Market Customer Segmentation with SVM and Random Forest Classifiers and K-Means Clustering

Abstract

Customer segmentation is the practice of classifying consumers into groups based on shared traits so that businesses may effectively and appropriately sell to each group. Customers are divided into groups based on common behaviors and customs.

1. Introduction

Understanding how the target market should be and how their clients should utilize their services should be a top priority for any organization. Each customer may utilize a company's services in a unique way. Thus, identifying the target market for a particular product/ service is the issue we're attempting to tackle. The identification of target audiences / consumer demographics will facilitate an increase in selling rates of particular services with a higher customer satisfaction. Using the above information, companies can then outperform their existing competition by devising uniquely appealing products/services.

The dataset we made use of was a publicly available dataset that was originally linked within a hackathon hosted by Analytics Vidhya: [datahttps://datahack.analyticsvidhya.com/contest/janatahack-customer-segmentation/#About](https://datahack.analyticsvidhya.com/contest/janatahack-customer-segmentation/#About) and was downloaded locally.
The dataset is now also available at Kaggle. We've used two separate modeling approaches for our problem: Supervised learning approaches such as SVM and RandomForest. In addition, we made use of an Unsupervised K-Means clustering approach. Hyperparameter tuning has been performed for all of our approaches.
For our SVM: we did a parameter space search for the parameters: C and gamma, where C is the penalty parameter of the error term. Gamma here is the spread of the radial basis kernel that we are making use of. For the RandomForest we are performing a GridSearchCV for the following hyperparameters: n_estimators which is the number of trees in the forest, max_features: maximum number of features for splitting a node, max_depth: the number of levels in each tree and criterion: which is the function to measure the quality of the split. In our opinion it was vital for us to explore the Random Forest option as this algorithm is intrinsically suited for multiclass problems while the SVM is more well suited for binary classification. In addition, the RandomForest works well with data that has a mixture of numerical and categorical features. Next, as for our unsupervised learning approach: The K-means clustering algorithm was employed. A collection of n datasets is divided into k sub groups or clusters using this iterative K-Means method depending on how similar they are to one another and how far off they are from the subgroup's centroid. K here signifies the number of clusters. For the K-means approach, the task at hand was to find the optimal number of clusters. The elbow method was chosen for this. The x-axis contained cluster numbers (spread in the range 2 through 10) and silhouette scores plotted on the y-axis.

2. Market Customer Segmentation Dataset

The dataset contains customer info such as gender, marital status, Education, etc. The data was sourced from Kaggle (linked in the references). There were Train and Test datasets with 8,067 and 2,626 samples respectively. Both sets had 9 common labels: 'ID', 'Gender', 'Ever_married', 'Age', 'Graduated', 'Profession', 'Work Experience', 'Spending_Score', 'Family' with the Test Set lacking the label: 'Segmentation'. The dataset needed to be dimensionally reduced for K-means clustering. PCA was used to choose the number of dimensions which was found to be 2. One important thing to note was the fact that the following features in the dataset had different frequency distributions.

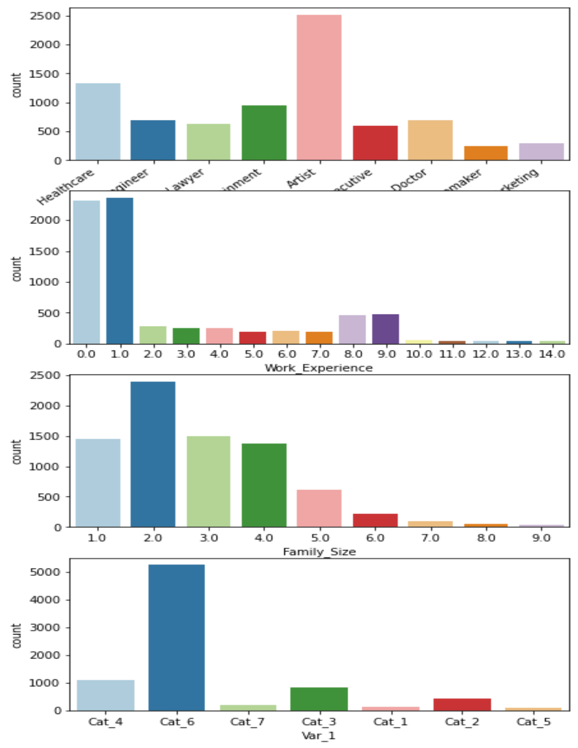

Figure 1: Infrequent feature distributions

To further explore the Train dataset: we performed EDA- Exploratory Data Analysis which can be further read in the .ipynb notebook attached at the link.

3. Data Preprocessing

Before performing any modeling, we have to make sure our data is cleaned and ready to use. We first make sure that there are no duplicate entries in the dataset. If present, they are dropped. The percentage of rows that have at least one null feature in the train and test sets are 17.38% and 18% respectively. Hence, we do not just drop these rows. Later we check for the percentage of rows that have 3 or more features with null values and they are found to be 0.23% for the train set and 0.22% for the test set. These are negligible amounts and so these rows have been dropped. The missing values have been filled using the KNN Imputer method which uses the k-Nearest Neighbors method to find the mean value from the parameter: n_neighbors. By default however, it uses the euclidean distance to compute the missing values. This method demands that all of the data be text data. Thus, before data imputation, we perform two types of encoding, namely: One-hot encoding for binary feature values and ordinal encoding for categorical feature values. Next, for the sake of modeling our data, we perform normalization as this limits our data to be between our selected feature range of 0 to 1 which is easier for our models to interpret. We have made use of the MinMax Scaler for this purpose.

4. Support Vector Machine Models

Support Vector Machines (abbreviated SVMs) are machine learning algorithms that are employed for regression and classification tasks. SVMs are a potent

machine learning method used for outlier detection, regression, and classification. A model is created by an SVM classifier that classifies data points into one of the predetermined categories. In SVMs, the main objective is to select a hyperplane with the maximum possible margin between support vectors in the given dataset. SVM searches for the maximum margin hyperplane by generating hyperplanes which segregate the classes in the best possible way. There are many hyperplanes that might classify the data. We should look for the best hyperplane that represents the largest separation, or margin, between the two classes.
Sometimes, the sample data points are so dispersed that it is not possible to separate them using a linear hyperplane. In such a situation, SVMs use a kernel trick to transform the input space to a higher dimensional space as shown in the diagram below. It uses a mapping function to transform the 2-D input space into the 3-D input space. Now, we can easily segregate the data points using linear separation.There are many kinds of kernels like Linear Kernel, Radial Basis Kernel, Polynomial Kernel, etc. Radial Basis Kernels work really well for both linearly separable and non-linearly separable data. Since we are not sure whether our data is linearly separable or not, Radial Basis Kernel has been used for our classifier. Usually SVM is not recommended for large datasets. Since our dataset has only about 8000 data points, SVM can prove to be a good classifier. The hyperparameter tuning is done using GridSearchCV for the Radial Basis Kernel SVM with two hyperparameters. The first hyperparameter is the regularization parameter ‘C’ that trades off correct classification of training examples against maximization of the decision function’s margin. For larger values of C, a smaller margin will be accepted if the decision function is better at classifying all training points correctly. A lower C will encourage a larger margin, therefore a simpler decision function, at the cost of training accuracy. The second one is ‘γ’ (gamma) where ‘γ’ is the spread of the radial basis kernel i.e it defines how far the influence of a single training example reaches, with low values meaning ‘far’ and high values meaning ‘close’.

5. Random Forest Classifier

One of the most adaptable and user-friendly algorithms is Random Forest. On the basis of the provided data samples, it constructs decision trees, obtains predictions from each tree, and votes for the top solution. It also serves as a fairly accurate measure of feature importance.

The Random Forest algorithm creates a forest of trees by combining various decision-trees, hence the name. The accuracy of the random forest classifier increases with the number of trees in the forest. However, Random Forests are computationally expensive and hence are not recommended for large datasets. Since the data we are using has only about 8000 data Random Forest can prove to be a good algorithm. Also in general Random Forest classifiers are used when the data has a mixture of categorical and numerical features. Our data has six categorical features and three numerical features.

Again we use GridSearchCV to find the best hyperparameters. The hyperparameters here are the number of trees in the forest (n_estimators), maximum number of features considered for splitting a node (max_features), max number of levels in each decision tree (max_depth) and loss function (criterion). Criterion helps to determine the best split of a tree.

6. K-Means Clustering Approach

K-Means, being an unsupervised learning technique, uses only the input features without referring to our labels / outcomes ('Segmentation' in our case). The objective is to group similar data points together and then be able to discover similar underlying patterns among the data in each cluster. We have to specify an initial k value for the number of centroids (center of each cluster) that we have. The algorithm then allocates each data point to the cluster whose center is the nearest. The choosing of our k value is done using the Elbow Method Technique. The method plots the explained variation as a function of the number of clusters(k) and then picks the elbow of the curve as the number of clusters to use. The silhouette coefficient has been used as a metric to choose the elbow point. The equation for the silhouette score is : **(b - a) /max(a,b)** where a = intra-cluster distance and b = mean nearest cluster distance for each sample. This score is a measure of how close or far each point in one cluster is to the points in the neighboring clusters. The

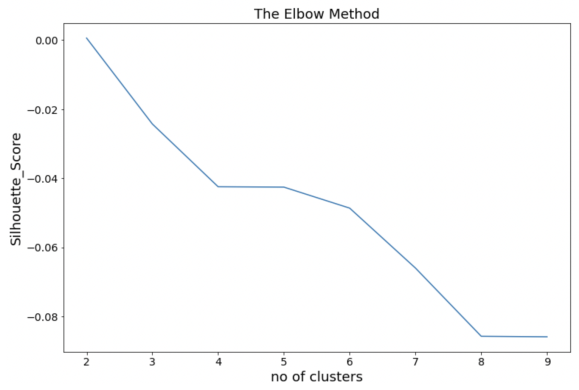

Figure 2: Elbow Graph

The elbow signifies that the optimal number of clusters is 4.
Dimensionality reduction by PCA was done to visualize the K-means clusters. Figure 3 shows how 2 components are sufficient for visualizing the result of our K-Means.

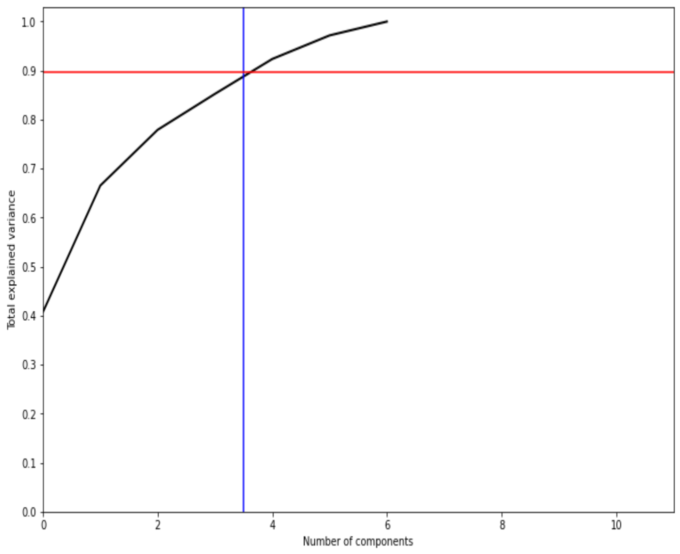

Figure 3: PCA analysis

A feature importance plot has been produced using Decision Trees as seen in Figure 4:

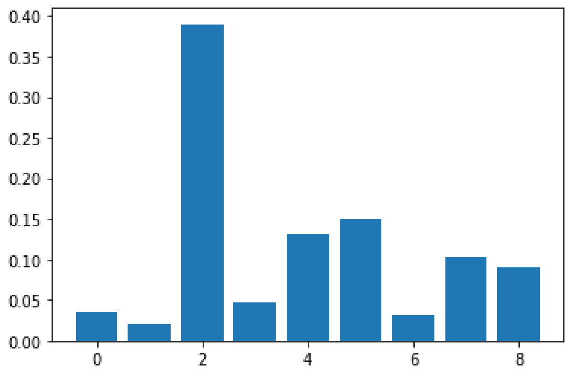

Figure 4: Feature Importances

The features 0- 'Gender' and 6- 'Spending Score' are seen to be the least important. However, based on our correlation matrix, there is an observed correlation between 'Ever_Married and 'Spending Score'. We choose to drop either one from these two and as a result 0-'Gender' and 1-

‘Ever_married’ have been dropped for clustering purposes and used as reporting variables. The result of our clustering on the train dataset is as follows:

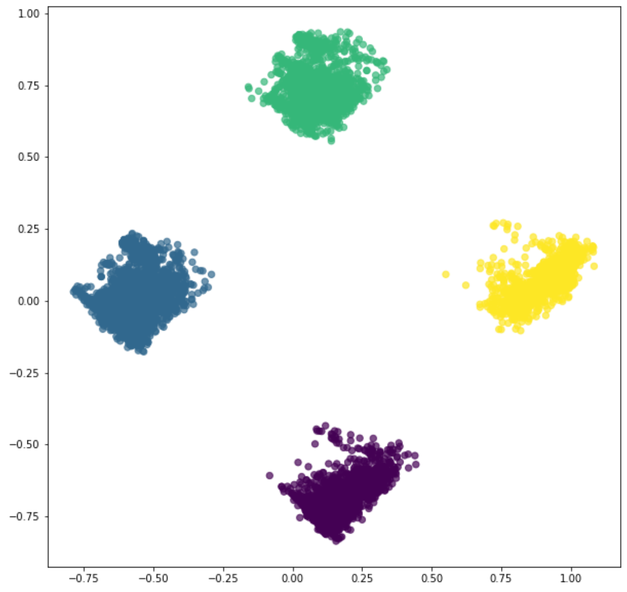

Figure 5: KMeans generated clusters (on train set)

7. Results

To summarize, we have built two supervised classification models and one clustering model, with data encoding, and hyperparameter tuning for all three models. Cross validation is also performed for the SVM. In addition, PCA dimensionality reduction is done for our K-Means model with elbow graph plotted with silhouette scores. Despite all our extensive approaches, we achieved quite poor results. Our error analysis leads us to conclude that the reasons for this outcome is as follows:

i) Lack of a higher number of features: In the real-world, classification and clustering especially are performed with a large feature space. We have also understood how a larger feature space improves accuracy scores as in our Homeworks. The concept of feature engineering could be performed to tackle this issue.

ii) The size of our dataset: Our train set is only around 8k samples. For multiclass classification and clustering, the more information available, the easier it would be to identify and learn underlying patterns in each segment.

iii) The correlation matrix suggests that the majority of features in our dataset, although influencing the target variable in a positive and negative sense, do not have a substantial influence on it. For better clusters or classes to be formed we need strong defining characteristics or features which are clearly lacking.

7.1 SVM

The best hyperparameters chosen via GridSearchCV are shown in the Table 1 below.

KERNEL	C	Gamma(γ)	Pr(Error)
RBF	1000	0.1	0.48

Table 1. SVM Best Hyperparameter Values

From the graph in Figure 6 we can see that the probability of errors for all hyperparameter values are on the higher side. The best performing case is when C is 1000 and γ is 0.1. However in this case as well the probability of error is quite high.

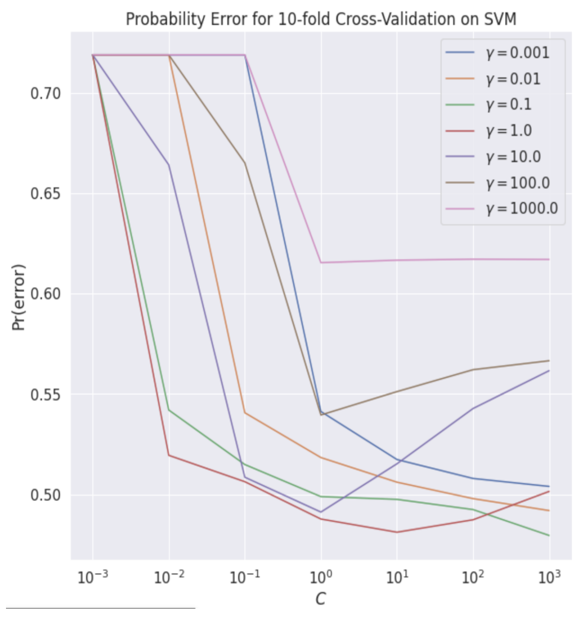

Figure 6: CV results for SVM

The validation metrics show that the precision and recall are almost the same. This means that our algorithm has classified an equal amount of data points as false positives, as it classified false negatives. Thus the model is balanced and Figure 7 and 8 show the confusion matrix and ROC curves respectively.

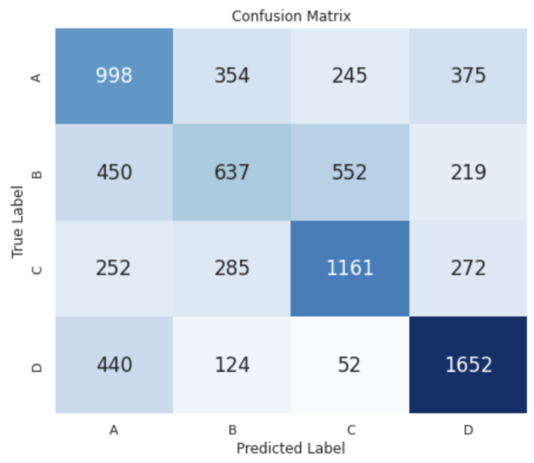

Figure 7:Confusion Matrix for SVM

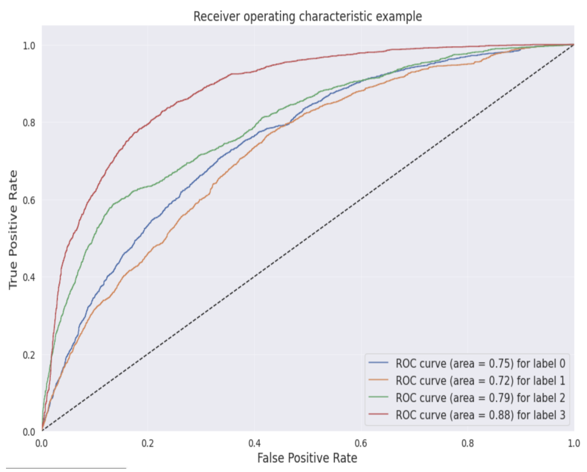

Figure 8: 10 Fold CV Results for SVM

The final performance metrics of SVM are as shown in Table 2.

PRECISION	RECALL	F1-SCORE	Pr(Error)
0.54	0.55	0.55	0.48

Table 2. SVM Final Performance Metrics

The F1 score in this case is quite low. This is due to the fact that there is no substantial relationship between the feature variables and the target variable. Another reason for the low score is due to the fact that our data has both numerical and categorical variables and SVM does not perform well in such a case. A model that does perform well under these circumstances is the Random Forest Classifier which has been implemented in the next section.

7.2 Random Forest Classifier

The best hyperparameters chosen via GridSearchCV are shown in Table 3.

n_estimators	max_features	max_depth	criterion
200	auto	8	Gini Impurity

Table 3. RF Best Hyperparameter Values

The best number of trees chosen for the model is 200 and the maximum number of features that are selected for the split is random. The maximum depth of the tree is 8 and Gini Impurity is used to determine how well a decision tree was split.

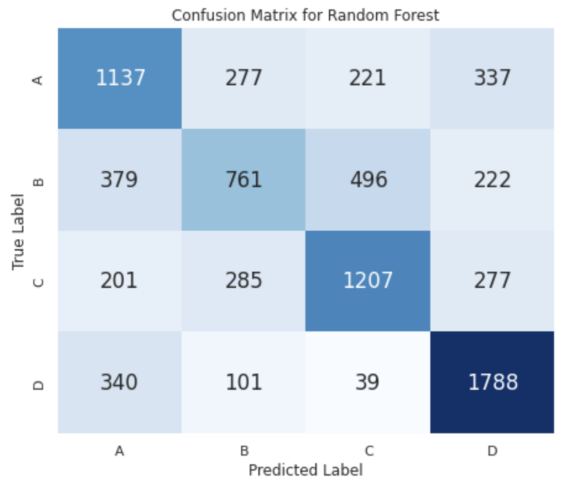

Figure 9: Confusion Matrix for Random Forest

Similar to the SVM we see that the precision and recall are almost similar and again the probability of error is quite high and F1 score is very low. This further proves that the training variables have a very weak relationship with the target variable.

PRECISION	RECALL	F1-SCORE	Pr(Error)
0.60	0.61	0.60	0.45

Table 2. Random Forest Final Performance Metrics

7.3 K-Means

The validation metrics for clustering tasks are usually as follows:

i) Compactness or cluster cohesion: Lower is the within cluster variation, better is the cohesion. The mean/median distance between data points in the cluster should be minimal. If this is not the case, we will see the majority of sample data points being quite dispersed in each cluster. In Figure 5 however, we see otherwise thus confirming a good cohesion.

ii) Separation: It measures how well-separated / discernible one cluster is from the other. We evaluate it by looking at the distance between cluster centers.

Figure 5. reveals how far off from each other these 4 centroids are and we can confidently say that these clusters are well formed and clearly defined.

Now, in addition we have made use of a supervised metric based on the fact that our train dataset has Segmentation target labels. The predicted class labels are generated for the train dataset and appended as a column stacked beside our label: ‘Segmentation’ . Encoding is performed for the Segmentation column that converts the labels A,B,C,D to 0,1,2,3 which is what our K Means algorithm outputs.

The accuracy score obtained was `18.8%`

This score being this low was expected considering the fact that we are primarily using a supervised metric of measurement against encoded target labels for comparison with an unsupervised technique. The result of applying K-Means on our test set is as follows:

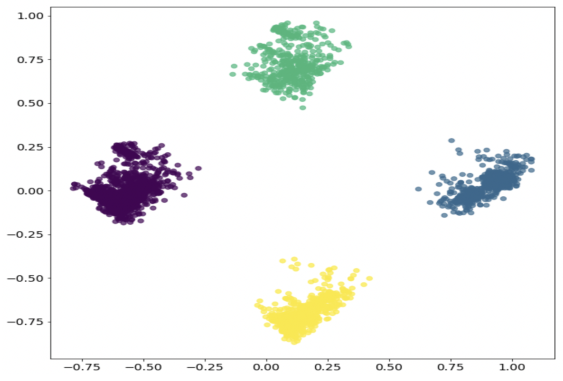

Figure 10: K-means on test set

8. Conclusion

In this paper, we present SVM and Random Forest classifier models to detect classes on a dataset of market customer segmentation.

The models presented a recall of 55% and 61% respectively. In our particular case, the Random Forest would be the favored approach as we are dealing with multi-class classification and a mixture of data types. The K-Means clustering yields well defined clusters, but with a low accuracy of 18.8%. Feature engineering, increasing the size of the data and features that have a discernible impact on the target variable should be used.

References

> Dataset hosted on Kaggle
https://www.kaggle.com/datasets/vetrirah/customer

> Random Forest and SVM combined model
https://iopscience.iop.org/article/10.1088/1757-899X/546/5/052066

>Xiahou, Xiancheng, and Yoshio Harada. "B2C E-Commerce Customer Churn Prediction Based on K-Means and SVM." *Journal of Theoretical and Applied Electronic Commerce Research* 17.2 (2022): 458-475.

> KNN Imputer
https://medium.com/@kyawsawhtoon/a-guide-to-knn-imputation-95e2dc496e

>K-Means and how it works
https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1

>Elbow method
https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/

>Customer Segmentation Using Machine Learning Techniques
https://www.sciencegate.app/document/10.1504/ijbidm.2022.10036753

>Customer Segmentaton Using Clustering and Data Mining
https://www.researchgate.net/publication/271302240_Customer_Segmentation_Using_Clustering_and_Data_Mining_Techniques

Appendix

Please find code for our project attached at:
https://github.com/Amey1398/EECE_5644_PROJECT

Thank you.