Credit Card Clustering and Segmentation

Unsupervised Learning ¶

Author : Abhinav Tyagi

Domain: Banking, Finance

Business Context:

This case requires to develop a customer segmentation to define marketing strategy. The sample Dataset summarizes the usage behavior of about 9000 active credit card holders during the last 6 months. The file is at a customer level with 18 behavioral variables.

Data Dictionary:

CUSTID: Identification of Credit Card holder (Categorical)

BALANCE: Balance amount left in their account to make purchases

BALANCEFREQUENCY: How frequently the Balance is updated, score between 0 and 1 (1 = frequently updated, 0 = not frequently updated)

PURCHASES: Amount of purchases made from account

ONEOFFPURCHASES: Maximum purchase amount done in one-go

INSTALLMENTSPURCHASES: Amount of purchase done in installment

CASHADVANCE: Cash in advance given by the user

PURCHASESFREQUENCY: How frequently the Purchases are being made, score between 0 and 1 (1 = frequently purchased, 0 = not frequently purchased)

ONEOFFPURCHASESFREQUENCY: How frequently Purchases are happening in one-go (1 = frequently purchased, 0 = not frequently purchased)

PURCHASESINSTALLMENTSFREQUENCY: How frequently purchases in installments are being done (1 = frequently done, 0 = not frequently done)

CASHADVANCEFREQUENCY: How frequently the cash in advance being paid

CASHADVANCETRX: Number of Transactions made with "Cash in Advance"

PURCHASESTRX: Numbe of purchase transactions made

CREDITLIMIT: Limit of Credit Card for user

PAYMENTS: Amount of Payment done by user

MINIMUM_PAYMENTS: Minimum amount of payments made by user

PRCFULLPAYMENT: Percent of full payment paid by user

TENURE: Tenure of credit card service for user

Steps:

1. Preprocessing the data (15 points)

a. Check a few observations and get familiar with the data. (1 points)

In [1]:
```
# import necessary tools
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# preprocessing
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA

# ignore warnings
import warnings
warnings.filterwarnings(action="ignore")

# clustering
from sklearn.cluster import KMeans, AgglomerativeClustering
from sklearn.mixture import GaussianMixture
from matplotlib import cm
from sklearn.metrics import silhouette_samples, silhouette_score
```

In [2]:
```
# load the data
data = pd.read_csv('data_credit_card.csv')
```

In [3]:
```
# data overview
print('Data shape: ' + str(data.shape))
data.head()
```

```
Data shape: (8950, 18)
```

Out[3]:

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMI
0	C10001	40.900749	0.818182	95.40	0.00	
1	C10002	3202.467416	0.909091	0.00	0.00	
2	C10003	2495.148862	1.000000	773.17	773.17	
3	C10004	1666.670542	0.636364	1499.00	1499.00	
4	C10005	817.714335	1.000000	16.00	16.00	

b. Check the size and info of the data set. (2 points)

In [4]:
```
data.describe()
```

Out[4]:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_
count	8950.000000	8950.000000	8950.000000	8950.000000	
mean	1564.474828	0.877271	1003.204834	592.437371	
std	2081.531879	0.236904	2136.634782	1659.887917	
min	0.000000	0.000000	0.000000	0.000000	
25%	128.281915	0.888889	39.635000	0.000000	
50%	873.385231	1.000000	361.280000	38.000000	
75%	2054.140036	1.000000	1110.130000	577.405000	
max	19043.138560	1.000000	49039.570000	40761.250000	

c. Check for missing values. Impute the missing values if there is any. (2 points)

In [5]:
```
data.isna().sum()
```

Out[5]:
```
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
```

We will impute these missing values with the median value.

In [6]:
```
# impute with median
data.loc[(data['MINIMUM_PAYMENTS'].isnull()==True),'MINIMUM_PAYMENTS'] = data['MI
data.loc[(data['CREDIT_LIMIT'].isnull()==True),'CREDIT_LIMIT'] = data['CREDIT_LIM
```

In [7]:
```
# double check
data.isna().sum()
```

Out[7]:
```
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
```

d. Drop unnecessary columns. (2 points)

In [8]:
```
# drop ID column
data = data.drop('CUST_ID', 1)
```

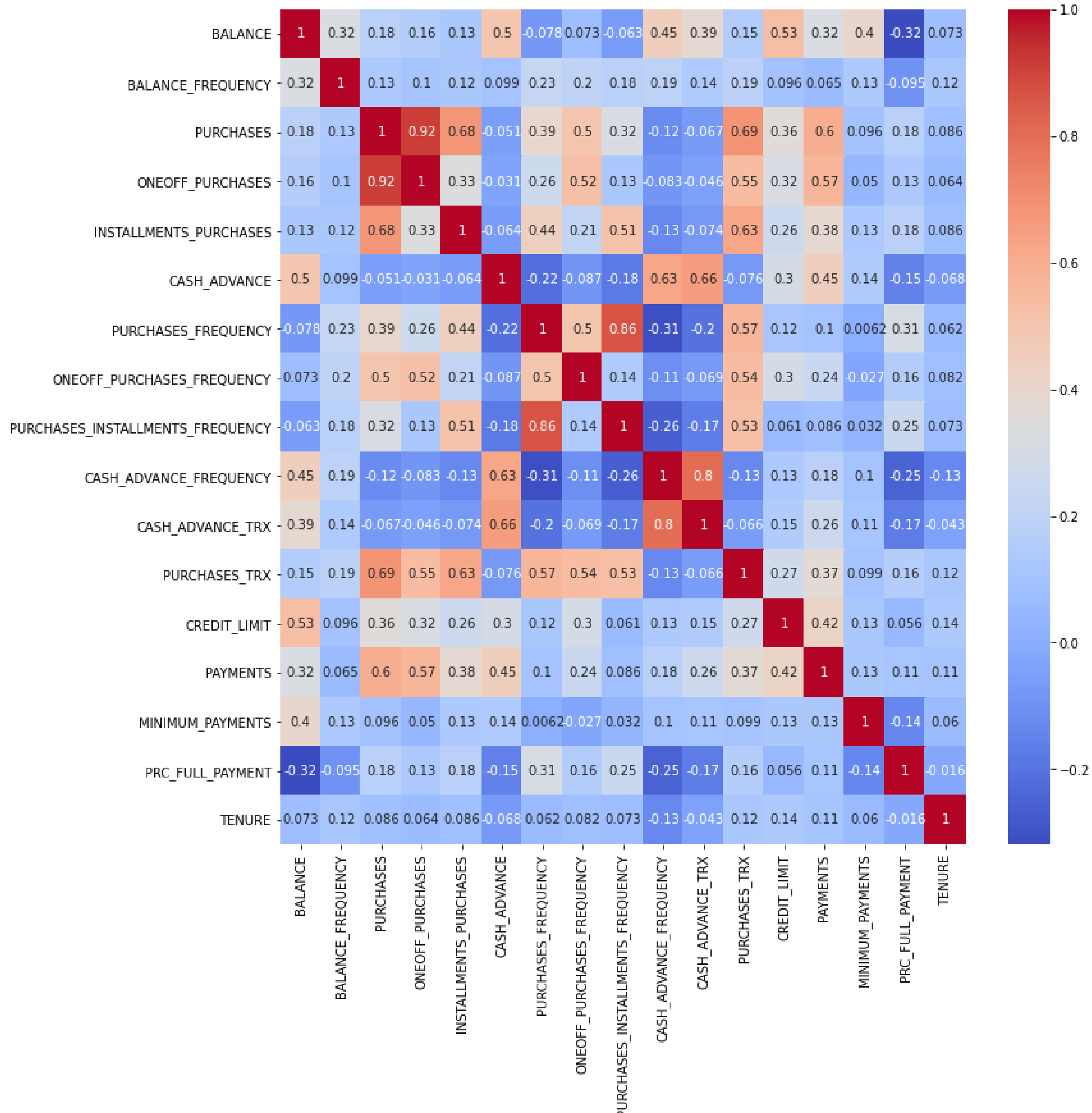
In [9]:
```
data.shape
```

Out[9]:
```
(8950, 17)
```

e. Check correlation among features and comment your findings. (3 points)

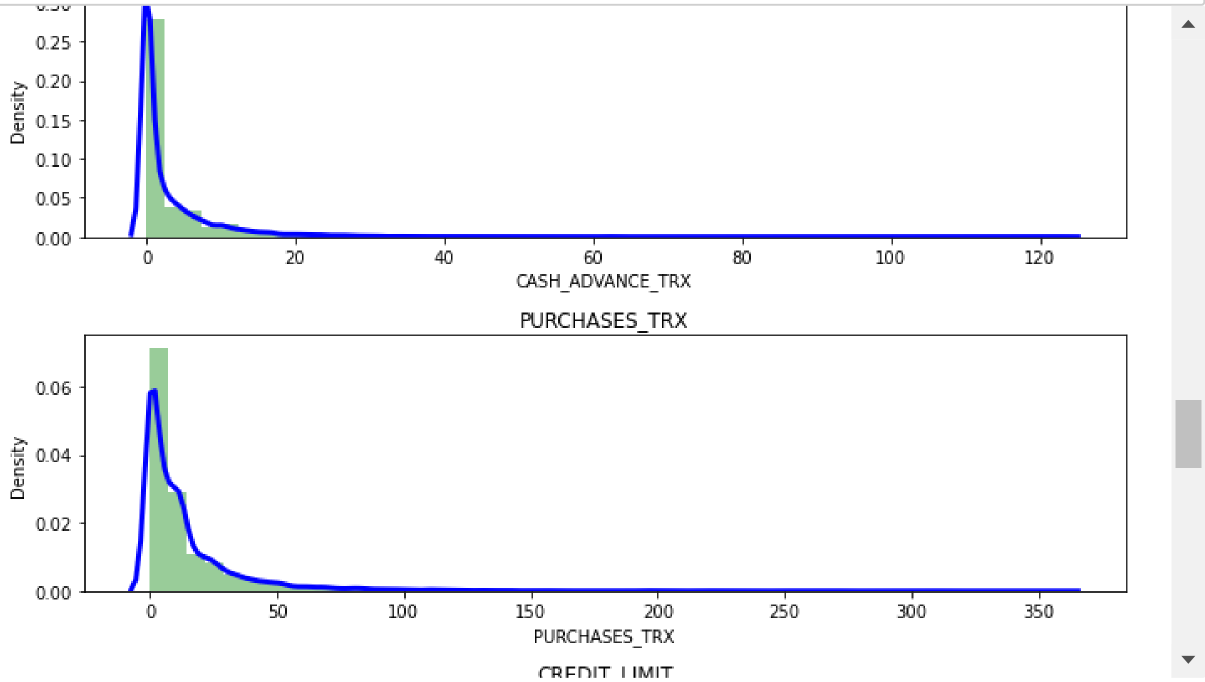
In [10]:
```
plt.figure(figsize = (12, 12))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm',
            xticklabels=data.columns,
            yticklabels=data.columns)
```

Out[10]: <AxesSubplot:>



Check distribution of features and comment your findings. (3 points)

In [11]:
```
# distplot with KDE
plt.figure(figsize=(10,50))
for i in range(len(data.columns)):
    plt.subplot(17,1,i+1)
    sns.distplot(data[data.columns[i]],kde_kws={'color':'b', 'lw':3, 'label':'KDE
    plt.title(data.columns[i])

plt.tight_layout()
```



Few observations

Mean of balance is somewhere between $1000 and $2000 'Balance_Frequency' for most customers is updated frequently at 1 For 'PURCHASES_FREQUENCY', there are two distinct group of customers at 0 and 1

For 'ONEOFF_PURCHASES_FREQUENCY' and 'PURCHASES_INSTALLMENT_FREQUENCY' most users don't do one off puchases or installment purchases frequently

Very small number of customers pay their balance in full 'PRC_FULL_PAYMENT'~0 Average credit limit is around $5000 Most customers have tenure between 11 and 12

g. Standardize the data using appropriate methods. (2 points)

In [12]:
```
# normalize values
scaler = StandardScaler()
data_scaled = scaler.fit_transform(data)
data_scaled.shape
```

Out[12]: (8950, 17)

In [13]:
```
data_imputed = pd.DataFrame(data_scaled, columns=data.columns)
```

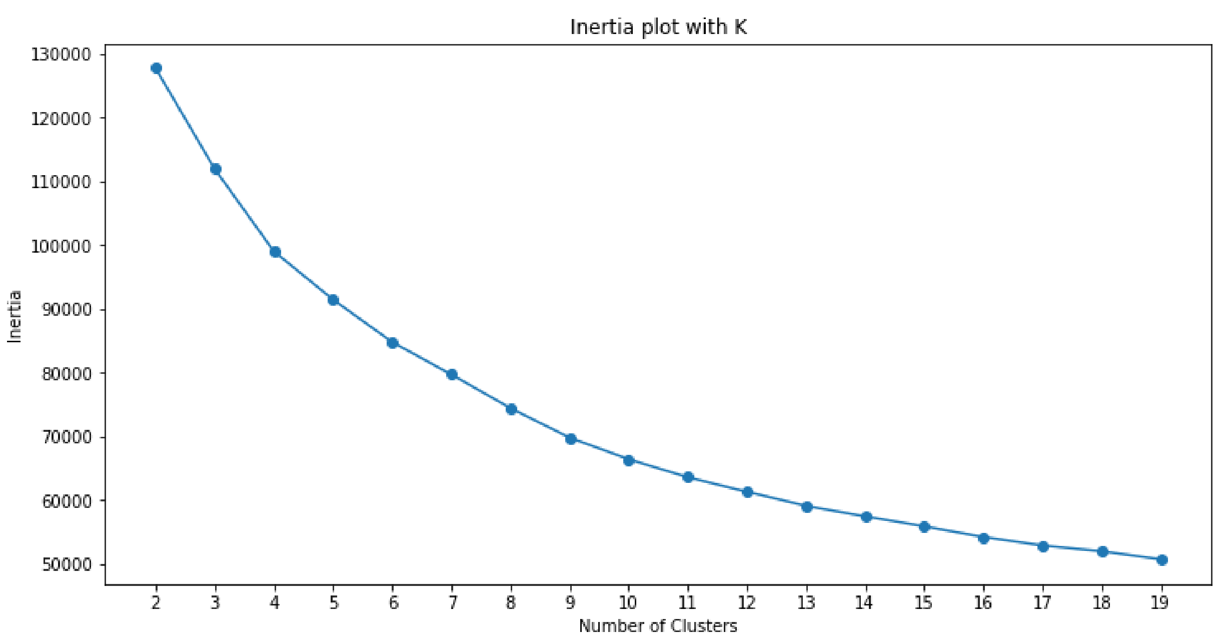
2. Build a k-means algorithm for clustering credit card data. Kindly follow the below steps and answer the following. (10 points)

a. Build k means model on various k values and plot the inertia against various k values

In [14]:
```
# inertia plotter function
def inertia_plot(clust, X, start = 2, stop = 20):
    inertia = []
    for x in range(start,stop):
        km = clust(n_clusters = x)
        labels = km.fit_predict(X)
        inertia.append(km.inertia_)
    plt.figure(figsize = (12,6))
    plt.plot(range(start,stop), inertia, marker = 'o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.title('Inertia plot with K')
    plt.xticks(list(range(start, stop)))
    plt.show()
```

c. Plot an elbow plot to find the optimal value of k

In [15]:
```
inertia_plot(KMeans, data_imputed)
```



As you can see from elbow plot , we can begin our clustering from 2 to 6

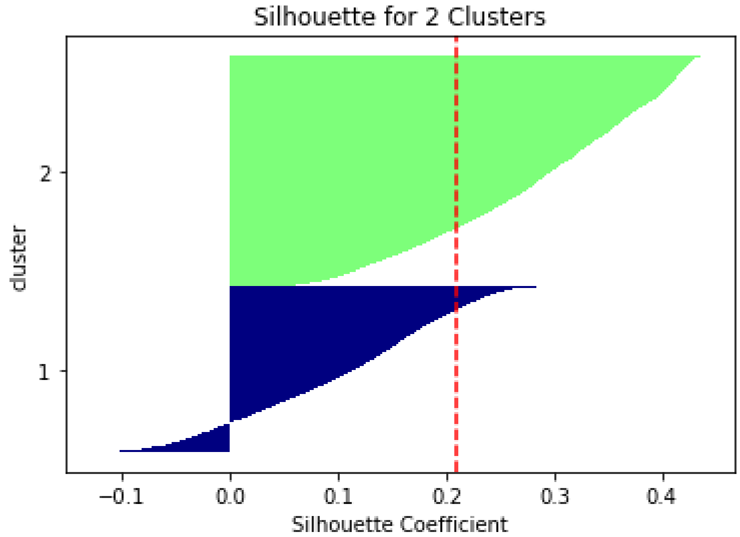
b. Evaluate the model using Silhouette coefficient

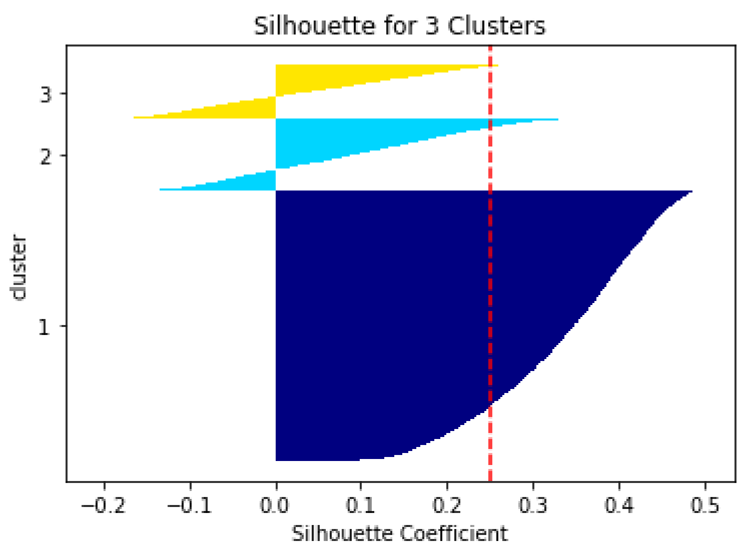
In [16]:
```
def silh_samp_cluster(clust,  X, start=2, stop=5, metric = 'euclidean'):
    # taken from sebastian Raschkas book Python Machine Learning second edition
    for x in range(start, stop):
        km = clust(n_clusters = x)
        y_km = km.fit_predict(X)
        cluster_labels = np.unique(y_km)
        n_clusters = cluster_labels.shape[0]
        silhouette_vals = silhouette_samples(X, y_km, metric = metric)
        y_ax_lower, y_ax_upper =0,0
        yticks = []
        for i, c in enumerate(cluster_labels):
            c_silhouette_vals = silhouette_vals[y_km == c]
            c_silhouette_vals.sort()
            y_ax_upper += len(c_silhouette_vals)
            color = cm.jet(float(i)/n_clusters)
            plt.barh(range(y_ax_lower, y_ax_upper),
                    c_silhouette_vals,
                    height=1.0,
                    edgecolor='none',
                    color = color)
            yticks.append((y_ax_lower + y_ax_upper)/2.)
            y_ax_lower+= len(c_silhouette_vals)

        silhouette_avg = np.mean(silhouette_vals)
        plt.axvline(silhouette_avg,
                   color = 'red',
                   linestyle = "--")
        plt.yticks(yticks, cluster_labels+1)
        plt.ylabel("cluster")
        plt.xlabel('Silhouette Coefficient')
        plt.title('Silhouette for ' + str(x) + " Clusters")
        plt.show()
```

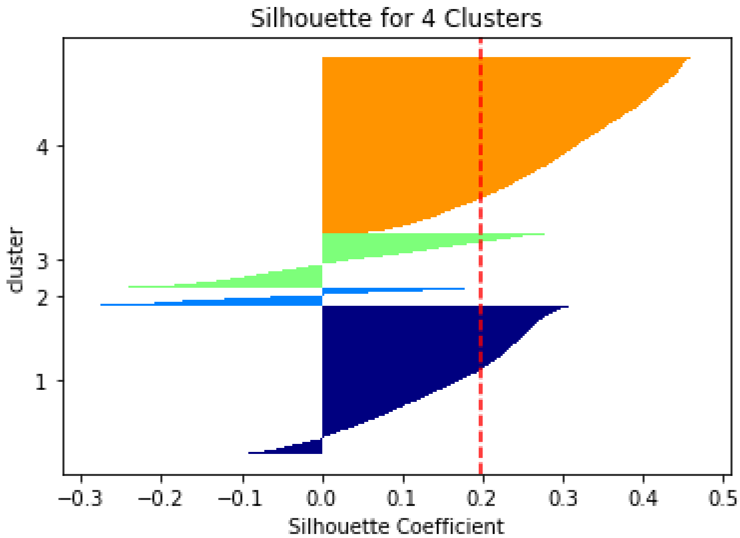
In [17]:
```
for x in range(2, 7):
    alg = KMeans(n_clusters = x, )
    label = alg.fit_predict(data_imputed)
    print('Silhouette-Score for', x,  'Clusters: ', silhouette_score(data_imputed
```

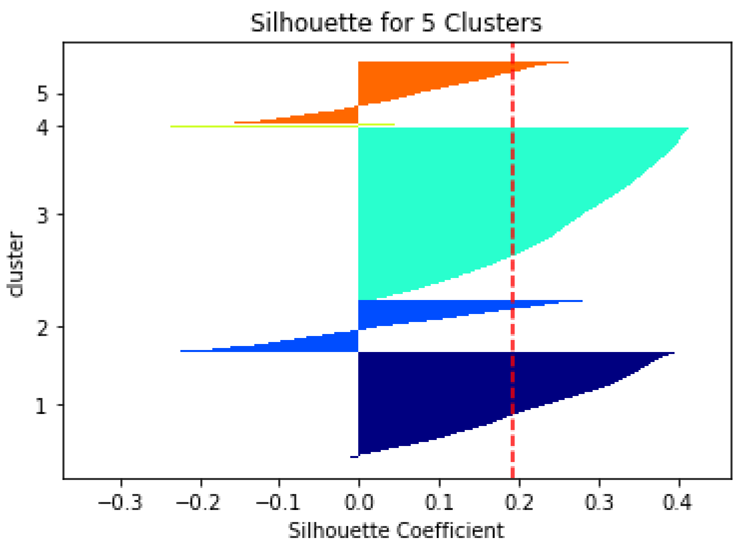
```
Silhouette-Score for 2 Clusters:  0.20943378745792432
Silhouette-Score for 3 Clusters:  0.25061926305697263
Silhouette-Score for 4 Clusters:  0.1976791965228765
Silhouette-Score for 5 Clusters:  0.193152207188095 6
Silhouette-Score for 6 Clusters:  0.20280947884863693
```

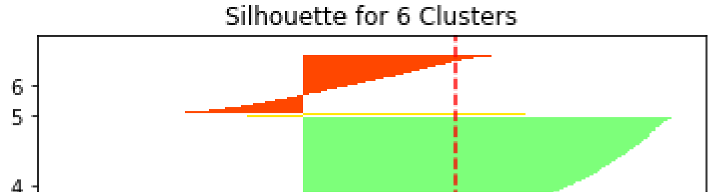
```
In [18]: silh_samp_cluster(KMeans, data_imputed, stop=7)
```






Silhouette for 4 Clusters
cluster
4
3
2
1
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
0.5
Silhouette Coefficient


Silhouette for 5 Clusters
cluster
5
4
3
2
1
−0.3
−0.2
−0.1
0.0
0.1
0.2
0.3
0.4
Silhouette Coefficient



d. Which k value gives the best result?

So far, we have a high average inertia, low silhouette scores, and very wide fluctuations in the size of the silhouette plots.

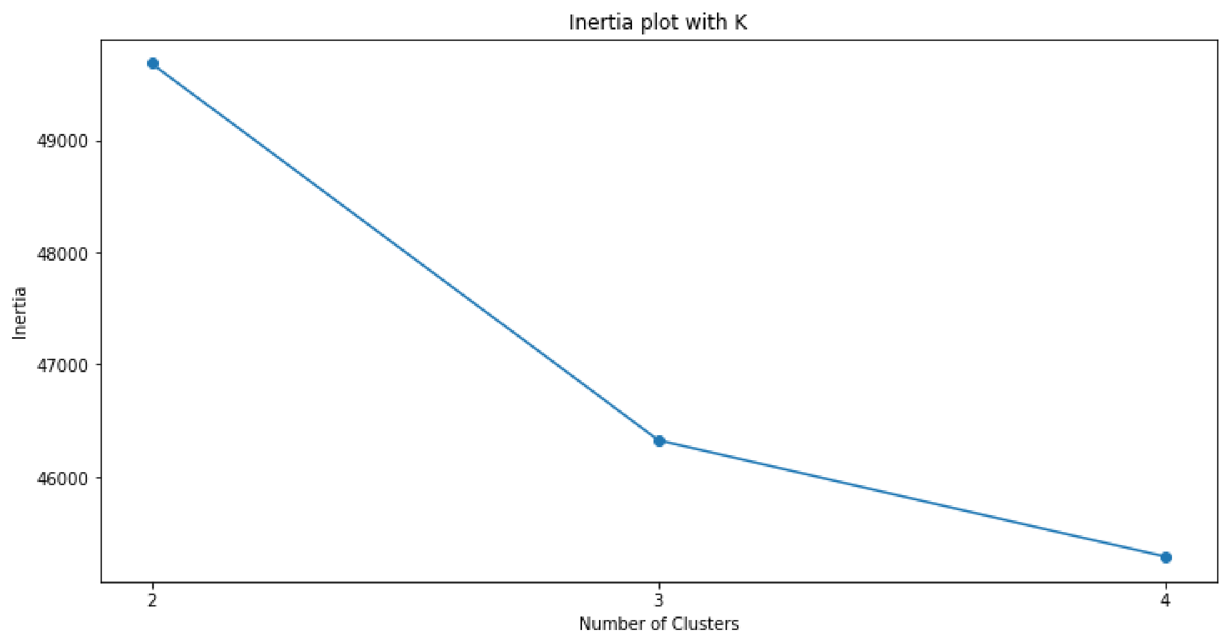
3. Apply PCA to the dataset and perform all steps from Q2 on the new features generated using PCA. (15points)

a. Build k means model on various k values and plot the inertia against various k values

In [19]:
```
# inertia plotter function
def inertia_plot_PCA(clust, X, start = 2, stop = 5):
    inertia = []
    for x in range(start,stop):
        pca = PCA(n_components=x)
        data_p = pca.fit_transform(data_imputed)
        alg = KMeans(n_clusters = x, )
        label = alg.fit_predict(data_p)
        inertia.append(alg.inertia_)

    plt.figure(figsize = (12,6))
    plt.plot(range(start,stop), inertia, marker = 'o')
    plt.xlabel('Number of Clusters')
    plt.ylabel('Inertia')
    plt.title('Inertia plot with K')
    plt.xticks(list(range(start, stop)))
    plt.show()
```

c. Plot an elbow plot to find the optimal value of k

In [20]:
```
inertia_plot_PCA(KMeans, data_imputed)
```



b. Evaluate the model using Silhouette coefficient

Clustering metrices

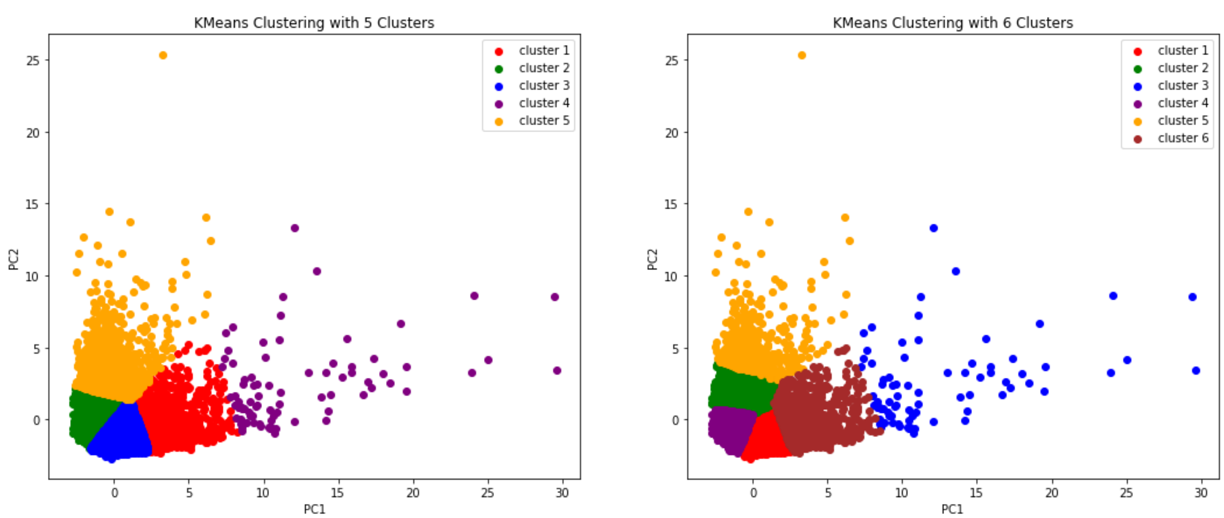
In [21]:
```
# apply PCA and display clustering metrics
for y in range(2, 5):
    print("PCA with # of components: ", y)
    pca = PCA(n_components=y)
    data_p = pca.fit_transform(data_imputed)
    for x in range(2,7):
        alg = KMeans(n_clusters = x, )
        label = alg.fit_predict(data_p)
        print('Silhouette-Score for', x,  'Clusters: ', silhouette_score(data_p,

    print()
```

```
PCA with # of components:  2
Silhouette-Score for 2 Clusters:  0.4615418454360034        Inertia:  49682.541
237453064
Silhouette-Score for 3 Clusters:  0.45208830427838265        Inertia:  33032.02
578251903
Silhouette-Score for 4 Clusters:  0.40736275385633836        Inertia:  24544.70
7526507646
Silhouette-Score for 5 Clusters:  0.40108971838046475        Inertia:  19475.98
3853955368
Silhouette-Score for 6 Clusters:  0.3834061926212805        Inertia:  16226.450
833537487

PCA with # of components:  3
Silhouette-Score for 2 Clusters:  0.341554339431114        Inertia:  62045.2071
743054
Silhouette-Score for 3 Clusters:  0.3797267472729141        Inertia:  46325.648
985244385
Silhouette-Score for 4 Clusters:  0.36911099565053856        Inertia:  34659.79
6707342226
Silhouette-Score for 5 Clusters:  0.36828487418767697        Inertia:  28591.59
5685369113
Silhouette-Score for 6 Clusters:  0.3314254572756625        Inertia:  24847.703
11958464

PCA with # of components:  4
Silhouette-Score for 2 Clusters:  0.3054493295215671        Inertia:  73185.440
84374607
Silhouette-Score for 3 Clusters:  0.3431853363667038        Inertia:  57561.213
50940583
Silhouette-Score for 4 Clusters:  0.32197036004797314        Inertia:  45288.35
360230382
Silhouette-Score for 5 Clusters:  0.31691692809050803        Inertia:  39218.58
015654367
Silhouette-Score for 6 Clusters:  0.2922214019214511        Inertia:  35303.487
68300758
```

Visualisation

In [22]:
```
data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(KMeans(n_clusters = 5,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

fig = plt.figure(figsize = (18, 7))
colors = ['red', 'green', 'blue', 'purple', 'orange', 'brown']
plt.subplot(121)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.legend()
plt.title('KMeans Clustering with 5 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')


data_p = pd.DataFrame(PCA(n_components = 2).fit_transform(data_imputed))
preds = pd.Series(KMeans(n_clusters = 6,).fit_predict(data_p))
data_p = pd.concat([data_p, preds], axis =1)
data_p.columns = [0,1,'target']

plt.subplot(122)
plt.scatter(data_p[data_p['target']==0].iloc[:,0], data_p[data_p.target==0].iloc
plt.scatter(data_p[data_p['target']==1].iloc[:,0], data_p[data_p.target==1].iloc
plt.scatter(data_p[data_p['target']==2].iloc[:,0], data_p[data_p.target==2].iloc
plt.scatter(data_p[data_p['target']==3].iloc[:,0], data_p[data_p.target==3].iloc
plt.scatter(data_p[data_p['target']==4].iloc[:,0], data_p[data_p.target==4].iloc
plt.scatter(data_p[data_p['target']==5].iloc[:,0], data_p[data_p.target==5].iloc
plt.legend()
plt.title('KMeans Clustering with 6 Clusters')
plt.xlabel('PC1')
plt.ylabel('PC2')
```

Out[22]:
```
Text(0, 0.5, 'PC2')
```



4. Create a new column as a cluster label in the original data frame and perform cluster analysis. Check the correlation of cluster labels with various features and mention your inferences. (Hint - Does cluster 1 have a high credit limit?) (5 points)

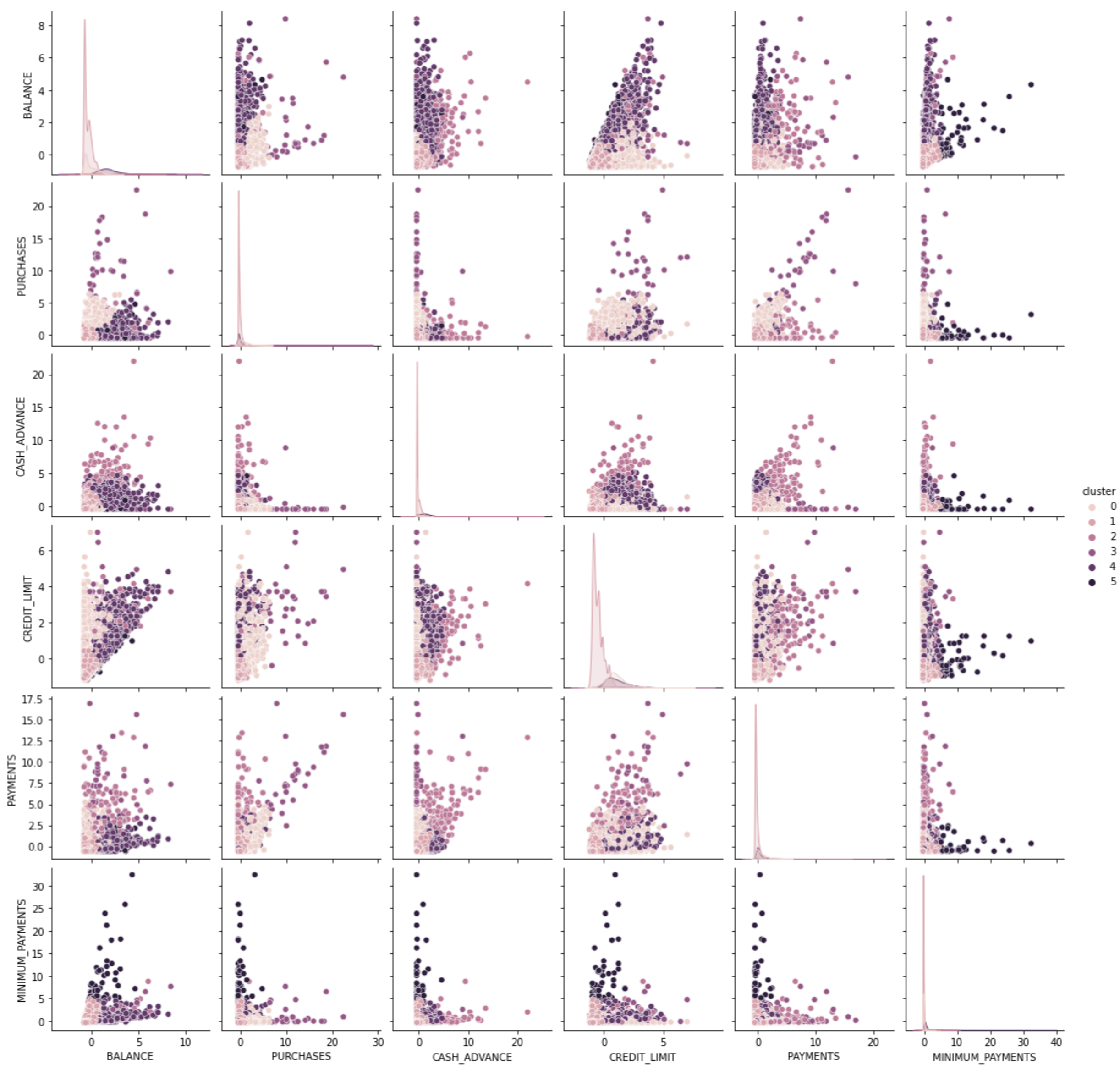
We are picking 6 clusters for this EDA. Let's make a Seaborn pairplot with selected/best columns to show how the clusters are segmenting the samples:

In [27]:
```
# select best columns
best_cols = ["BALANCE", "PURCHASES", "CASH_ADVANCE", "CREDIT_LIMIT", "PAYMENTS",

# dataframe with best columns
data_final = pd.DataFrame(data_imputed[best_cols])

print('New dataframe with best columns has just been created. Data shape: ' + str
```

```
New dataframe with best columns has just been created. Data shape: (8950, 6)
```

In [28]:
```
# apply KMeans clustering
alg = KMeans(n_clusters = 6)
label = alg.fit_predict(data_final)

# create a 'cluster' column
data_final['cluster'] = label
best_cols.append('cluster')

# make a Seaborn pairplot
sns.pairplot(data_final[best_cols], hue='cluster')
```

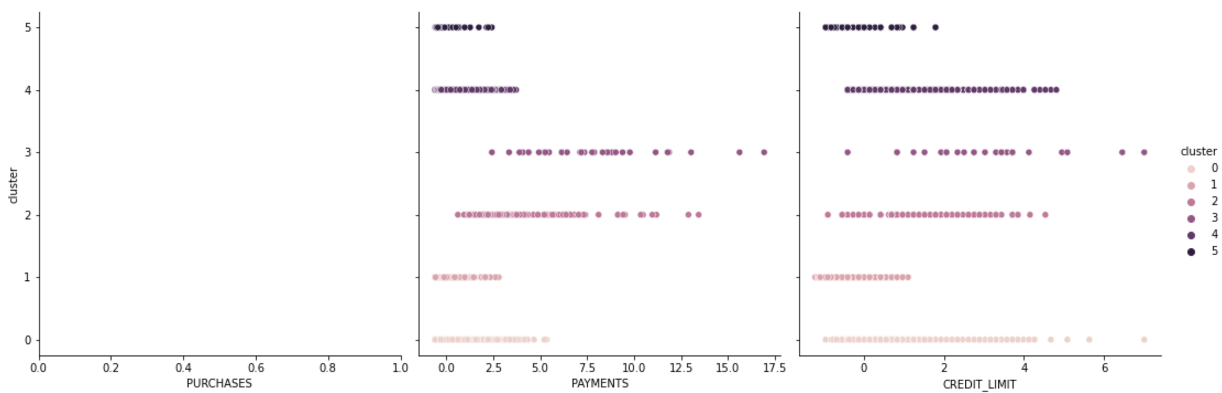
Out[28]: <seaborn.axisgrid.PairGrid at 0x1bb040f19d0>



Cluster 0

In [29]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS
             y_vars=['cluster'],
             height=5, aspect=1)
```

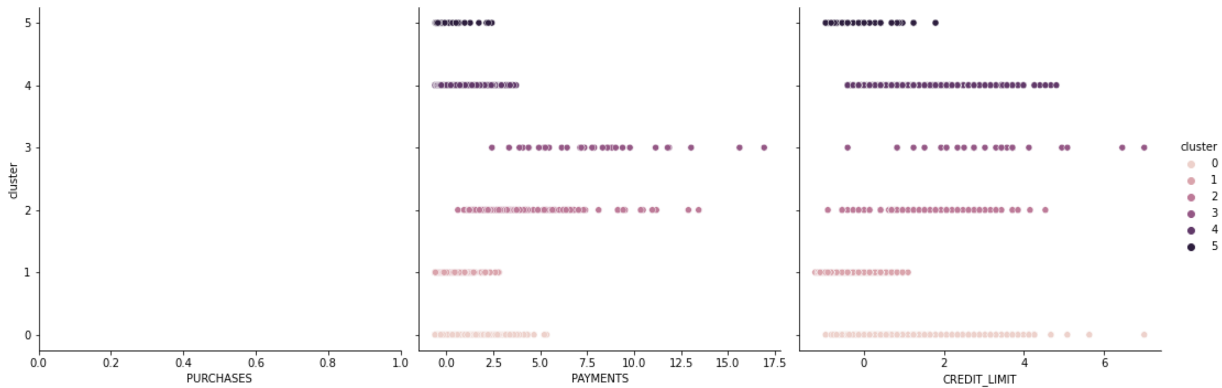
Out[29]: <seaborn.axisgrid.PairGrid at 0x1bb03b6e610>



Cluster 1

In [30]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS
             y_vars=['cluster'],
             height=5, aspect=1)
```

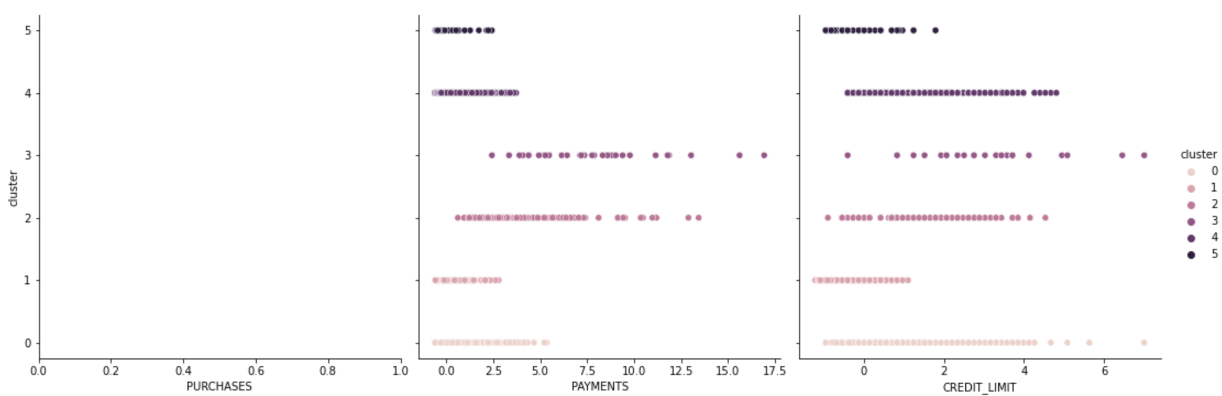
Out[30]: <seaborn.axisgrid.PairGrid at 0x1bb040f12e0>



Cluster 2

In [32]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES', 'PAYMENTS
             height=5, aspect=1)
```

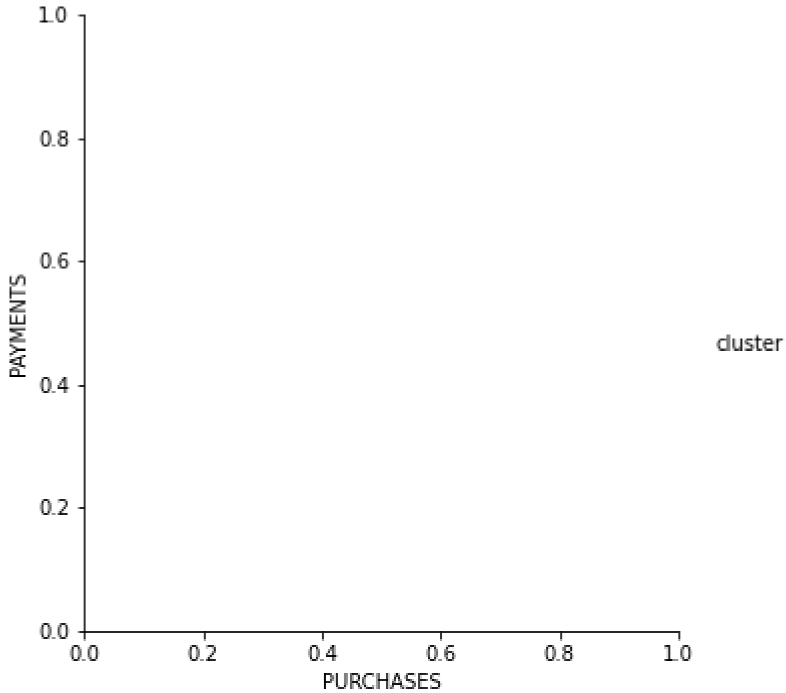
Out[32]: <seaborn.axisgrid.PairGrid at 0x1bb042ac520>



Cluster 3

In [33]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['PURCHASES'], y_vars=[
             height=5, aspect=1)
```

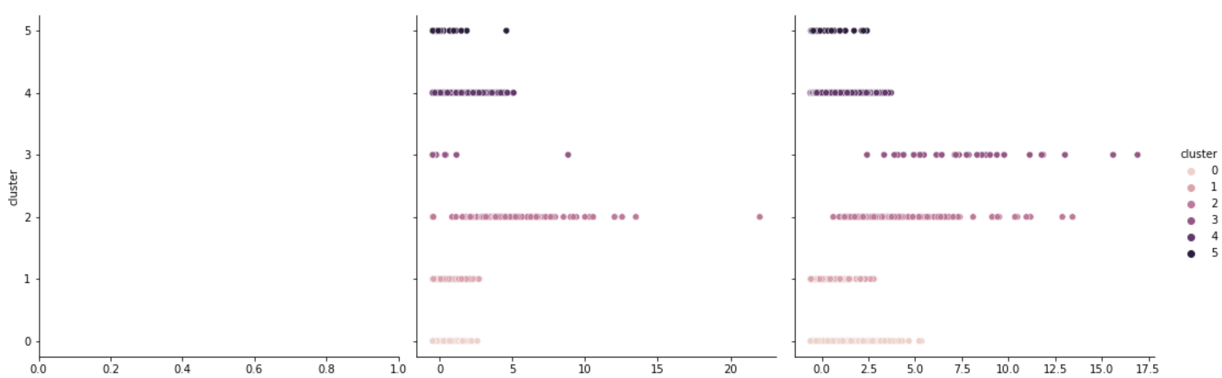
Out[33]: <seaborn.axisgrid.PairGrid at 0x1bb3d490b50>



Cluster 4

In [34]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE', 'CASH_ADVAN
             y_vars=['cluster'],
             height=5, aspect=1)
```

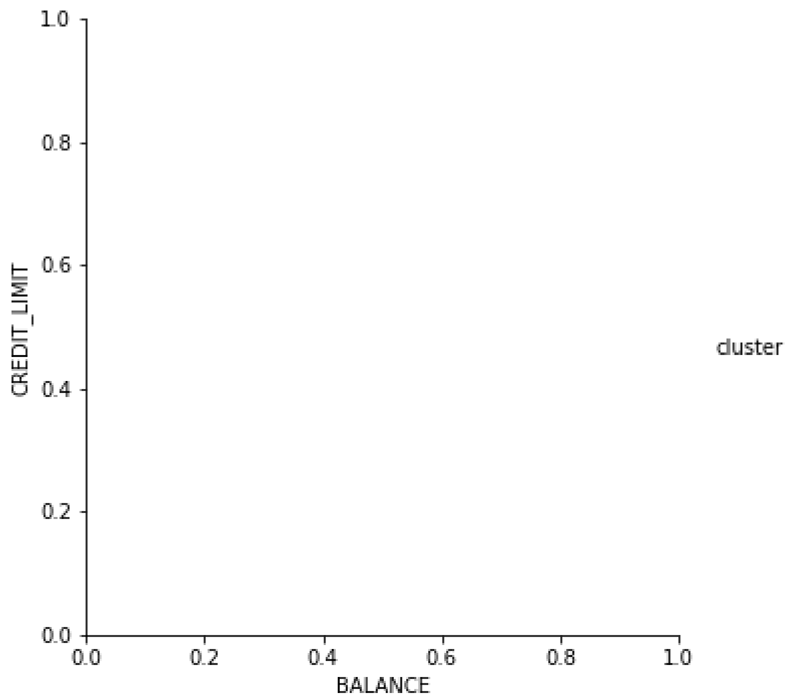
Out[34]: <seaborn.axisgrid.PairGrid at 0x1bb06465df0>



Cluster 5

In [35]:
```
sns.pairplot(data_final[best_cols], hue='cluster', x_vars=['BALANCE'], y_vars=['C
             height=5, aspect=1)
```

Out[35]: <seaborn.axisgrid.PairGrid at 0x1bb044f1c70>



5. Comment your findings and inferences and compare the performance. Does applying PCA give a better result in comparison to earlier? (5 points)

We have learned a lot from this dataset by segmenting the customers into six smaller groups: the Average Joe, the Active Users, the Big Spenders, the Money Borrowers, the High Riskers, and the Wildcards. To conclude this cluster analysis, let's sum up what we have learned and some possible marketing strategies:

The Average Joe do not use credit card very much in their daily life. They have healthy finances and low debts. While encouraging these people to use credit cards more is necessary for the company's profit, business ethics and social responsibility should also be considered.

Identify active customers in order to apply proper marketing strategy towards them. These people are the main group that we should focus on.

Some people are just bad at finance management - for example, the Money Borrowers. This should not be taken lightly.

Although we are currently doing a good job at managing the High Riskers by giving them low credit limits, more marketing strategies targeting this group of customers should be considered.

Conclusion

Conclusion

In this project, we have performed data preprocessing, feature extraction with PCA, looked at various clustering metrics (inertias, silhouette scores), experimented with various Clustering algorithms (KMeans Clustering, Agglomerative Hierarchical Clustering, Gaussian Mixture Clustering), data visualizations, and business analytics.

In [ ]:

```

```