MACHINE LEARNING ENGINEER NANODEGREE
CAPSTONE PROJECT PROPOSAL
CREDIT CARD CUSTOMER SEGMENTATION

James Wanderi

August, 2019

I. Definition

Project Overview

This project intends to show how to use credit card data belonging to different credit card holders to gain insights that can be used to group the holders into clusters that can be useful to the credit card issuers.

Data belonging to different customers were collected and summarized guaranteeing anonymity. This data was prepared by Arjun [4] and is available on Kaggle website [3, 4]. This usage data offers an opportunity to conduct behavioral analysis or spending pattern for these card holder. In my opinion, this was Arjun's intention when availing the data to the Kaggle community.

There exists many financial bank institutions especially in Africa who haven't embraced Machine Learning in realizing the potential of vast transactional data in their possession.

One of the major revenue contributors of a financial institution such as a bank is cash advance to their customers in order to spend and pay them later at an interest. Credit card is a payment card issued by such an institution to their customers to enable the cardholder to pay a merchant for goods and services based on the customer's promise to the institution to pay them for the amounts plus other agreed charges [1].

In marketing and risk-avoidance strategies targeting the credit card holders, the banks should be able to know the holders' spending behaviours, affordability and credit risk in such a way that decisions can be made easily. Knowing if a card holder expends their credit card limit in the first week of the month might inform the bank to double the limit of the holder by updating the limit fortnightly on condition that holder can repay the credit advance.

This project shows a way of segmenting the card holders so that such decisions can be made easily by the bank in order to be adaptive in their product offerings, avoid risks, be effective and make profits.

Problem Statement

A financial institution should be able to adaptively target a card holder with different product offerings depending on the holder's usage behavior of the card.

This will not only increase the institution's revenue but also reduce the cost associated with wasteful marketing and promotions.

We will approach this as an Unsupervised Machine Learning [9] problem. We have unlabelled N observations (usage data) which we need to apply a label that can be used to infer other new observations.

First, we will analyse the data in order to understand the data and how the features relate to another. We will find out the most relevant features and hopefully use them for clustering.

We will then apply labels to the data and use Supervised Machine Learning to be able predict which group or cluster a card holder belongs to.

Our solution will comprise the following:

1. Explorative analysis to the training data
2. Solution to the problem
3. Solution artefacts such as Jupyter notebook showing how the solution was being sought.
4. Final report (this one) defining the problem and its solution

Metrics

This is a clustering problem ad we will use **silhouette** score [7, 8] to determine how good our clusters have been formed. We look at how close members of clusters are close to one another.

The Silhouette Coefficient is calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample. The Silhouette Coefficient for a sample is (b - a) / max(a, b). To clarify, b is the distance between the sample and the nearest cluster that the sample is not a part of. Note that Silhouette Coefficient is only defined if number of labels is 2 <= n_labels <= n_samples - 1.

The best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters. Negative values generally indicate that a sample has been assigned to the wrong cluster, as a different cluster is more similar. [10]

We also be calculating the **coefficient of determination, R^2**, [11] to quantify our simple prediction model's performance on our newly derived labelled data. The

coefficient of determination for a model is a useful statistic in regression analysis, as it often describes how "good" that model is at making predictions.

II. Analysis

Data Exploration

There exists 8950 records in our raw data. This data has 18 features.

	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	...	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
0	C10001	40.900749	0.818182	95.40	...	201.802084	139.509787	0.000000	12
1	C10002	3202.467416	0.909091	0.00	...	4103.032597	1072.340217	0.222222	12
2	C10003	2495.148862	1.000000	773.17	...	622.066742	627.284787	0.000000	12
3	C10004	1666.670542	0.636364	1499.00	...	0.000000	NaN	0.000000	12
4	C10005	817.714335	1.000000	16.00	...	678.334763	244.791237	0.000000	12

5 rows × 18 columns

```
input_data.shape
```

```
(8950, 18)
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
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

Note how some features' standard deviation are greater than the mean, e.g BALANCE, PURCHASES.

This clearly shows how the data varies and may create very distinct set of clusters.

It is also obvious that the data needs to be normalized.

Exploratory Visualization

We have 18 features that may or may not be useful. CUST_ID is one that is not useful in clustering nor in prediction.

We will see how the remaining features are related to one another. We will do this by using other features to predict another.

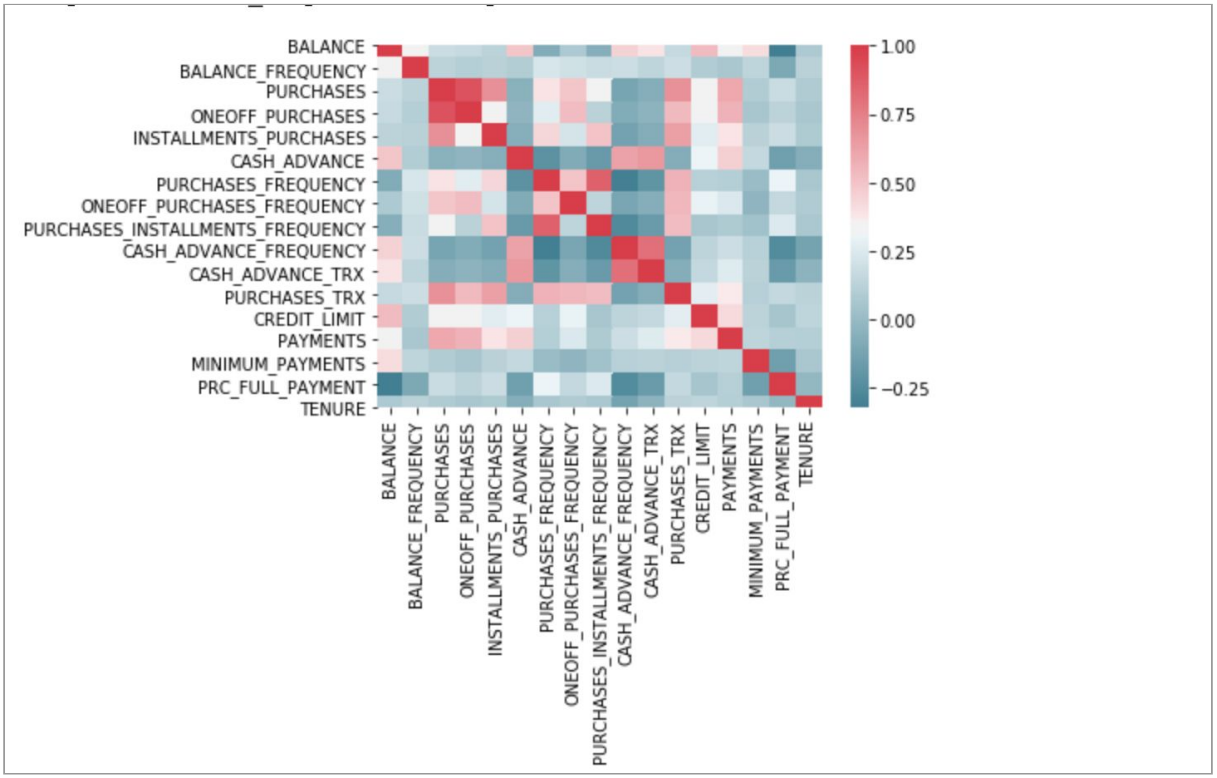
	_predicted_feature	_score	BALANCE	BALANCE_FREQUENCY	...	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
6	PURCHASES_FREQUENCY	0.990537	0.000780	0.000662	...	0.000757	0.000713	0.000381	0.001361
8	PURCHASES_INSTALLMENTS_FREQUENCY	0.967247	0.002446	0.000377	...	0.003000	0.002528	0.001304	0.002157
2	PURCHASES	0.935755	0.000723	0.000256	...	0.009756	0.000558	0.001655	0.000002
0	BALANCE	0.933432	0.000000	0.009265	...	0.013016	0.688152	0.008118	0.002415
...	...	...	...	...	...	...	...	...	...
15	PRC_FULL_PAYMENT	0.662871	0.387955	0.022931	...	0.113412	0.144568	0.000000	0.015452
10	CASH_ADVANCE_TRX	0.603961	0.022103	0.014797	...	0.055934	0.031329	0.003157	0.023213
12	CREDIT_LIMIT	0.440120	0.399634	0.015877	...	0.099121	0.086963	0.034882	0.013392
14	MINIMUM_PAYMENTS	0.313313	0.342701	0.012967	...	0.304488	0.000000	0.001953	0.023191

7 rows × 19 columns

You will notice that some features are predicted very well by other features. The features that are well predicted shows good relationship with other features with high importance relevance.

From the above table, we see that PURCHASES_FREQUENCY is the mostly predicted by the other features. However, PURCHASES_TRX appears to be the only relevant feature in predicting PURCHASES_FREQUENCY.

Let us look at the correlation of the features. We expect to see PURCHASES_FREQUENCY to be highly correlated with other features.



From the table above and as confirmed by the graph above, you will see that utmost 4 features are relevant in predicting another feature. An example is seen with CASH_ADVANCE, INSTALLMENTS_PURCHASES, ONE-OFF_PURCHASES and PURCHASES being relevant in predicting PAYMENTS

From an early observation, we cannot use all the features in our solution to the problem. We will have to carry further analysis in selecting important features for clustering.

Furthermore, graphs showing evenly distributed data points all over the plot area show non-correlation between the pairs. A good example is paring PRC_FULL_PAYMENT with either PURCHASES_FREQUENCY, ONEOFF_PURCHASES_FREQUENCY, PURCHASES_INSTALLMENTS_FREQUENCY, CASH_ADVANCE_FREQUENCY or TENURE

Good examples of correlation is pairing PURCHASES with either BALANCE, BALANCE_FREQUENCY, ONEOFF_PURCHASES, INSTALLMENTS_PURCHASES, PURCHASES_TRX, CREDIT_LIMIT or PAYMENTS. You will also see that PURCHASES can be easily predicted by other features. The features that correlate well with PURCHASES have high score in feature importance.

Algorithm and Techniques

We are looking to solve a clustering problem on data whose features values vary greatly in magnitude. The data we have not labelled, so clustering will enable us to label it and then demonstrate building a simple prediction model with it.

We will follow these steps in solving our problem:

1. Prepare our data by scaling and selecting useful features
2. Detect and remove outliers
3. Reduce our feature space
4. Clustering
5. Build a simple prediction model

K-Means

This is one of the simplest and faster clustering algorithms in use. However, it fails in local minima and several runs might produce different results. The objective of this algorithm is to discover similarities in data and group according to those similarities.

It infers from unlabelled dataset and tries to label the data.

You'll define a target number k, which refers to the number of centroids you need in the dataset. A centroid is the imaginary or real location representing the center of the cluster.

Every data point is allocated to each of the clusters through reducing the in-cluster sum of squares.

In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

The 'means' in the K-means refers to averaging of the data; that is, finding the centroid. [12, 13].

Agglomerative Clustering

This is a hierarchical clustering technique that uses bottom approach in connecting every single data point to another in order to make a cluster. Each data point starts its own cluster and clusters are successively merged together.

Spectral Clustering

These are techniques that use eigenvectors of matrices derived from data to come up with clusters.

C. H Martin [14] gives a very good overview of this technique. The goal of spectral clustering is to cluster data that is connected but not necessarily compact or clustered within convex boundaries.

The basic idea:

1. Project your data into R^n
2. Define an Affinity matrix A , using a Gaussian Kernel K or say just an Adjacency matrix (i.e. $A_{i,j} = \delta_{i,j}$)
3. Construct the Graph Laplacian from A (i.e. decide on a normalization)
4. Solve an Eigenvalue problem , such as $Lv = \lambda v$ (or a Generalized Eigenvalue problem $Lv = \lambda Dv$)
5. Select k eigenvectors $\{v_i, i = 1, k\}$ corresponding to the k lowest (or highest) eigenvalues $\{\lambda_i, i = 1, k\}$, to define a k-dimensional subspace $P^t LP$
6. Form clusters in this subspace using, say, k-means

This technique is preferred over K-Means when the clusters do not appear to be round or even sizes. In this case, the measure of centers or spread of these clusters cannot be a complete description of the clusters.

It has very simple implementation and good performance in many graph-based clustering.

DecisionTreeClassifier

Decision Trees is a non-parametric supervised learning technique used for regression and classification.
They are easy to understand and interpret and can be used numerical and categorical data. However, they are known to be susceptible to over-fitting.

Data Scaling

This is sometimes called feature scaling or data normalization. Some machine learning algorithms don't work well with features that varies significantly in range. A huge variation in a feature values will be biase a machine learning outcome. Therefore, all feature values are normalized so that selected features can contribute proportionately to the outcome.
We will apply a non-linear scaling using Yeo-Johnson[15] (I.K. Yeo and R.A. Johnson, 2000) transformation that reduces skewness.

Dimensionality Reduction

This is a technique for reducing feature space to enable machine learning algorithms perform better and faster. Data with a certain number of features can be reduced to have less.

We will use Principal Component Analysis (PCA) in reducing our feature space. PCA is a method used in dimensionality reduction. Many variables are reduced into few while containing much information in them.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. Because smaller data sets are easier to explore and visualize and make analyzing data much easier and faster for machine learning algorithms without extraneous variables to process [16]

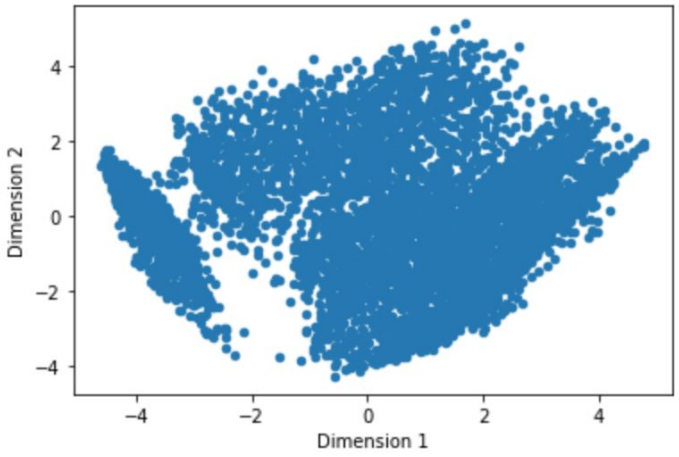
Benchmark

The benchmark for this solution is based on K-Means clustering with 3 clusters. The choice of hard clustering is informed by a scatter plot of our data that has been transformed into to having only 2 dimensions.

Looking at the figure below, it is easy to conclude that the data has 3 clusters of uneven size and non-radial shape.

```
# Visualize reduced data
reduced_data.plot.scatter(x="Dimension 1", y="Dimension 2")
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x1272bbc18>
```



However, we ended up determining the scores for other numbers of clusters.

```
# benchmark model
N_CLUSTER_TRIALS = range(2, 6)

def get_cluster_and_score(n_clusters):
    kmeans = cluster.KMeans(n_clusters=n_clusters)
    kmeans.fit(reduced_data)
    pred = kmeans.predict(reduced_data)
    score = silhouette_score(
        reduced_data, pred)
    return score

trials = {x: get_cluster_and_score(x) for x in N_CLUSTER_TRIALS}

display(pd.DataFrame(
    data={"Cluster {}".format(x): [y] for x,y in trials.items()},
    index=["Scores"]
))
```

	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Scores	0.473172	0.452947	0.469355	0.432473

You will notice that K-Means with 2 clusters has a better silhouette score, this is also true with 4 clusters. Despite this, since we are conducting hard clustering (determining clusters of definite number), then we will settle with 3 clusters.

We will use this as just a guide only to our next section of work because, a fact, clusters have non-radial shapes and their size are uneven.

So, next, we will look for algorithm that is sensitive to the shape of the clusters.

III. Methodology

Data Preprocessing

Data Normalization

The figure below shows our data distribution:

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY	C
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	

Since our data is not normally distributed, i.e the mean and median vary significantly (indicating a large skew), we will apply a non-linear scaling using Yeo-Johnson[15] (I.K. Yeo and R.A. Johnson, 2000) transformation that reduces skewness.

I would have used Box-Cox [17] transformation but our data has zeros. Yeo-Johnson transformation is able to transform data that zeros and negative numbers.

You will notice from the figure below that our data has been transformed with standard deviation being reduced significantly to signify a normalized data.

```
pt = PowerTransformer(method="yeo-johnson", standardize=True)
scaled_data_ndarray = pt.fit_transform(raw_data_without_id)
scaled_data = pd.DataFrame.from_records(scaled_data_ndarray, columns=raw_data_without_id.columns)
```

```
scaled_data.describe()
```

	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASES_FREQUENCY
count	8.055000e+03	8.055000e+03	8.055000e+03	8.055000e+03	8.055000e+03	8.055000e+03	8.055000e+03
mean	-3.881301e-17	-6.457074e-16	3.528456e-17	-5.028049e-17	-4.763415e-17	-4.145935e-17	1.764228e-17
std	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00	1.000062e+00
min	-2.124192e+00	-2.002307e+00	-1.510734e+00	-1.002051e+00	-1.091958e+00	-9.385686e-01	-1.284667e+00
25%	-8.210843e-01	-6.114449e-01	-6.387413e-01	-1.002051e+00	-1.091958e+00	-9.385686e-01	-1.026975e+00
50%	1.535537e-01	6.268625e-01	1.614421e-01	2.388183e-01	3.599591e-01	-9.385686e-01	1.005076e-01
75%	7.205233e-01	6.268625e-01	7.062814e-01	9.908086e-01	9.018661e-01	1.064359e+00	1.041255e+00
max	2.721945e+00	6.268625e-01	3.564315e+00	1.959567e+00	2.225352e+00	1.723131e+00	1.213207e+00

Outliers Detection and Removal

This forms an integral part of data pre-processing. We remove data that tend to skew the distribution of data. The presence of the data points may lead to skewed results.

K-Means clustering is very sensitive to outliers. Therefore, it is important we can remove the outliers from our data if we intend to use K-Means as our benchmark model.

We used Tukey's Fences [18] in identifying outliers. Tukey's rules:

- The first quartile Q1 is the value ≥ 1/4 of the data
- The second quartile Q2 or the median is the value ≥ 1/2 of the data,
- The third quartile Q3 is the value ≥ 3/4 of the data.

Therefore, The interquartile range, IQR, is Q3 − Q1. Tukey's rule says that the outliers are values more than 1.5 times the interquartile range from the quartiles either below Q1 − 1.5IQR, or above Q3 + 1.5IQR. [19]

```
outliers_, outliers_count_ = otl.detect(scaled_data)

print("Rows that are outliers: ", len(outliers_), "\n")

print("Feature rows that are outliers:\n")
for feat, count in outliers_count_.items():
    print("\t{}: {}".format(feat, count))
```

```
Rows that are outliers:  1524

Feature rows that are outliers:

        PURCHASES: 21
        CREDIT_LIMIT: 8055
        PAYMENTS: 356
        MINIMUM_PAYMENTS: 8055
        TENURE: 1235
```

```
# Check if our samples are among the outliers
for index in sample_indices:
    if index in outliers_:
        print("Index {} is an outlier".format(index))
```

```
# Remove outliers from our data
scaled_data_with_id = scaled_data.copy()
scaled_data_with_id.insert(0, "CUST_ID", raw_data["CUST_ID"])
outliers_df = pd.DataFrame(scaled_data.loc[outliers_], columns = scaled_data.keys())
good_data_df = scaled_data_with_id.drop(scaled_data_with_id.index[outliers_])

# I think we can disregard `CREDIT_LIMIT` and `MINIMUM_PAYMENTS` since their data points seem unrelated
good_data = good_data_df.drop(['CUST_ID','CREDIT_LIMIT', 'MINIMUM_PAYMENTS'], axis=1, inplace=False)
```

Implementation

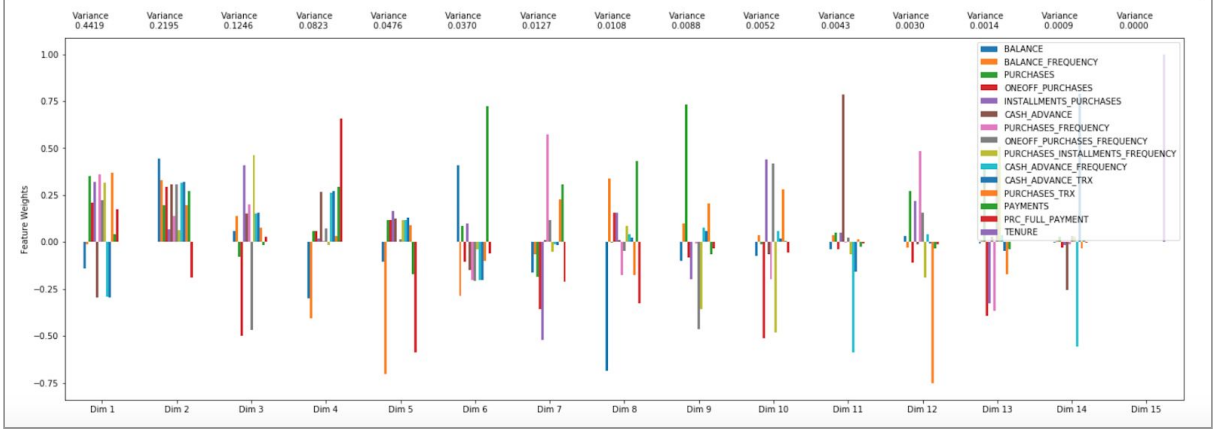
Feature Transformation

We are going to use Principal Component Analysis (PCA) for reducing our dimension.

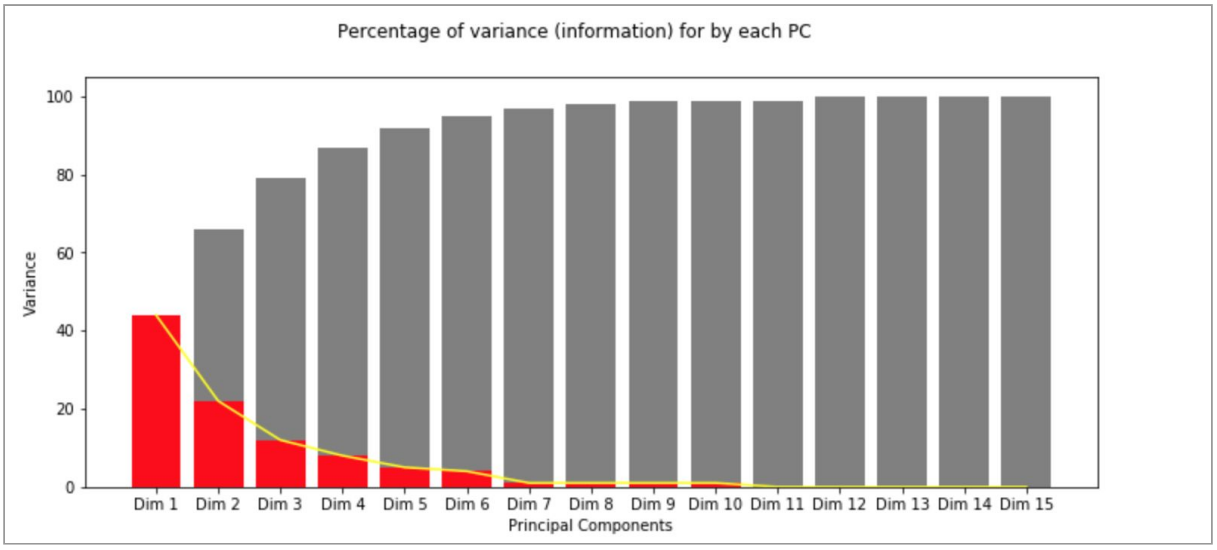
```
pca = PCA(n_components=good_data.shape[1])
pca.fit(good_data)

# Transform samples data using the PCA fit above
samples_scaled_2 = samples_scaled.drop(['CREDIT_LIMIT', 'MINIMUM_PAYMENTS'], axis=1)
pca_samples = pca.transform(samples_scaled_2)
```

```
# Generate PCA results plot
pca_results = vs.pca_results(good_data, pca)
```

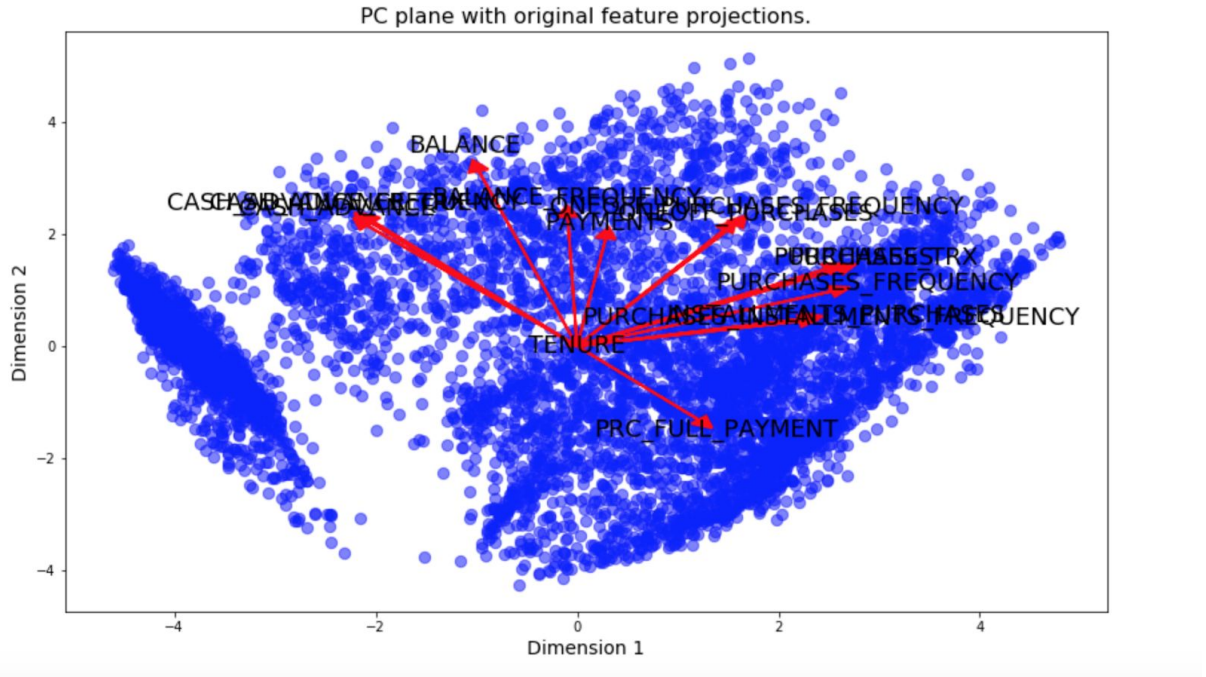


The graph above shows variance between our principal components and the original features.



From the above graph, we can easily pick the first 2 dimensions to represent the features. The total variance covered by the two dimensions is about 65%.

Therefore, we can use 2 dimensions in represent the 15 features.



From the biplot above you will notice that there is one cluster that spends more while another is liquid in terms of cash and balance. The cluster on the bottom-right does a lot of purchasing as evidenced by huge variance in purchase related features. The cluster in the top maintains good balance in the card, pays the card issuer regularly and purchase with cash often.

There is a cluster on the left that has similarities with the top one. However, for this cluster, it seems they use cash a lot with less payments back to the issuer. And also seems to not be purchasing or using the credit card.

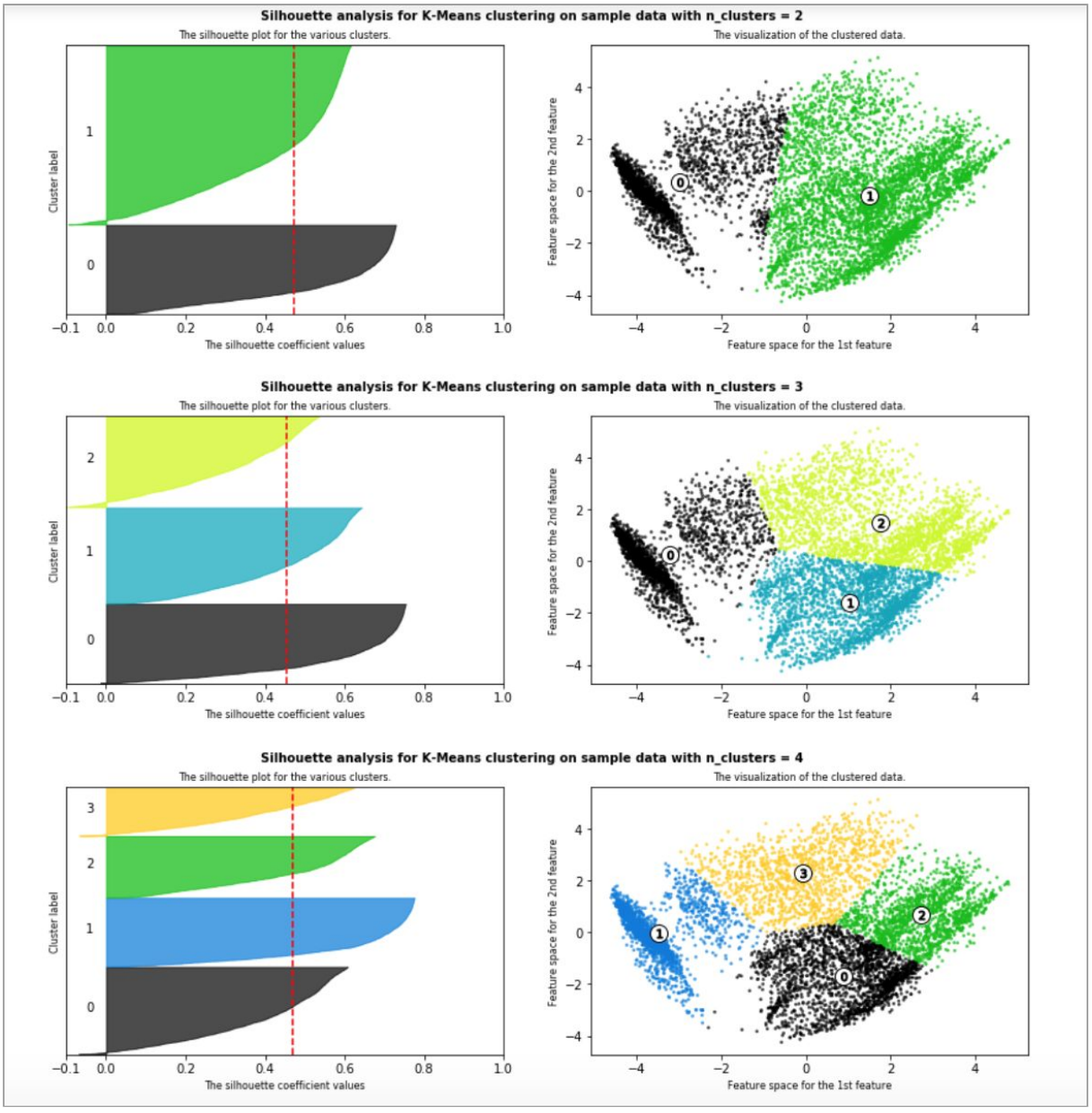
Clustering

We will solve our problem by hard clustering (A bias of needing 3 clusters). This is evidenced from our scatter plots above.

From the plot, it is also obvious that our clusters are non-radial with uneven sizes which makes K-Means unsuitable to use.

Conducting silhouette analysis, we found that K-Means performs better than other algorithms. However, the assumption is that center of these clusters provide meaningful information and that the cluster have even sizes.

The visualizations below shows the silhouette analysis.



So we are going to compare K-Means with other algorithms as shown in the figure below.

Since we are interested to find 3 clusters then silhouette analysis is more ambivalent in deciding between K-Means and Spectral Clustering with 0.453 and 0.452 scores respectively.

K-Means performs better in flat geometry and with even cluster size. Therefore, we will settle for SpectralClustering as our algorithm of choice.

We need to optimize it such that it can fit our clusters properly..

```python
def get_cluster_and_score_cls(clf: tuple) -> list:
    result = []
    for algorithm in clf:
        algorithm.fit(reduced_data)
        if hasattr(algorithm, 'labels_'):
            y_pred = algorithm.labels_.astype(np.int)
        else:
            y_pred = algorithm.predict(X)
        score = silhouette_score(reduced_data, y_pred)
        result.append(score)
    return result

def get_clfs(n_clusters: int) -> dict:
    return (
        cluster.KMeans(n_clusters, random_state=RANDOM_STATE),
        cluster.AgglomerativeClustering(n_clusters=n_clusters, linkage="average", affinity="euclidean"),
        cluster.SpectralClustering(n_clusters=n_clusters, random_state=RANDOM_STATE)
    )

ctrials = {x: get_cluster_and_score_cls(get_clfs(x)) for x in N_CLUSTER_TRIALS}

display(pd.DataFrame(
    data={"Cluster {}".format(x): y for x,y in ctrials.items()},
    index=["K-Means", "AgglomerativeClustering", "SpectralClustering"]
))
```

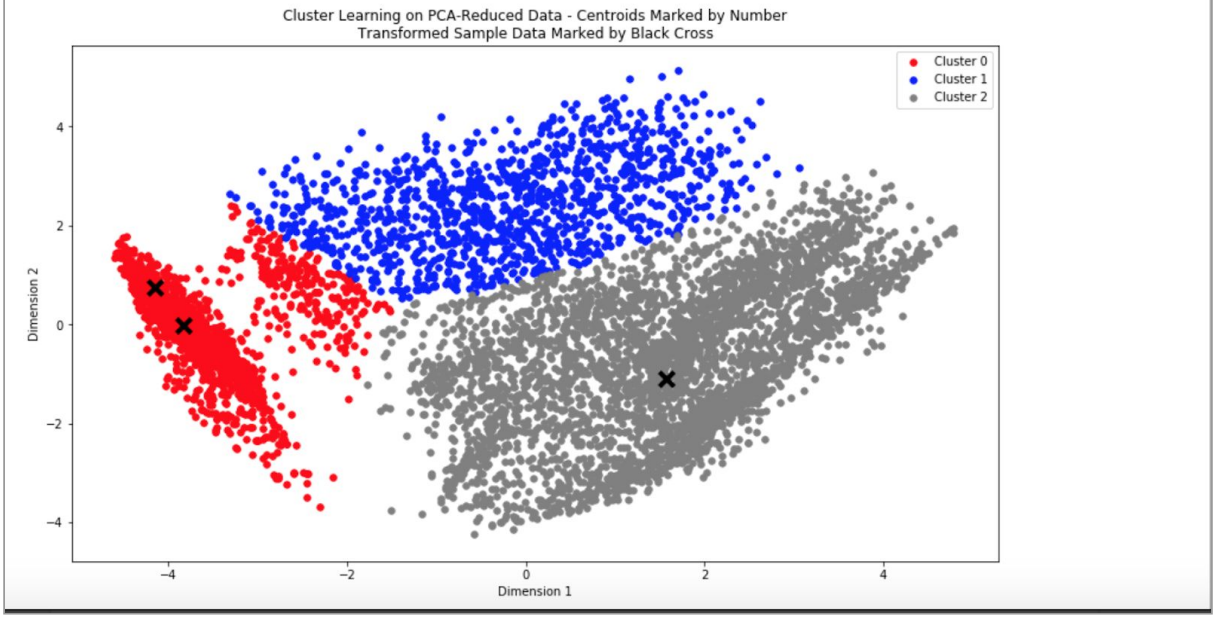
	Cluster 2	Cluster 3	Cluster 4	Cluster 5
K-Means	0.473289	0.452947	0.469355	0.432475
AgglomerativeClustering	0.477017	0.431731	0.448804	0.406967
SpectralClustering	0.473021	0.451762	0.458073	0.436167

Refinement

SpectralClustering with default parameters and 3 clusters gave us a score of 0.45.

```python
# # Display the results of the clustering from implementation
pre_alg = cluster.SpectralClustering(n_clusters=3, random_state=RANDOM_STATE)
pre_y_pred = pre_alg.fit_predict(reduced_data)
print("Score: ", silhouette_score(reduced_data, pre_y_pred))
vs.cluster_results(reduced_data, pre_y_pred, reduced_data_sample.to_numpy())
```

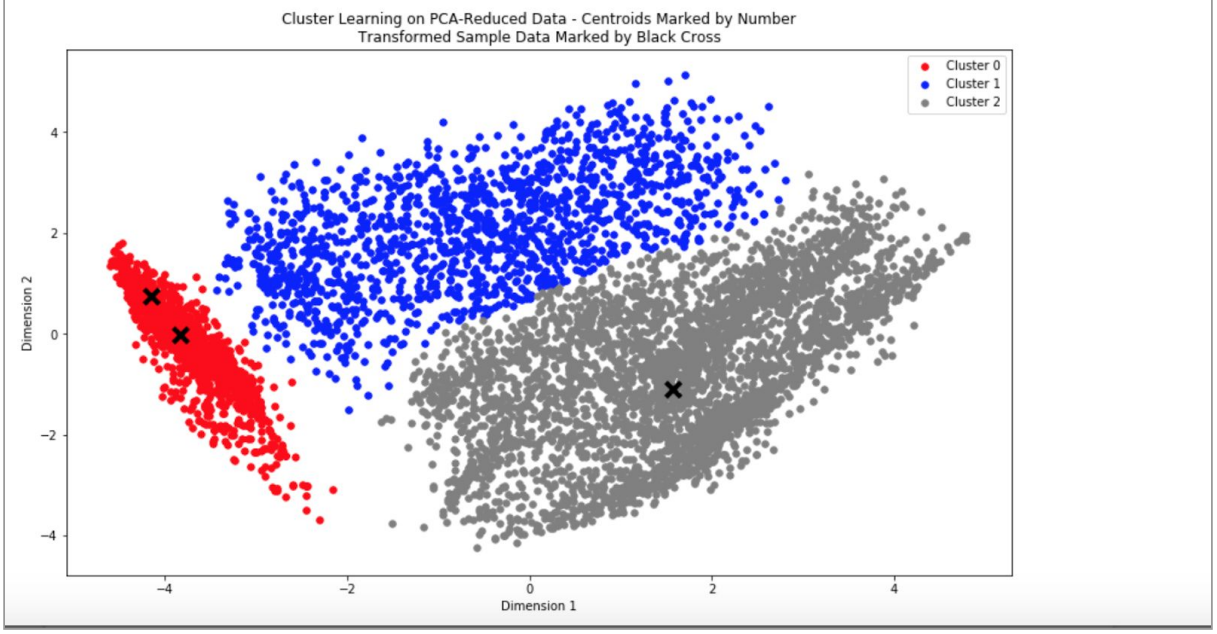
```
Score:  0.45176192126272646
```



Even with the above score, it seems that clusters are not well separated. Therefore, we will adjust the gamma (radial basis function coefficient) parameter of the algorithm.

```
# # Display the results of the clustering from implementation
alg = cluster.SpectralClustering(n_clusters=3, gamma=50, random_state=RANDOM_STATE)
y_pred = alg.fit_predict(reduced_data)
print("Score: ", silhouette_score(reduced_data, y_pred))
vs.cluster_results(reduced_data, y_pred, reduced_data_sample.to_numpy())
```

```
Score:  0.4351234999178268
```



We now have a score of 0.43. This change doesn't improve our score. However, we can see that clusters are well defined.

IV. Results

Model Evaluation and Validation

Unfortunately, our observation bias supersedes our metric. We have settled for a lower silhouette score with a well defined cluster (through observation), partly because we feel our metric does not measure the connectedness of our cluster members.
Silhouette score of a cluster is influenced externally by another. If non-radial clusters, some members (if not many) of one cluster might be very close another cluster, this reduces the global average score our clusters.

Therefore, we will use our clusters whose silhouette score is 0.43 to build a simple prediction model and use this.

We now have 6531 labelled out of 8950 unlabelled records to use for our prediction.

Prediction can be easily done by a supervised learning technique. We are going to re-use the already curated data to build a simple prediction model.

```
# Lets build a simple models using Rand
from sklearn.model_selection import train_test_split
from sklearn.metrics import r2_score
from sklearn.tree import DecisionTreeClassifier

cluster_labels = y_pred # Re-using y_pred from clustering since these are our labels
features = reduced_data # Re-using PCA transformed data earlier, these are our reduced feature space
X_train, X_test, y_train, y_test = train_test_split(
    features, cluster_labels, test_size=0.20, random_state=RANDOM_STATE)

clf = DecisionTreeClassifier(random_state = RANDOM_STATE)
clf.fit(X_train, y_train)

# calculating the coefficient of determination, R2, to quantify your model's performance
print("Performace score of {:.2f}%".format(clf.score(X_test, y_test) * 100))
```

```
Performace score of 99.23%
```

Notwithstanding overfitting problems, our data can be easily predicted.

Justification

A higher prediction score, $\mathbf{R^2}$, of 99% shows that we have been able to identify some patterns in our data. Even with a silhouette score of 0.43, we have been able to assign reproducible label to a data point.

Reproducibility is a big test on discovered patterns that informed the formation of the 3 clusters.

V. Conclusion

Free-Form Visualization

The goals of the project was to be able to derive meaning-full clusters that can by the card issuers.

We picked 3 sample data points to track during our experiment. The following table shows our samples:

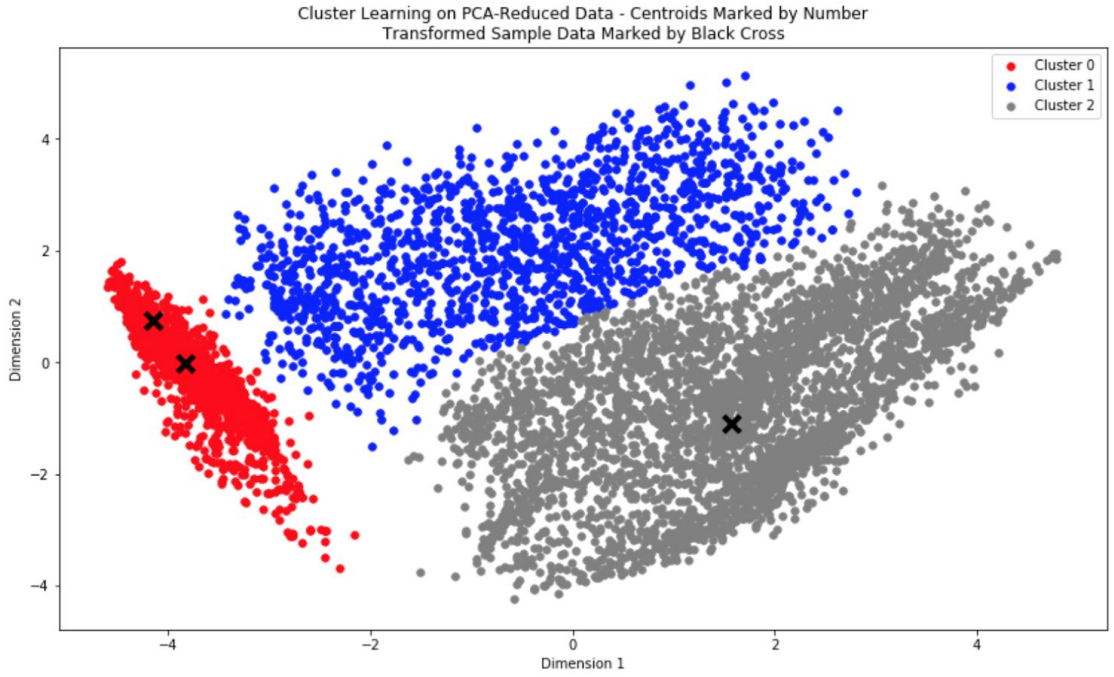
	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	...	PAYMENTS	MINIMUM_PAYMENTS	PRC_FULL_PAYMENT	TENURE
265	C11411	5745.789490	1.0	0.00	...	1684.800793	1779.683281	0.0	12
1043	C17776	378.320894	1.0	315.02	...	206.090316	202.391607	0.0	12
2627	C15038	1129.185643	1.0	0.00	...	1275.071841	293.203915	0.0	12

3 rows × 18 columns

The our our samples reduced to only 2 dimensions:

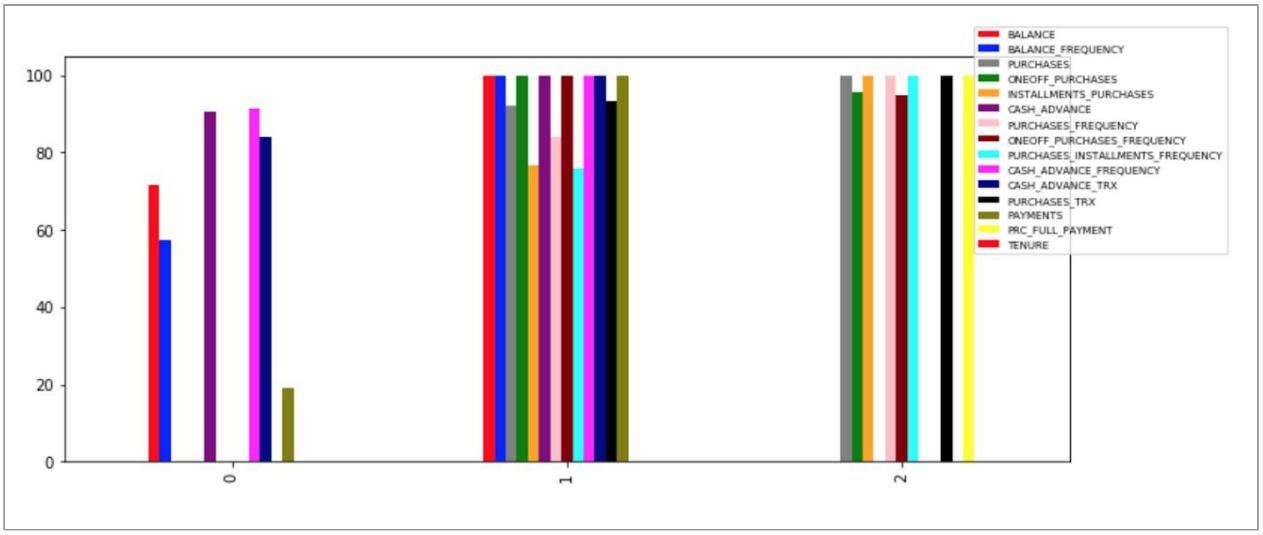
	Dimension 1	Dimension 2
265	-4.147725	0.758747
1043	1.567826	-1.093872
2627	-3.832400	-0.016114

Our samples appeared in the 3 clusters. See below:



Sample row numbered **265** and **2627** appeared in **Cluster 0** while sample **1043** appeared in **Cluster 2.**

Let us see how our 3 clusters fair on the original features.



From the above we can have the following conclusions:

Cluster 0: Low-Income-Earners

This group will make regular repayments to ensure that they have good credit score. Regular repayments will ensure they have relatively high balances.

This group gets paid in cash and can therefore make cash-advance purchases.

Sample **265** and **2627** are in this cluster.

Cluster 1: High-Income-Earners

They maintain a high balance in their card.
They have high purchases as well.
They are liquid financially and can therefore purchase in cash.
Due to high activity in their cards, they make regular repayments to the card issuer.

Cluster 2: Middle-Income-Earners

They prefer to purchase in cash, cash advance or in installments. Because of this, they will clear all their credit card dues in one full repayment.

Therefore, you will find that their credit card has a low balance and the frequency of updating their credit card balance is low.

Sample **1043** is in this cluster.

Even our simple prediction was able to assign these 3 samples correctly to their clusters.

	Group	CUST_ID	BALANCE	BALANCE_FREQUENCY	PURCHASES	ONEOFF_PURCHASES	INSTALLMENTS_PURCHASES	CASH_ADVANCE	PURCHASE
265	Low-Income-Earners	C11411	5745.789490	1.0	0.00	0.0	0.00	965.726556	
1043	Middle-Income-Earners	C17776	378.320894	1.0	315.02	0.0	315.02	0.000000	
2627	Low-Income-Earners	C15038	1129.185643	1.0	0.00	0.0	0.00	1385.126221	

Reflection

The raw data that we used in this project was ambiguous as most of it had been scaled. Therefore, understanding it and using it in explaining our results took long and hard at the same time.

We seemed to have non-radial clusters which silhouette score seemed to be not appropriate in our model evaluation.

Nevertheless, I enjoyed the project and made me learn a lot. Notwithstanding the limited time that I had, completing the project was hard and took a toll on me.

Improvement

As much as we have shown that our clusters are perfect can be used for prediction, the following areas in my opinion felt short and can be improved:

1. We could have used other metrics that not only looks at compactness of the cluster such as silhouette coefficient but also connectivity and separation. We need to find a better metric for non-radial and uneven clusters. We could try and use Dunn Index [20]
2. It would be interesting to subject to a lot of data and see how it performs. I think approximately 8000 data points was not enough.
3. Under optimization in our algorithms to see how they can perform. I under-tuned SpectralClustering algorithm.

VI. References

[1] - O'Sullivan, Arthur; Steven M. Sheffrin (2003). Economics: Principles in action (Textbook). Upper Saddle River, New Jersey 07458: Pearson Prentice Hall. p. 261. ISBN 0-13-063085-3.

[2] - Mathur, Varun (2018). How Credit Card Issuers Can Gain from Machine Learning. https://www.publicissapient.com/news/How-credit-card-issuers-can-gain-from-machine-learning. Accessed on June 17, 2019.

[3] - Kaggle - (https://www.kaggle.com)

[4] - Bhasin, Arjun (2018). Credit Card Dataset for Clustering. https://www.kaggle.com/arjunbhasin2013/ccdata. Accessed on June 17, 2019

[5] - Jupyter - (https://jupyter.org/)

[6] - Wikipedia - (https://en.wikipedia.org/wiki/K-means_clustering)

[7] - Peter J. Rousseeuw (1987). Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis. Computational and Applied Mathematics 20: 53-65. https://www.sciencedirect.com/science/article/pii/0377042787901257

[8] - Wikipedia - (https://en.wikipedia.org/wiki/Silhouette_(clustering))

[9] - Hastie, Trevor; Tibshirani, Robert (2009). The Elements of Statistical Learning: Data mining, Inference, and Prediction. New York: Springer. pp. 485–586. https://web.stanford.edu/~hastie/Papers/ESLII.pdf. Accessed on July 29. 2019

[10] - Scikit-Learn - (https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html)

[11] - Stat Trek. Coefficient of Determination - https://stattrek.com/statistics/dictionary.aspx?definition=coefficient_of_determination. Accessed on July 29. 2019

[12] - Garbade J.M (2013). Understanding K-means Clustering in Machine Learning. Towards Data Science. https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1. Accessed on July 29, 2019.

[13] - Piech C (2012). K Means. https://stanford.edu/~cpiech/cs221/handouts/kmeans.html. Accessed on July 29, 2019.

[14] - C.H Martin (2012). SPECTRAL CLUSTERING: A QUICK OVERVIEW. https://calculatedcontent.com/2012/10/09/spectral-clustering/. Accessed on July 13.

[15] - Yeo and R.A. Johnson, “A new family of power transformations to improve normality or symmetry.” Biometrika, 87(4), pp.954-959, (2000)

[16] - Zakaria Jaadi. A step by step explanation of Principal Component Analysis. Feb 28, 2019. https://towardsdatascience.com/a-step-by-step-explanation-of-principal-component-analysis-b836fb9c97e2. Accessed on Aug 1, 2019.

[17] - G.E.P. Box and D.R. Cox, “An Analysis of Transformations”, Journal of the Royal Statistical Society B, 26, 211-252 (1964)

[18] - Tukey, John W (1977). Exploratory Data Analysis. Addison-Wesley. ISBN 978-0-201-07616-5. OCLC 3058187. https://archive.org/details/exploratorydataa00tuke_0. Accessed on Aug 1, 2019.

[19] - C. Shalizi (2013). Lab 5: Testing Our Way to Outliers. https://www.stat.cmu.edu/~cshalizi/statcomp/13/labs/05/lab-05.pdf. Accessed on Aug 1, 2019

[20] - Dunn, J. C. (1973-09-17). "A Fuzzy Relative of the ISODATA Process and Its Use in Detecting Compact Well-Separated Clusters". Journal of Cybernetics. 3 (3): 32–57. https://www.tandfonline.com/doi/abs/10.1080/01969727308546046. Accessed on Aug 11, 2019.