DATA MINING PROJECT

Telco Customer Churn Using K-Modes Clustering

Ambika Chundru (ajc7832@psu.edu)

Ashwath Ramesh (akr6021@psu.edu)

Rudraksh Mishra (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.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 7043 entries, 0 to 7042
Data columns (total 20 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   gender            7043 non-null   object
 1   seniorcitizen     7043 non-null   int64 
 2   partner           7043 non-null   object
 3   dependents        7043 non-null   object
 4   tenure            7043 non-null   int64 
 5   phoneservice      7043 non-null   object
 6   multiplelines     7043 non-null   object
 7   internetservice   7043 non-null   object
 8   onlinesecurity    7043 non-null   object
 9   onlinebackup      7043 non-null   object
 10  deviceprotection  7043 non-null   object
 11  techsupport       7043 non-null   object
 12  streamingtv       7043 non-null   object
 13  streamingmovies   7043 non-null   object
```

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.

	gender	seniorcitizen	partner	dependents	tenure	phoneservice	multiplelines	internetservice	onlinesecurity	onlinebackup	deviceprotection	techsu
0	0	0	1	0	1	0	0	0	0	1	0	
1	1	0	0	0	34	1	0	0	1	0	1	
2	1	0	0	0	2	1	0	0	1	1	0	
3	1	0	0	0	45	0	0	0	1	0	1	
4	0	0	0	0	2	1	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	1	0	1	1	24	1	1	0	1	0	1	
7039	0	0	1	1	72	1	1	1	0	1	1	
7040	0	0	1	1	11	0	0	0	1	0	0	
7041	1	1	1	0	4	1	1	1	0	0	0	
7042	1	0	0	0	66	1	0	1	1	0	1	

7032 rows × 19 columns

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()
df1=df.copy()
df2
```

	gender	seniorcitizen	partner	dependents	tenure	phoneservice	multiplelines	internetservice	onlinesecurity	onlinebackup	deviceprotection	techsu
0	0	0	1	0	1	0	0	0	0	1	0	
1	1	0	0	0	34	1	0	0	1	0	1	
2	1	0	0	0	2	1	0	0	1	1	0	
3	1	0	0	0	45	0	0	0	1	0	1	
4	0	0	0	0	2	1	0	1	0	0	0	
...	...	...	...	...	...	...	...	...	...	...	...	
7038	1	0	1	1	24	1	1	0	1	0	1	
7039	0	0	1	1	72	1	1	1	0	1	1	
7040	0	0	1	1	11	0	0	0	1	0	0	
7041	1	1	1	0	4	1	1	1	0	0	0	
7042	1	0	0	0	66	1	0	1	1	0	1	

7032 rows × 19 columns

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)

Out[41]:

	count	mean	std	min	25%	50%	75%	max
gender	7032.0	0.504693	0.500014	0.00	0.0000	1.00	1.0000	1.00
seniorcitizen	7032.0	0.162400	0.368844	0.00	0.0000	0.00	0.0000	1.00
partner	7032.0	0.482509	0.499729	0.00	0.0000	0.00	1.0000	1.00
dependents	7032.0	0.298493	0.457629	0.00	0.0000	0.00	1.0000	1.00
tenure	7032.0	32.421786	24.545260	1.00	9.0000	29.00	55.0000	72.00
phoneservice	7032.0	0.903299	0.295571	0.00	1.0000	1.00	1.0000	1.00
multiplelines	7032.0	0.421928	0.493902	0.00	0.0000	0.00	1.0000	1.00
internetservice	7032.0	0.872582	0.737271	0.00	0.0000	1.00	1.0000	2.00
onlinesecurity	7032.0	0.286547	0.452180	0.00	0.0000	0.00	1.0000	1.00
onlinebackup	7032.0	0.344852	0.475354	0.00	0.0000	0.00	1.0000	1.00
deviceprotection	7032.0	0.343857	0.475028	0.00	0.0000	0.00	1.0000	1.00
techsupport	7032.0	0.290102	0.453842	0.00	0.0000	0.00	1.0000	1.00

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

	gender	seniorcitizen	partner	dependents	tenure	phoneservice	multiplelines	internetservice	onlinesecurity	onlinebackup	devi
gender	1.000000	-0.001819	-0.001379	0.010349	0.005285	-0.007515	-0.008883	-0.002236	-0.016328	-0.013093	
seniorcitizen	-0.001819	1.000000	0.016957	-0.210550	0.015683	0.008392	0.142996	-0.032160	-0.038576	0.066663	
partner	-0.001379	0.016957	1.000000	0.452269	0.381912	0.018397	0.142561	0.000513	0.143346	0.141849	
dependents	0.010349	-0.210550	0.452269	1.000000	0.163386	-0.001078	-0.024307	0.044030	0.080786	0.023639	
tenure	0.005285	0.015683	0.381912	0.163386	1.000000	0.007877	0.332399	-0.029835	0.328297	0.361138	
phoneservice	-0.007515	0.008392	0.018397	-0.001078	0.007877	1.000000	0.279530	0.387266	-0.091676	-0.052133	
multiplelines	-0.008883	0.142996	0.142561	-0.024307	0.332399	0.279530	1.000000	0.011346	0.098592	0.202228	
internetservice	-0.002236	-0.032160	0.000513	0.044030	-0.029835	0.387266	0.011346	1.000000	-0.392174	-0.313708	
onlinesecurity	-0.016328	-0.038576	0.143346	0.080786	0.328297	-0.091676	0.098592	-0.392174	1.000000	0.283285	
onlinebackup	-0.013093	0.066663	0.141849	0.023639	0.361138	-0.052133	0.202228	-0.313708	0.283285	1.000000	
deviceprotection	-0.000807	0.059514	0.153556	0.013900	0.361520	-0.070076	0.201733	-0.305757	0.274875	0.303058	
techsupport	-0.008507	-0.060577	0.120206	0.063053	0.325288	-0.095138	0.100421	-0.388535	0.354458	0.293705	
streamingtv	-0.007124	0.105445	0.124483	-0.016499	0.280264	-0.021383	0.257804	-0.241330	0.175514	0.281601	
streamingmovies	-0.010105	0.119842	0.118108	-0.038375	0.285402	-0.033477	0.259194	-0.250144	0.187426	0.274523	
contract	0.000095	-0.141820	0.294094	0.240556	0.676734	0.003019	0.107529	0.099579	0.245660	0.155262	
paperlessbilling	-0.011902	0.156258	-0.013957	-0.110131	0.004823	0.016696	0.163746	-0.138166	-0.004051	0.127056	
paymentmethod	-0.011974	-0.033775	0.161327	0.094464	0.396772	0.001159	0.113030	-0.040351	0.174631	0.147661	

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.

```
gender              2
seniorcitizen       2
partner             2
dependents          2
tenure              2
phoneservice        2
multiplelines       2
internetservice     3
onlinesecurity      2
onlinebackup        2
deviceprotection    2
techsupport         2
streamingtv         2
streamingmovies     2
contract            3
paperlessbilling    2
paymentmethod       2
monthlycharges      2
churn               2
dtype: int64
```

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]))
```

```
gender: 0.254297
seniorcitizen: 133.482766
partner: 81.857769
dependents: 131.271509
tenure: 331.204748
phoneservice: 0.092948
multiplelines: 6.514651
internetservice: 9.715269
onlinesecurity: 147.165601
onlinebackup: 31.209832
deviceprotection: 20.216007
techsupport: 135.439602
streamingtv: 17.320615
streamingmovies: 15.930611
contract: 1111.759054
paperlessbilling: 104.979224
paymentmethod: 175.733987
monthlycharges: 142.193735
```

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.

	seniorcitizen	partner	dependents	tenure	onlinesecurity	techsupport	contract	paperlessbilling	paymentmethod	monthlycharges	churn
0	0	1	0	0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	1	0	0	0	0
2	0	0	0	0	1	0	0	1	0	0	1
3	0	0	0	1	1	1	1	0	1	0	0
4	0	0	0	0	0	0	0	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
7038	0	1	1	0	1	1	1	1	0	1	0
7039	0	1	1	1	0	0	1	1	1	1	0
7040	0	1	1	0	1	0	0	1	0	0	0
7041	1	1	0	0	0	0	0	1	0	1	1
7042	0	0	0	1	1	1	2	1	1	1	0

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

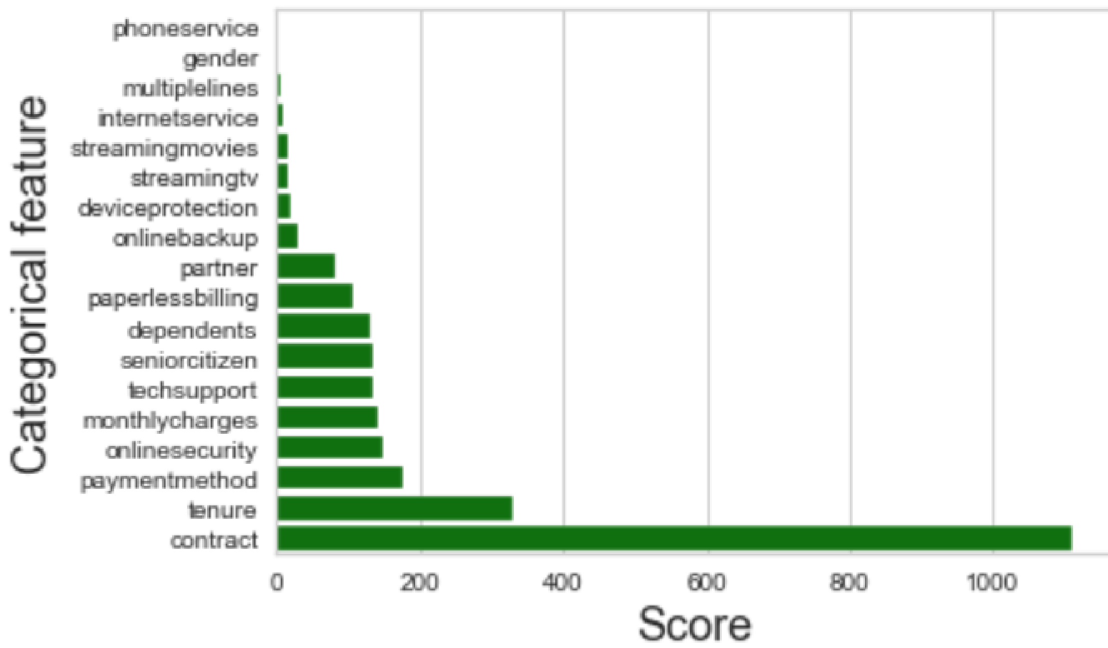
datset1['feature'] = X1.columns[range(len(sf_fit1.scores_))] - we assign the 10 feature columns to the dataframe

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.



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.

	gender	seniorcitizen	partner	dependents	tenure	phoneservice	multiplelines	internetservice	onlinesecurity	onlinebackup	deviceprotection	techsu
2	1	0	0	0	2	1	0	0	1	1	0	
4	0	0	0	0	2	1	0	1	0	0	0	
5	0	0	0	0	8	1	1	1	0	0	1	
8	0	0	1	0	28	1	1	1	0	0	1	
13	1	0	0	0	49	1	1	1	0	1	1	
...	...	...	...	...	...	...	...	...	...	...	...	
7021	1	0	0	0	12	1	0	0	0	0	0	
7026	0	0	0	0	9	1	0	0	0	0	0	
7032	1	1	0	0	1	1	1	1	0	0	0	
7034	0	0	0	0	67	1	1	1	1	1	1	
7041	1	1	1	0	4	1	1	1	0	0	0	

```
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

	support	itemsets
0	1.000000	(tenure)
2	1.000000	(monthlycharges)
3	1.000000	(churn)
5	1.000000	(tenure, monthlycharges)
6	1.000000	(churn, tenure)
9	1.000000	(churn, monthlycharges)
12	1.000000	(churn, tenure, monthlycharges)
1	0.909042	(phoneservice)
4	0.909042	(tenure, phoneservice)
7	0.909042	(monthlycharges, phoneservice)
8	0.909042	(churn, phoneservice)
10	0.909042	(tenure, monthlycharges, phoneservice)
11	0.909042	(churn, tenure, phoneservice)
13	0.909042	(churn, monthlycharges, phoneservice)
14	0.909042	(churn, tenure, monthlycharges, phoneservice)

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
```

	seniorcitizen	partner	dependents	tenure	onlinesecurity	techsupport	contract	paperlessbilling	paymentmethod	monthlycharges	churn
0	0	1	0	0	0	0	0	1	0	0	0
1	0	0	0	1	1	0	1	0	0	0	0
2	0	0	0	0	1	0	0	1	0	0	1
3	0	0	0	1	1	1	1	0	1	0	0
4	0	0	0	0	0	0	0	1	0	1	1
...	...	...	...	...	...	...	...	...	...	...	...
7038	0	1	1	0	1	1	1	1	0	1	0
7039	0	1	1	1	0	0	1	1	1	1	0
7040	0	1	1	0	1	0	0	1	0	0	0
7041	1	1	0	0	0	0	0	1	0	1	1
7042	0	0	0	1	1	1	2	1	1	1	0

k++ Elbow Method.

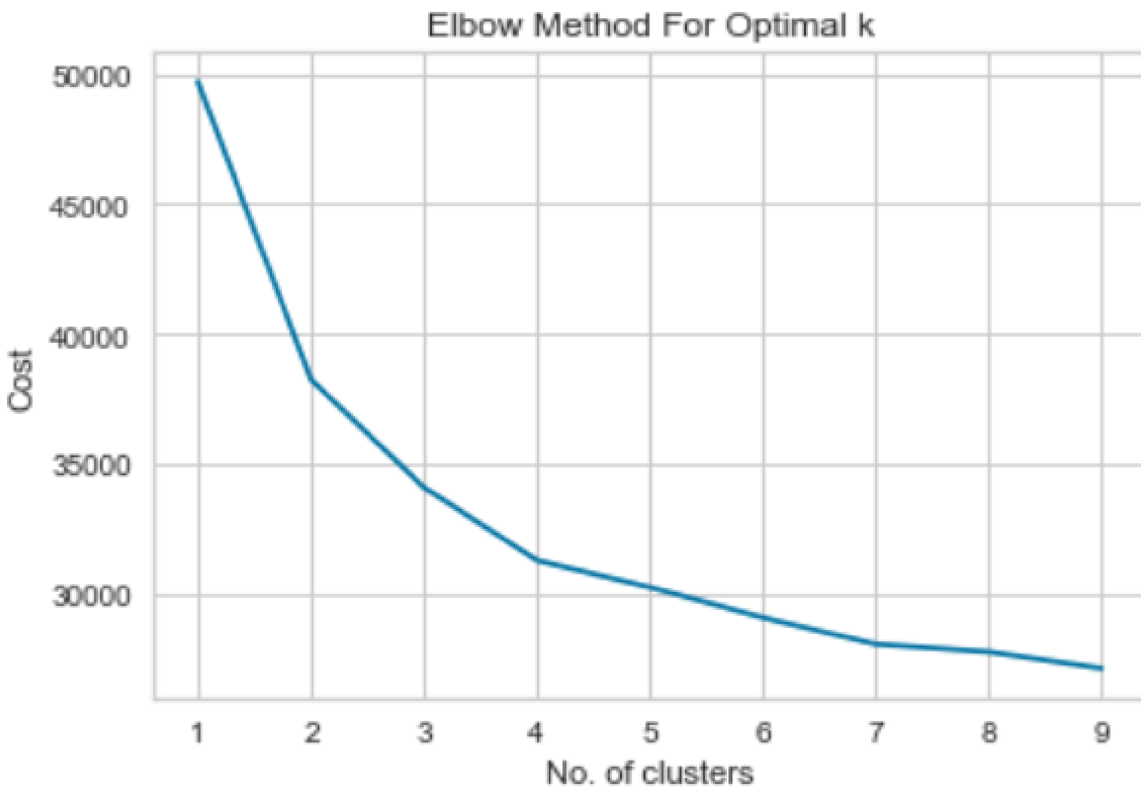
```
cost = []
K = range(1,10)
for num_clusters in list(K):
    kmode = KModes(n_clusters=num_clusters, init = "random", n_init = 5, verbose=1)
    kmode.fit_predict(df)
    cost.append(kmode.cost_)
```

```
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 1, iteration: 1/100, moves: 0, cost: 49704.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 2, iteration: 1/100, moves: 0, cost: 49704.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 3, iteration: 1/100, moves: 0, cost: 49704.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
Run 4, iteration: 1/100, moves: 0, cost: 49704.0
Init: initializing centroids
Init: initializing clusters
Starting iterations...
```

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()
```



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()

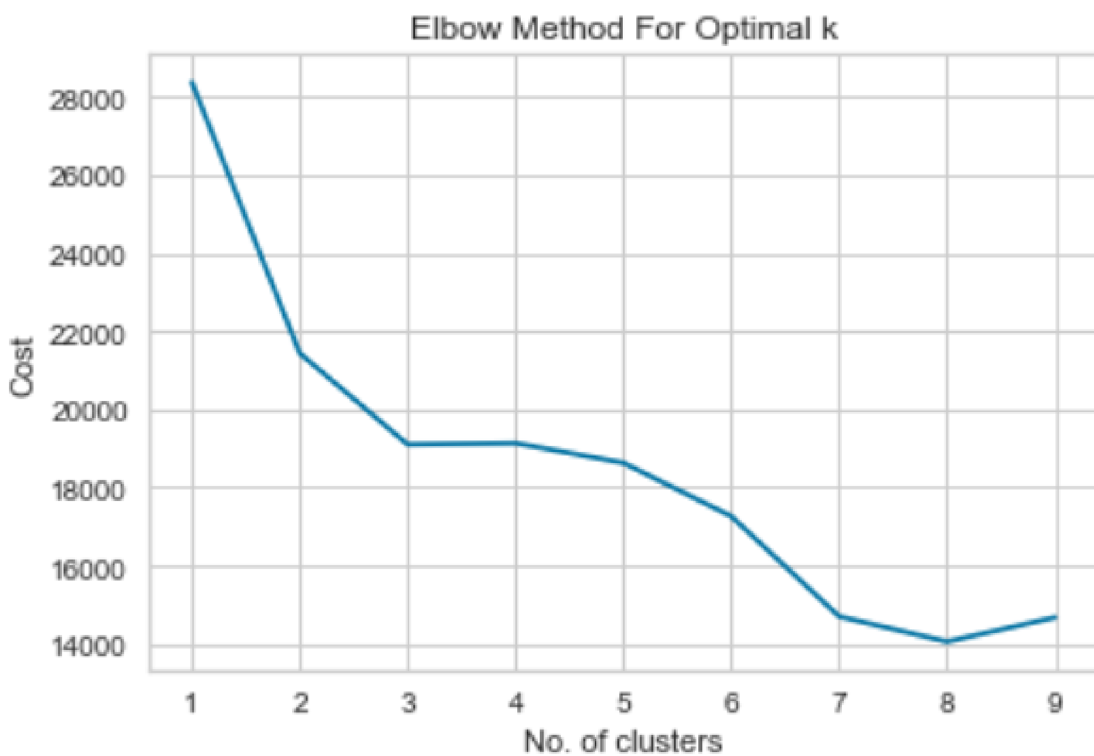
Another Method

```
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))
```

```
Cluster initiation: 1
Cluster initiation: 2
Cluster initiation: 3
Cluster initiation: 4
Cluster initiation: 5
Cluster initiation: 6
Cluster initiation: 7
Cluster initiation: 8
Cluster initiation: 9
```

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()
```



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

Silhouette Method.

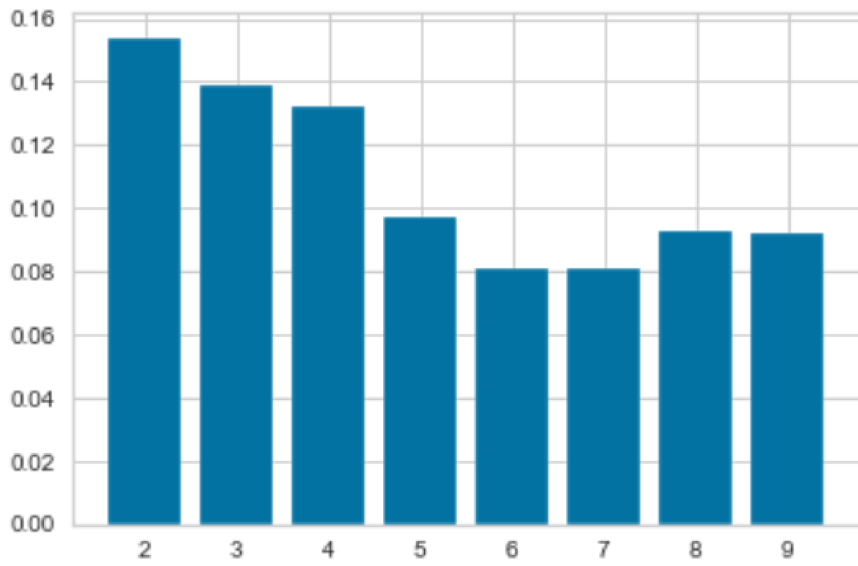
```
from sklearn.metrics import silhouette_score
K = range(2,10)
score_list=[]
for n_clusters in range(2,10):
        clusterer = KModes (n_clusters=n_clusters).fit(df)
        preds = clusterer.predict(df)
        centers = clusterer.cluster_centroids_
        score = silhouette_score (df, preds, metric='euclidean')
        score_list.append(score)
        print  ("For  n_clusters  =  {},  silhouette  score  is {})".format(n_clusters, score))
```

```
For n_clusters = 2, silhouette score is 0.15393652474912434)
For n_clusters = 3, silhouette score is 0.13919418718591922)
For n_clusters = 4, silhouette score is 0.13219882522469917)
For n_clusters = 5, silhouette score is 0.09753912402917762)
For n_clusters = 6, silhouette score is 0.08081128935552943)
For n_clusters = 7, silhouette score is 0.08121526655911628)
For n_clusters = 8, silhouette score is 0.09248224313438029)
For n_clusters = 9, silhouette score is 0.091896225895231)
```

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().



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

	seniorcitizen	partner	dependents	tenure	onlinesecurity	techsupport	contract	paperlessbilling	paymentmethod	monthlycharges	churn	Cluster Labels
0	0	1	0	0	0	0	0	1	0	0	0	1
1	0	0	0	1	1	0	1	0	0	0	0	2
2	0	0	0	0	1	0	0	1	0	0	1	1
3	0	0	0	1	1	1	1	0	1	0	0	0
4	0	0	0	0	0	0	0	1	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
7038	0	1	1	0	1	1	1	1	0	1	0	2
7039	0	1	1	1	0	0	1	1	1	1	0	2
7040	0	1	1	0	1	0	0	1	0	0	0	2
7041	1	1	0	0	0	0	0	1	0	1	1	1
7042	0	0	0	1	1	1	2	1	1	1	0	0

```
dfl['Cluster Labels'] = kmodes.labels_
```

rity	onlinebackup	deviceprotection	techsupport	streamingtv	streamingmovies	contract	paperlessbilling	paymentmethod	monthlycharges	churn	Cluster Labels
0	1	0	0	0	0	0	1	0	29.85	0	1
1	0	1	0	0	0	1	0	0	56.95	0	2
1	1	0	0	0	0	0	1	0	53.85	1	1
1	0	1	1	0	0	1	0	1	42.30	0	0
0	0	0	0	0	0	0	1	0	70.70	1	1
...	...	...	...	...	...	...	...	...	...	...	...
1	0	1	1	1	1	1	1	0	84.80	0	2
0	1	1	0	1	1	1	1	1	103.20	0	2
1	0	0	0	0	0	0	1	0	29.60	0	2
0	0	0	0	0	0	0	1	0	74.40	1	1
1	0	1	1	1	1	2	1	1	105.65	0	0

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.

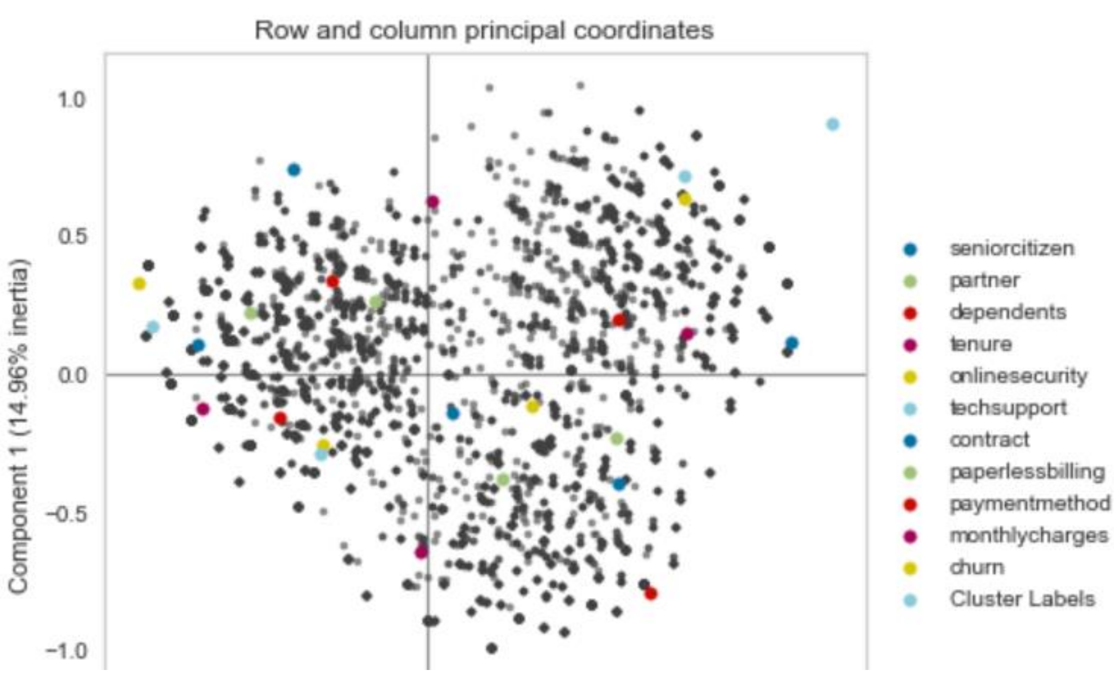
```
mca.fit(dff)
dff_mca = mca.transform(dff)
```

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

dff_mca.head()

	0	1
0	-0.425880	-0.218552
1	0.267768	-0.438697
2	-0.608838	0.139386
3	0.688313	0.293884
4	-0.770407	0.215646

mca.plot_coordinates(X = dff,legend_n_cols=1).legend(loc='center left', bbox_to_anchor=(1, 0.5)) – The result is plotted.



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

vice	onlinesecurity	onlinebackup	deviceprotection	techsupport	streamingtv	streamingmovies	contract	paperlessbilling	paymentmethod	monthlycharges	churn	Cluster Labels
0	0	1	0	0	0	0	0	1	0	29.85	0	1
0	1	1	0	0	0	0	0	1	0	53.85	1	1
1	0	0	0	0	0	0	0	1	0	70.70	1	1
1	0	0	1	0	1	1	0	1	0	99.65	1	1
1	0	1	0	0	1	0	0	1	1	89.10	0	1
...	...	...	...	...	...	...	...	...	...	...	...	...
1	1	1	1	0	1	0	0	1	1	102.95	1	1
1	0	0	0	0	1	0	0	1	1	78.70	0	1
0	0	1	1	1	1	1	1	0	0	60.65	0	1
2	0	0	0	0	0	0	2	1	1	21.15	0	1

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()
```