to perform various powerful mathematical operations.

**import pandas as pd** – Used to import pandas which allows you to handling tables and perform various data analysis techniques.

**import matplotlib.pyplot as plt** – Used to import matplotlib which allows you to use data visualization technique in python. Also imported to use seaborn

**import seaborn as sns** – Used to import seaborn which is a powerful data visualization technique in python.

**import prince** – Used to import prince which is used for multiple correspondence analysis.

**from sklearn.feature\_selection import SelectKBest, chi2** – Used for chi-squared feature selection.

**from sklearn.preprocessing import LabelEncoder, OrdinalEncoder** – Sklearn.preprocessing is a library which is used for encoding categorical data and assign it with numerical values.

**from sklearn.model\_selection import train\_test\_split** – It is used to split the data array into two subsets for training data and for testing data.

**from mlxtend.frequent\_patterns import apriori,**

**association\_rules** – It is used to extract frequent itemset using apriori algorithm in association rule mining.

**from kmodes.kmodes import** **KModes** – It is used for clustering of categorical data.

**from pyod.models.abod import ABOD** – It is used for outlier detection.

**from pyod.models.cblof import CBLOF** – It is used for outlier detection.

**from pyod.models.feature\_bagging import FeatureBagging** - It is used for outlier detection.

**from pyod.models.hbos import HBOS** – It is an unsupervised anomaly detection and is used for outlier detection.

**from pyod.models.iforest import IForest** – It is an Outlier ensemble, outlier detection.

**from pyod.models.knn import KNN** – It is a proximity based, outlier detection.

**from pyod.models.lof import LOF** - It is a proximity based, outlier detection.

**from sklearn.preprocessing import MinMaxScaler** – It is for preprocessing where the data is transformed by scaling each data to a given range.

**from scipy import stats** – It contains a large number of statistics and probability distribution functions.

**import matplotlib.font\_manager** – Used for managing and using various kinds of fonts.

%matplotlib inline

**Importing the data.**

**df = pd.read\_csv('telco.csv')**

**df.columns = [label.lower() for label in df.columns]**

**df.drop(labels=['customerid'], axis=1,inplace=True)**

**dfo=df.copy()**

**df.info()**

The data file ‘telco.csv’ is imported using pandas (pd.read\_csv()) and the column ‘customerid’(df.drop(labels=['customerid'], axis=1,inplace=True)) is dropped since it doesn’t provide us with any important information.

Table

Description automatically generated

**Converting the data to categorical data.**

We use the .replace() here to replace the data with mostly binary data value.

**df['gender'].replace(to\_replace=[['Female'],['Male']],value=[0,1],inplace=True) –** The data is given binary values where the male are given 1 and female are given 0.

**df['partner'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** – Here the ‘Yes’ is given 1 and it means that the person has a partner and ‘No’ is given 0 which means they don’t have a partner.

**df['dependents'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has dependents and ‘No’ is given 0 which means they don’t have a partner.

**df['phoneservice'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has a phone and ‘No’ is given 0 which means they don’t have a phone service.

**df['paperlessbilling'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has paperless billing and ‘No’ is given 0 which means they don’t have a paperless billing.

**df['multiplelines'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True) -** - Here the ‘Yes’ is given 1 and it means that the person has multiple line connections and ‘No’ is given 0 which means they don’t have multiple lines.

**df['onlinesecurity'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has online security and ‘No’ is given 0 which means they don’t have online security.

**df['onlinebackup'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has online backup and ‘No’ is given 0 which means they don’t have online backup.

**df['deviceprotection'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has device protection and ‘No’ is given 0 which means they don’t have device protection.

**df['techsupport'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has tech support and ‘No’ is given 0 which means they don’t have tech support.

**df['streamingtv'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person can stream tv and ‘No’ is given 0 which means they can’t stream tv.

**df['streamingmovies'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person can stream movies and ‘No’ is given 0 which means they can’t stream movies.

**df['internetservice'].replace(to\_replace=[['DSL'],['Fiber optic'],['No']],value=[0,1,2],inplace=True)** – Here the ‘DSL’ is given 0 and it means that the internet service provided is a DSL network, ‘Fiber optic’ is given 1 and it means that the internet service provided is a fiber optic network and ‘No’ is given 2 and it means that they have no internet service.

**df['paymentmethod'].replace(to\_replace=[['Electronic check'],['Mailed check'],['Bank transfer (automatic)'],['Credit card (automatic)']],value=[0,0,1,1],inplace=True)** – Here the ‘Electronic check’ is given 0 since it’s not automatic and it means that the person’s payment method is via an electronic check, ‘Mailed check’ is also given 0 since it also is not automatic and it means that the person’s payment method is via a mailed check, ‘Bank transfer’ is given 1 since it’s automatic and it means that the person’s payment method is via a bank transfer, and Credit cart is also given 1 since it also is automatic and it means that the person’s payment method is using a credit card.

**df['contract'].replace(to\_replace=[['Month-to-month'],['One year'], ['Two year']],value=[0,1,2],inplace=True)** – Here the ‘month-to-month’ is given 0 and it means that the person has a monthly based contract, ‘One year’ is given 1 and it means that the person has a yearly contract, and ‘two year’ is given 2 and it means that the person has a two year contract.

**df['churn'].replace(to\_replace=[['No'],['Yes']],value=[0,1],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person churn and ‘No’ is given 0 which means they don’t churn.

**df['multiplelines'].replace(to\_replace=[['Yes'],['No phone service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has multiple lines and both ‘No Phone service’ and ‘No’ are given 0 which means they either don’t have multiple lines or they don’t have a phone service.

**df['onlinesecurity'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has online security and both ‘No Internet service’ and ‘No’ are given 0 which means they either don’t have online security or they don’t have an internet service.

**df['onlinebackup'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has online backup and both ‘No Internet service’ and ‘No’ are given 0 which means they either don’t have online backup or they don’t have an internet service.

**df['deviceprotection'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has online device protection and both ‘No Internet service’ and ‘No’ are given 0 which means they either don’t have device protection or they don’t have an internet service.

**df['techsupport'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person has tech support and both ‘No Internet service’ and ‘No’ are given 0 which means they either don’t have tech support or they don’t have an internet service.

**df['streamingtv'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person can stream tv and both ‘No Internet service’ and ‘No’ are given 0 which means they either can’t stream tv or they don’t have an internet service.

**df['streamingmovies'].replace(to\_replace=[['Yes'],['No internet service'],['No']],value=[1,0,0],inplace=True)** - Here the ‘Yes’ is given 1 and it means that the person can stream movies and both ‘No Internet service’ and ‘No’ are given 0 which means they either can’t stream movies or they don’t have an internet service.

A picture containing text, light, screenshot, document

Description automatically generated

**Dropping Unwanted Rows.**

When tenure is 0 the corresponding row is dropped and also the rows where monthy charges column is empty using .drop().

**df.drop(labels=df[df['tenure'] == 0].index, axis=0, inplace=True)**

**df.drop(labels=['totalcharges'], axis=1,inplace=True)**

**df**

**df1=df.copy()**

**df2=df.copy()**

**dfl=df.copy()**

**df2**

A picture containing text, screenshot, light, document

Description automatically generated

**df.isnull().sum().sum()-** It is used to see the number of missing values and the output returned was 0 and hence there were no missing values.

**df.isna().sum().sum() –** It is used to see the number of missing values in a column and the output returned was 0 and hence we can see that there were no missing values

**df.describe().T –** Used to get the mean, count, standard deviation, and the quartiles(min, q1, q2, q3, max)

Table

Description automatically generated

**df.corr().T** – Used to get the pairwise correlation matrix of all the columns.

Table, Excel

Description automatically generated

**Converting the tenure and monthly charges to categorical data**

A function tenure() is defined which states that if the given data is greater than 0 and less than or equal to 32 then its value is 0, if else it is 1. Then this function is applied to the tenure column in df using .apply(). Similarly a charges() function is defined which states that if the given data is greater than 0 and less than or equal to 70 then its value is 0, if else it is 1. Then this function is applied to the monthlycharges column in df using .apply().

**def tenure(data):**

**if 0 < data <= 32 :**

**a=0**

**return a**

**else:**

**b=1**

**return b**

**df['tenure'] = (df['tenure'].apply(tenure))**

**def charges(data):**

**if 0 < data <= 70 :**

**c=0**

**return c**

**else:**

**d=1**

**return d**

**df['monthlycharges'] = df['monthlycharges'].apply(charges)**

**Outlier detection.**

**scaler = MinMaxScaler(feature\_range=(0, 1))**

**dfo = dfo.astype({ 'tenure': 'float','monthlycharges': 'float'})** – The type of tenure and monthlycharges is set as float

**dfo[['tenure','monthlycharges']]=scaler.fit\_transform(dfo[['tenure','monthlycharges']]) –** the min max scaler model is fit into the dfo.

**X1 = dfo['tenure'].values.reshape(-1,1) –** the values tenure column of dfo are reshaped into the (-1,1) since the original data is 1-D, -1 had to be used.

**X2 = dfo['monthlycharges'].values.reshape(-1,1) –** Similarly, the monthlycharges column is also reshaped.

**X = np.concatenate((X1,X2),axis=1) –** Using np.concatenate() the X1 and X2 are combined.

**X**

**random\_state = np.random.RandomState(42)**

**outliers\_fraction = 0.06**

**classifiers={ 'IsolationForest':IForest(contamination=outliers\_fraction,random\_state=random\_state),'Angle-based Outlier Detector (ABOD)': ABOD(contamination=outliers\_fraction)**

**}**

**for i, (clf\_name, clf) in enumerate(classifiers.items()):**

**clf.fit(X)**

**scores\_pred = clf.decision\_function(X) \* -1**

**y\_pred = clf.predict(X)**

**n\_inliers = len(y\_pred) - np.count\_nonzero(y\_pred)**

**n\_outliers = np.count\_nonzero(y\_pred == 1)**

**plt.figure(figsize=(30, 30))**

**dfx = dfo**

**dfx['outlier'] = y\_pred.tolist()**

**IX1 = np.array(dfx['tenure'][dfx['outlier'] == 0]).reshape(-1,1)**

**IX2 = np.array(dfx['monthlycharges'][dfx['outlier'] == 0]).reshape(-1,1)**

**OX1 = dfx['tenure'][dfx['outlier'] == 1].values.reshape(-1,1)**

**OX2 = dfx['monthlycharges'][dfx['outlier'] == 1].values.reshape(-1,1)**

**print('outliers in this model:',n\_outliers,'normal points:',n\_inliers, clf\_name)**

**threshold = stats.scoreatpercentile(scores\_pred,100 \* outliers\_fraction)**

**Z = clf.decision\_function(np.c\_[xx.ravel(), yy.ravel()]) \* -1**

**Z = Z.reshape(xx.shape)**

**plt.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 5),c='yellow')**

**a = plt.contour(xx, yy, Z, levels=[threshold],linewidths=5, colors='red')**

**plt.contourf(xx, yy, Z, levels=[threshold, Z.max()],colors='green')**

**b = plt.scatter(IX1,IX2, c='white',s=30, edgecolor='k')**

**c = plt.scatter(OX1,OX2, c='black',s=30, edgecolor='k')**

**plt.axis('tight')**

**plt.legend(**

**[a.collections[0], b,c],['learned decision function', 'inliers','outliers'], prop=matplotlib.font\_manager.FontProperties(size=45),**

**loc=2)**

**plt.xlim((0, 1))**

**plt.ylim((0, 1))**

**plt.title(clf\_name)**

**plt.show()**

**Checking for the number of unique values.**

**df.nunique()-** Returns the number of unique values and the output is 2 other than for internetservice and contract which has 3.

Text

Description automatically generated with medium confidence

**Chi2 Feature Extraction.**

The test has been performed to test the dependency of the churn based on the other factors.We use the chi2 value to determine this.Higher the chi2 value the more the churn is dependent on that factor.Here churn is the target variable, and the rest are the input features.

**X1 = df.drop(['churn'], axis=1) # input features**

**y1 = df['churn'] # target variable**

Since churn is our target variable, everything other than churn is stored in X1 and churn is stored in Y1.

**oe = OrdinalEncoder()**

**oe.fit(X1)**

**X\_enc = oe.transform(X1)**

**le = LabelEncoder()**

**le.fit(y1)**

**y\_enc = le.transform(y1)**

Ordinal encoded array of X1 is created and stored in X\_enc.

Similarly Using label encoder and array Y\_enc is created and stores the value of the encoded array of Y1.

**sf = SelectKBest(chi2, k=10)**

**sf\_fit1 = sf.fit(X1, y1)**

**for i in range(len(sf\_fit1.scores\_)):**

**print(' %s: %f' % (X1.columns[i], sf\_fit1.scores\_[i]))**

Text

Description automatically generated

We perform feature extraction using SelectKBest by assigning k = 10 and use chi2 and print the result.

**x\_new=sf.fit\_transform(X\_enc, y\_enc)**

**cols = sf.get\_support(indices=True)**

**dff = X1.iloc[:,cols]**

**dff['churn'] = df1['churn'].values**

**dff**

We use the obtained value and assign dff based on the top 10 chi2 value columns and churn column.

A picture containing text, light, screenshot, document

Description automatically generated

**datset1 = pd.DataFrame()**- we create a dataframe

**datset1['feature'] = X1.columns[ range(len(sf\_fit1.scores\_))]** – we assign the 10 feature columns to the dataframe

**datset1['scores'] = sf\_fit1.scores\_** - we assign the corresponding chi2 scores to the features

**datset1 = datset1.sort\_values(by='scores', ascending=True)** – we sort the dataset in the ascending order

**sns.barplot(datset1['scores'], datset1['feature'], color='green')**

**sns.set\_style('whitegrid')**

**plt.ylabel('Categorical feature', fontsize=18)**

**plt.xlabel('Score', fontsize=18)**

**plt.show()**

Then by using sns.barplot() we create the barplot of the result of the chi2 feature extraction in ascending order and label it accordingly.

Chart, bar chart, histogram

Description automatically generated

**Association Rule.**

Association rule mining being done to find interesting frequent item sets. We can use the findings to better predict if the customer will churn or not. We use the apriori algorithm to find the frequent itemsets with a minimum support of 0.9 as the threshold. Association rules has been done with lift as 1 as the minimum threshold.

**df\_1=df2[df2['churn']==1]**

**df\_0=df2[df2['churn']==0]**

**df\_1**

We define df\_1 and give it the value of ‘df’ but where churn = 1.

Similarly, df\_2 is defined and is given the value of ‘df’ where churn = 0.

A picture containing text, light, document, screenshot

Description automatically generated

**def hot\_encode(x):**

**if(x<= 0):**

**return 0**

**if(x>= 1):**

**return 1**

- The function hot\_encode() is defined where if x is <= 0 it returns 0 and if x>= 1 then it returns 1

**df\_encoded = df.applymap(hot\_encode)** – The function hotencode() is applied to each of the element of ‘df’ using .applymap() and stored in ‘df\_encoded’

**df\_0\_encoded= df\_0.applymap(hot\_encode)** – Similarly, the function hotencode() is applied to each of the element of ‘df\_0’ using .applymap() and stored in ‘df\_0\_encoded’

**df\_1\_encoded = df\_1.applymap(hot\_encode)** – Similarly, the function hotencode() is applied to each of the element of ‘df\_1’ using .applymap() and stored in ‘df\_1\_encoded’

**frq\_items = apriori(df\_1\_encoded, min\_support = 0.9, use\_colnames = True)** – frequent itemsets in df\_1 is found using apriori().

**rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)** – rules are generated using association\_rules() which uses the frequent items gernerated and uses lift as the metric with a minimum threshold of 1 where the lift has to be higher than or equal to 1.

**rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])** – The rules are sorted in descending order on the basis of confidence and lift.

**freq\_items=frq\_items.sort\_values(['support'], ascending=False)** – The frequent itemset is sorted based on support in descending order.

**freq\_items**

Table

Description automatically generated

The same process is again performed for churn of 0

**frq\_items = apriori(df\_0\_encoded, min\_support = 0.90, use\_colnames = True)**

**rules = association\_rules(frq\_items, metric ="lift", min\_threshold = 1)**

**rules = rules.sort\_values(['confidence', 'lift'], ascending =[False, False])**

**freq\_items=frq\_items.sort\_values(['support'], ascending=False)**

**freq\_items**

**dff**

Table

Description automatically generated

**k++ Elbow Method.**

**cost = []**

**K = range(1,10)**

**for num\_clusters in list(K):**

**kmode = KModes(n\_clusters=num\_clusters, init = "random", n\_init = 5, verbose=1)**

**kmode.fit\_predict(df)**

**cost.append(kmode.cost\_)**

Text

Description automatically generated

Since the data is categorical we use Kmodes clustering. We use Kmodes() in a loop to run multiple iterations where the number of random centroid seeds is 5. The number of clusters is based on the K which ranges from 1 to 10. So, for every loop the n\_clusters changes from 1 till 10.

**plt.plot(K, cost, 'bx-')**

**plt.xlabel('No. of clusters')**

**plt.ylabel('Cost')**

**plt.title('Elbow Method For Optimal k')**

**plt.show()**

Chart, line chart

Description automatically generated

We use matplotlib.pyplot library to visualize the K++ Elbow method where the imported matplotlib.pyplot is saved as plt. Plt.plot() is used and labels are given using plt.xlabel() and plt.ylabel(). The title is given using plt.title(). We then visualize it using plt.show()

**Another Method**

**costs = []**

**for cluster in range(1, 10):**

**kmodes = KModes(n\_jobs = -1, n\_clusters = cluster, init = 'Huang', random\_state = 0)**

**kmodes.fit\_predict(dff)**

**costs.append(kmodes.cost\_)**

**print('Cluster initiation: {}'.format(cluster))**

Text

Description automatically generated

Here in Kmodes() we use n\_jobs parameter is used and is given a value of -1 which means parallel computing is done. It also uses init = ‘Huang’.

**plt.plot(K, costs, 'bx-')**

**plt.xlabel('No. of clusters')**

**plt.ylabel('Cost')**

**plt.title('Elbow Method For Optimal k')**

**plt.show()**

Chart, line chart

Description automatically generated

Similar to the previous method, the result is plotted and from the resulting graph we can find the optimum value of K.

**Silhouette Method.**

**from sklearn.metrics import silhouette\_score**

**K = range(2,10)**

**score\_list=[]**

**for n\_clusters in range(2,10):**

**clusterer = KModes (n\_clusters=n\_clusters).fit(df)**

**preds = clusterer.predict(df)**

**centers = clusterer.cluster\_centroids\_**

**score = silhouette\_score (df, preds, metric='euclidean')**

**score\_list.append(score)**

**print ("For n\_clusters = {}, silhouette score is {})".format(n\_clusters, score))**

Text, letter

Description automatically generated

Another method is to calculate the silhouette score to find the optimum K value. Here we define K from a range of 2 – 10.

We create a for loop for n\_clusters in K. In the loop we use KMode() and we use that value and run it with .predict() to find the predicted value using ‘df’ as parameter. Then the silhouette score is calculated using ‘df’, predicted value as parameters and metric is set at default at ‘euclidean’. The score returned as output.

**plt.bar(K,score\_list)**

**plt.show()**

The result is then plotted in form of a bar chart using plt.bar().

Chart, bar chart

Description automatically generated

**KMode.**

**kmodes = KModes(n\_jobs = -1, n\_clusters = 3, init = 'Huang', random\_state = 0) ­**– Kmode() is run with n\_jobs = -1 which means parallel processing occurs. N\_clusters = 3 which was found out to be optimum and init = ‘Huang’ and random state = 0 which means seed generation of centralization is random.

**kmodes.fit\_predict(dff)** - .fir\_predict() is used to compute cluster centers and predict cluster index for each sample of the ‘dff’ dataset.

**kmodes.cluster\_centroids\_** – Cluster centroids is found using this.

**kmodes.n\_iter\_** - Used to check the iterations of the cluster created.

**kmodes.cost\_ -** check the cost of the clusters created.

**labels = kmodes.labels\_ -** Lable of the clusters is created and the three clusters are given the label of (0,1,2)

**dff['Cluster Labels'] = kmodes.labels\_ -** Clusters are added to the dataframe after being labeled.

**Dff**

Table

Description automatically generated

**dfl['Cluster Labels'] = kmodes.labels\_**

A picture containing text, screenshot, white, light

Description automatically generated

**Multiple Correspondence Analysis(MCA).**

**for col in ['seniorcitizen',**

**'partner',**

**'dependents',**

**'tenure',**

**'onlinesecurity',**

**'techsupport',**

**'contract',**

**'paperlessbilling',**

**'paymentmethod',**

**'monthlycharges',**

**'churn','Cluster Labels']:**

**dff[col] = dff[col].astype('category')**

A for loop is run so that all the columns in ‘dff’ has the dtype of ‘category’

**from prince import MCA –** MCA is imported

**mca = MCA(n\_components = 2, n\_iter = 3, random\_state = 101)** – MCA() is used.

**mca.fit(dff)**

**dff\_mca = mca.transform(dff)**

You use .fit() and .transform() to get the result

**dff\_mca.head()**

Table

Description automatically generated

**mca.plot\_coordinates(X = dff,legend\_n\_cols=1).legend(loc='center left', bbox\_to\_anchor=(1, 0.5)) –** The result is plotted.

Chart, scatter chart

Description automatically generated

**d = dfl[dfl['Cluster Labels'] == 1]**

**d**

A picture containing text, white, screenshot, light

Description automatically generated

**Plotting the Results for k-modes .**

**colors = ['#eb8034', '#34baeb', '#49eb34']**

**markers = ['^', 'o', 'd']**

**dfl['mca0']=dff\_mca[0]**

**dfl['mca1']=dff\_mca[1]**

**dfl**

**for c in dfl['Cluster Labels'].unique():**

**d = dfl[dfl['Cluster Labels'] == c]**

**rgb = (np.random.rand(3,))**

**plt.scatter(d['mca0'], d['mca1'], marker=markers[c], color=colors[c])**

**#plt.show()**

Chart, scatter chart

Description automatically generated