Credit Card Default

Clustering Analysis

Main Objective of Analysis

Default is a serious credit card status that happens when an account holder becomes severely delinquent on payments and it affects credit standing and the likelihood of getting approved for other credit-based services. As of January 2020, the default rate for credit cards was 3.28%. Using various clustering models, the goal of this analysis was to determine which model provided the best results based on comparing the values in each cluster in the models to the actual values for credit card account holders who are in default from the selected dataset. In this case, the clusters represented the education level of the credit card account holders. These clustering models can then be handed over to decision makers at the credit card company for customer segmentation purposes and to better understand demographic information and assess the risk of default.

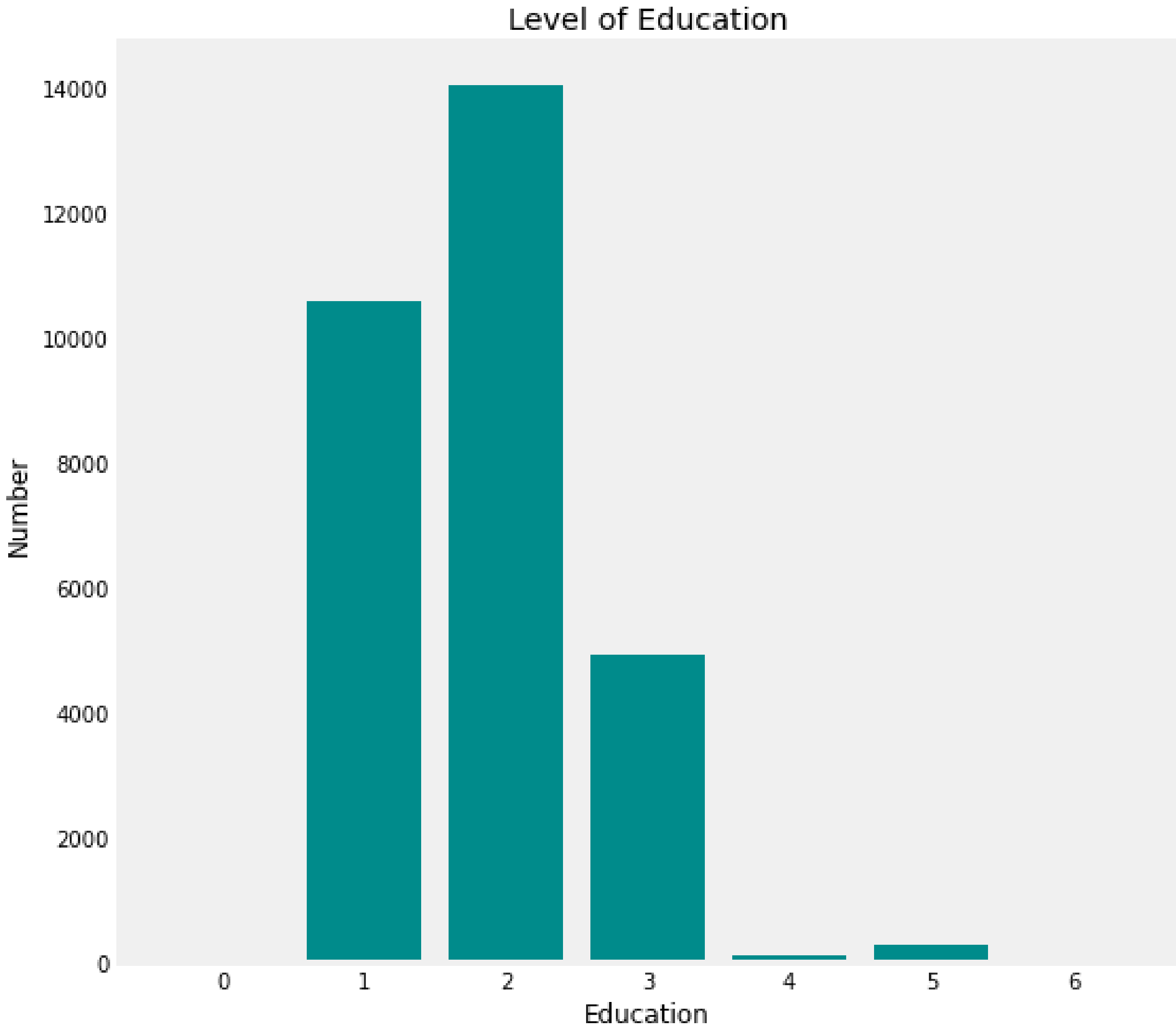
Description of Dataset

The dataset is from an anonymous source with sensitive personal information like name, birthdate, and credit card number excluded. It is a sample of 30,000 accounts and includes demographic information such as gender, education level, marital status, and age. In addition, there is information on spending and payment habits and the balance limit for a particular account. The datatypes are all numeric, either integer or float values:

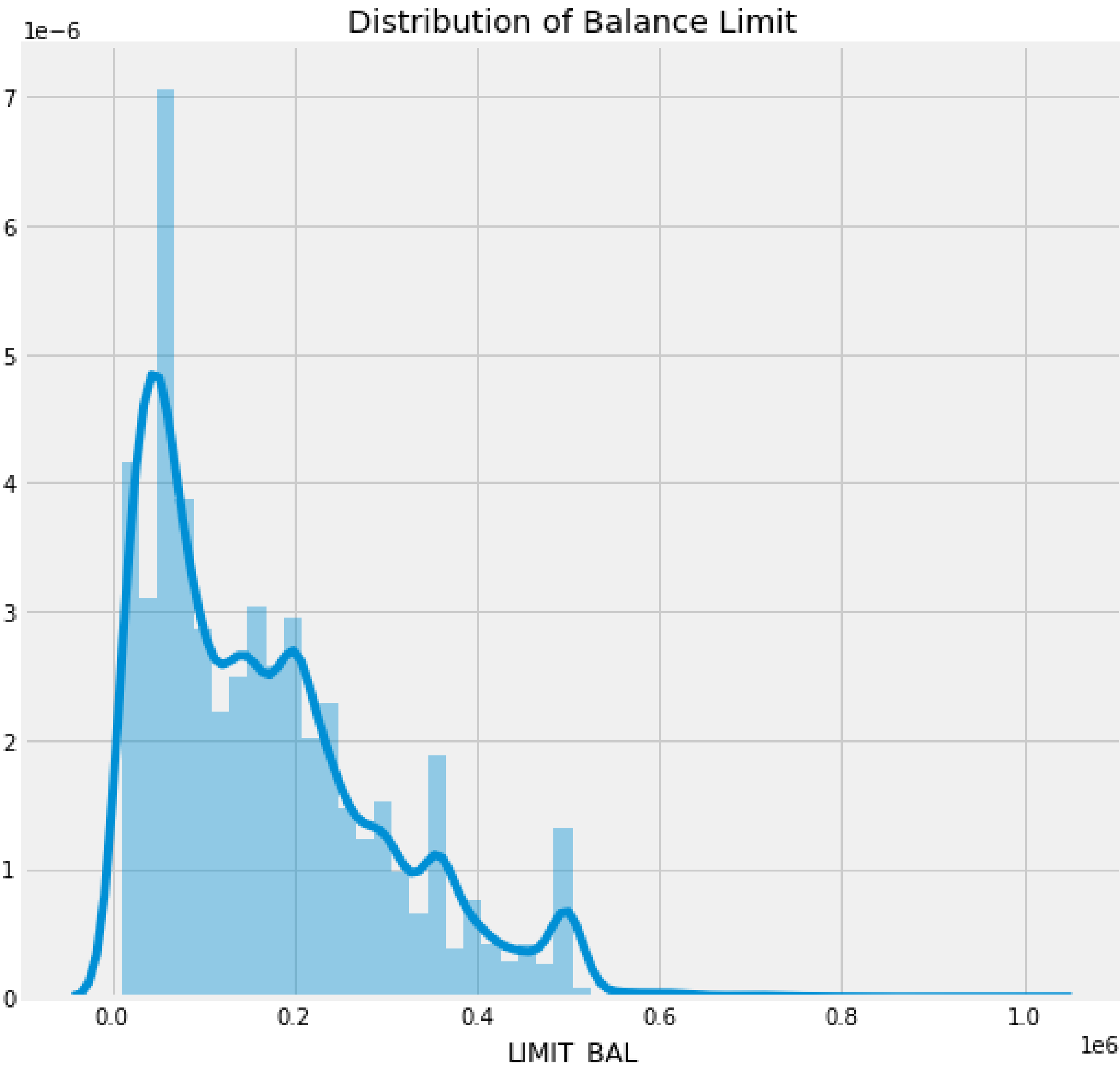
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 25 columns):
 #   Column                      Non-Null Count  Dtype
---  ------                      --------------  -----
 0   ID                          30000 non-null  int64
 1   LIMIT_BAL                   30000 non-null  float64
 2   SEX                         30000 non-null  int64
 3   EDUCATION                   30000 non-null  int64
 4   MARRIAGE                    30000 non-null  int64
 5   AGE                         30000 non-null  int64
 6   PAY_0                       30000 non-null  int64
 7   PAY_2                       30000 non-null  int64
 8   PAY_3                       30000 non-null  int64
 9   PAY_4                       30000 non-null  int64
 10  PAY_5                       30000 non-null  int64
 11  PAY_6                       30000 non-null  int64
 12  BILL_AMT1                   30000 non-null  float64
 13  BILL_AMT2                   30000 non-null  float64
 14  BILL_AMT3                   30000 non-null  float64
 15  BILL_AMT4                   30000 non-null  float64
 16  BILL_AMT5                   30000 non-null  float64
 17  BILL_AMT6                   30000 non-null  float64
 18  PAY_AMT1                    30000 non-null  float64
 19  PAY_AMT2                    30000 non-null  float64
 20  PAY_AMT3                    30000 non-null  float64
 21  PAY_AMT4                    30000 non-null  float64
 22  PAY_AMT5                    30000 non-null  float64
 23  PAY_AMT6                    30000 non-null  float64
 24  default.payment.next.month  30000 non-null  int64
dtypes: float64(13), int64(12)
memory usage: 5.7 MB
```

Exploratory Data Analysis, Data Cleaning, and Feature Engineering

The clusters from the models attempt to represent the education levels of the dataset for credit card default. Unfortunately, the dataset labels level of education in integer values rather than clearly stating labels such as “high school,” “2 year college,” or “graduate school,” but the values are ordinal, so information can at least be inferred from lowest to highest levels of education. The highest count of credit card default is in the third from the bottom education level and as education level increases, the number of default decreases. Perhaps this is due to higher income typical of individuals who have attained more formal education.



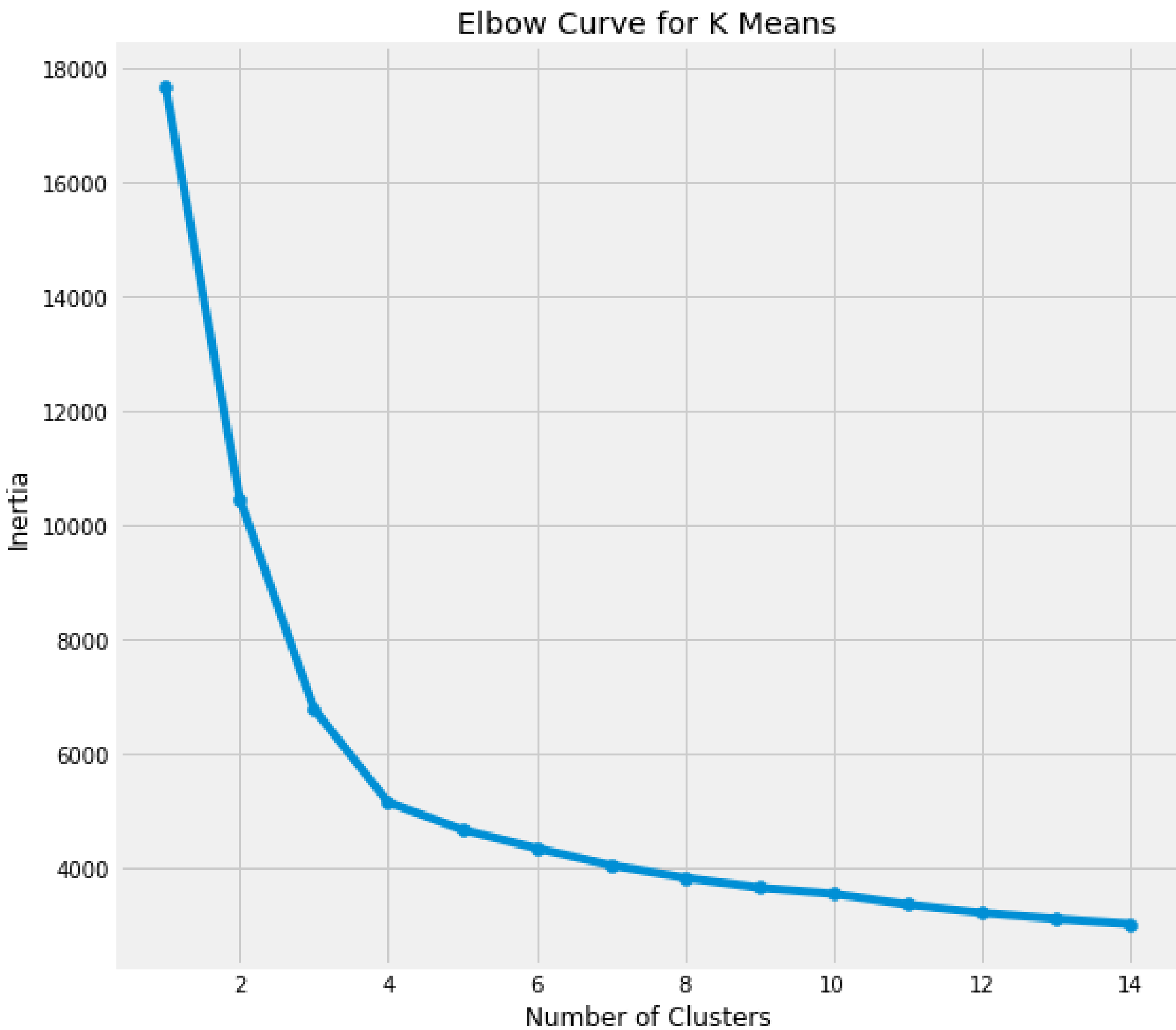
Interestingly, the distribution in balance limit shows that account holders with lower limits tend to default more than those with higher limits. A lower balance limit indicates that the account holder most likely had lower credit standing to begin with, and that could mean that their responsibility and accountability in reliably making payments are more of a risk.



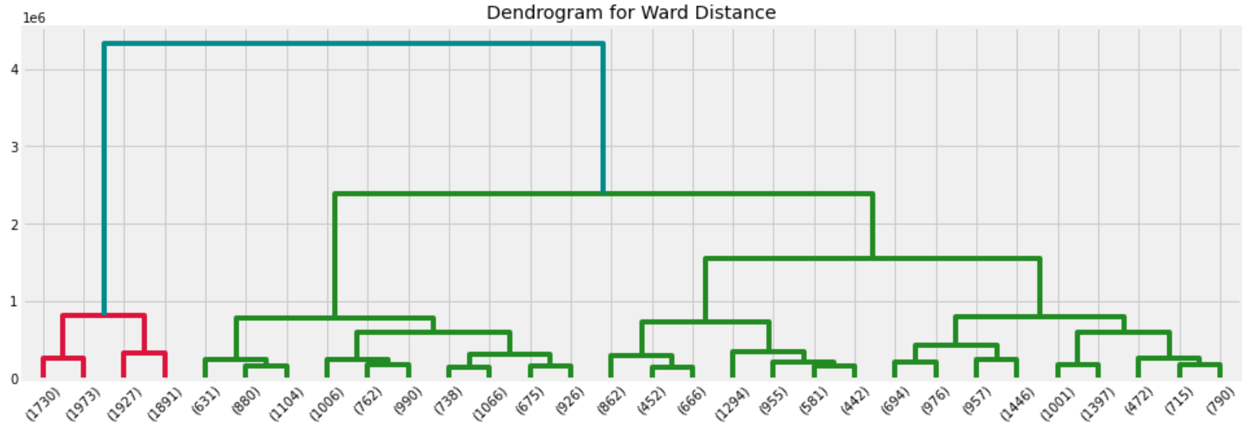
No null values were present in the dataset and because all of the variables were already numerical, encoding objects was unnecessary. The only column dropped was the ID number, which provided no meaningful information and would have only served to complicate clustering. For fitting the clustering models, the education column was dropped and saved into a separate DataFrame to be later used to compare the performance of the models. `MinMaxScaler()` was used so all of the data would be in the same scale, as the ranges of values varied for each variable.

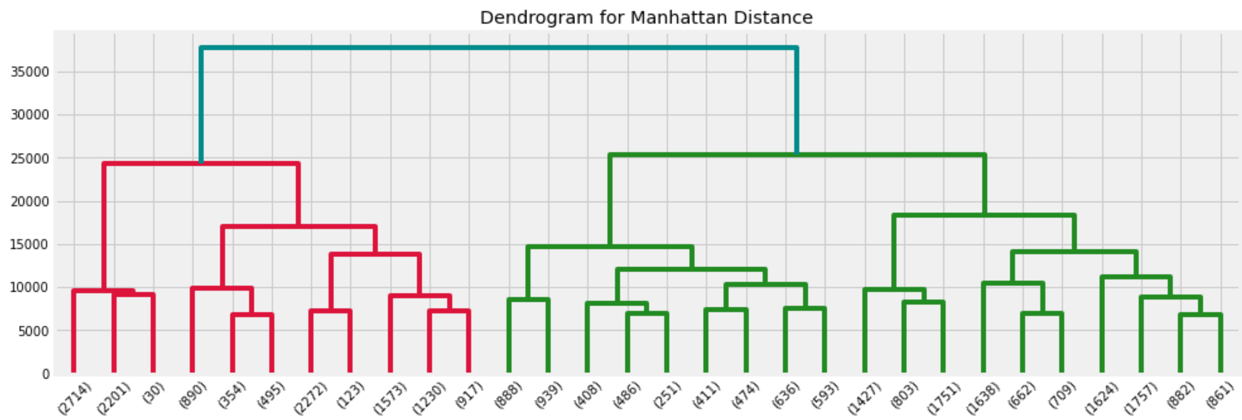
Clustering Models

For the purposes of this analysis, the following clustering models were used: K-Means, Hierarchical Agglomerative Clustering, Density-Based Spatial Clustering of Applications with Noise (DBSCAN). For K Means, an elbow curve was generated to observe inertia over 15 clusters:



The original dataset had 7 different categories for education levels and as indicated in the elbow curve, 7 clusters appear to be reasonable for a good inertia value, so a K Means model with `n_clusters=7` was chosen to fit the model. The same number of clusters was used for two models of Agglomerative Clustering, one with `affinity='euclidean'` and `linkage='ward'` and the other with `affinity='manhattan'` and `linkage='average'`. Dendrograms were generated for each Agglomerative Clustering model:





Finally, a DBSCAN model was fitted with `eps=3` and `min_samples=2`. The labels assigned from each model were consolidated into one DataFrame that also included a column for the actual education level values from the original dataset:

	EDUCATION	kmeans	agg	agg_manhattan	db
0	2	2	0	0	0
1	2	2	0	0	0
2	2	1	6	2	1
3	2	4	5	4	2
4	2	6	4	1	0
5	1	0	3	2	1
6	1	0	3	2	1
7	2	1	6	2	1
8	3	4	5	4	2
9	3	0	3	2	1

Percentages of correctly clustered data points from each model were then calculated and compared to each other:

```
Percentages Correct for Models
------------------------------
KMeans: 16.52
Agglomerative (Euclidean): 13.47
Agglomerative (Manhattan): 20.77
DBSCAN: 15.31
```

All models performed poorly and even when selecting 7 clusters for K Means and Agglomerative Clustering, the generated clusters did not accurately represent the actual labels in the dataset for education level. What is interesting to note is that the Agglomerative Clustering model that used Manhattan distance performed significantly better than when using the same algorithm using Euclidean distance, which makes sense given Manhattan distance is used in business cases with high dimensionality. The dendrograms visualize how the two models differ. In this analysis, Agglomerative Clustering with Manhattan distance performed the best, with K Means second and interestingly, Agglomerative Clustering with Euclidean distance the worst.

Key Findings and Insights

The clustering algorithms selected fitted models that did not perform well on this dataset for credit card default. Even with 7 clusters chosen for K Means and Agglomerative Clustering, representative of the number of different education levels from the original dataset, most of the observations were mislabeled. While Agglomerative Clustering with Manhattan distance performed the best out of all of the models, the same algorithm using Euclidean distance performed the worst. The dendrograms for the two Agglomerative Clustering models differ in structure as well as in scale. And as expected, the Agglomerative Clustering models took the most time to train. Because these clustering models performed so poorly, they cannot be used in the future for practical customer segmentation purposes for credit card default.

Next Steps

Many improvements can be made to improve clustering from this dataset. A potential pitfall for this analysis could have been the method for scaling. Rather than `MinMaxScaler()`, perhaps `StandardScaler()` or `RobustScaler()` could have been used. Also, different hyperparameters could have been chosen, but keeping the number of clusters at 7 is most reasonable. Perhaps the hyperparameters for the DBSCAN algorithm can be tweaked the most, but as there are two of them, this could pose a challenge. Finally, Mean Shift, which is an algorithm that was not included in this analysis could be utilized with potential performance advantages.