Zahra Jalilpour

Email: h19zahja@du.se

Report of AMI23B – Business Intelligence Lab3

Task1:

Introduction:

Clustering is one of the most common exploratory data analysis technique used to get an intuition about the structure of the data. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different. Unlike supervised learning, clustering is considered an unsupervised learning method since we don't have the ground truth to compare the output of the clustering algorithm to the true labels to evaluate its performance. We only want to try to investigate the structure of the data by grouping the data points into distinct subgroups.

Data Description:

After we have loaded dataset in Python data frame, the first step was to understand data, and to check if data is structured and to check for missing values. The results are presented in the table 1, while 5-number summary statistics for the quantitative variables are presented in the table 1.

Features Name	Missing Value	Relevant for model
Kommun_Name	No	For index
Kommun_code	No	No
Year	No	No
Revenue	No	Yes
Employee	No	Yes
Population	No	Yes
population_University	No	Yes
Percent_University	No	Yes
Productivity	No	Yes
SalesIndex	No	Relevant to Revenue
Infrast	No	Yes
Border	No	yes

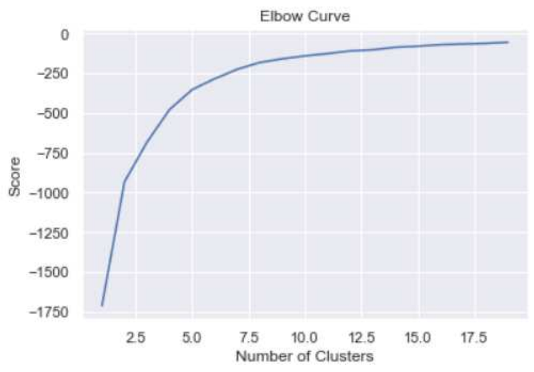
table1-Features of dataset

There are 207 observations and 12 variables (9 variables were taken as features for clustering and one for index). There are 10 observations (cities) with infrastructure, and 8 cities are close to border. From 18 cities that Ikea are located on, 12 cities are presented in this data set.
At first we should clean the data, By creating a new data frame and Kommun_name as index, and analyzing for missing values, we have 207 observations and 9 features. For scaling the data, we should standardize the data. For analyzing data set with multi features, it is better to use feature reduction. and one of the best method in feature reduction, is PCA. By using Scaled_data, and 4 components for PCA, we will have a new data frame.

	PC1	PC2	PC3	PC4
Kommun_name				
Haparanda	-0.442137	6.220053	-0.046862	-0.486328
Kalmar	1.748724	0.382760	-1.056933	0.397704
Karlstad	2.457948	0.387208	-1.303152	0.354918
Upplands Väsby	0.243091	-0.066118	-0.469815	0.106233
Vallentuna	-0.325750	-0.412781	-0.566916	-0.065218

K-means algorithm:

The "elbow" method helps data scientists to select the optimal number of clusters by fitting the model with a range of values for K. If the line chart resembles an arm, then the "elbow" (the point of inflection on the curve) is a good indication that the underlying model fits best at that point.



From the picture above we can see that optimal number of cluster is between 4 and 7. It is obvious from the picture that after 7 clusters, adding more clusters do not affect significantly the change in Score. Here we consider number of Cluster=7.

K-means Output

After we have decided on the number of clusters, the last step is to run the algorithm and to interpret results.

	PC1	PC2	PC3	PC4	clusters
Kommun_name					
Haparanda	-0.442137	6.220053	-0.046862	-0.486328	3
Kalmar	1.748724	0.382760	-1.056933	0.397704	0
Karlstad	2.457948	0.387208	-1.303152	0.354918	0
Upplands Väsby	0.243091	-0.066118	-0.469815	0.106233	2

Now we put a new column and add the cities where , ikea is located on. From 18 cities, only 12 cities are presented in the data set. In the next step we create 7 different data frames with cluster=0 to cluster=6, and make a different data_frames by the name of cities as index, Cluster, Kommun_code, Population, Revenue, SalesIndex, and analyze the 6 different clusters by comparing revenue, population, salesindex and the location of city.

Conclusion:

Finally, to make conclusion about which community should be good for IKEA to open new stores we created 7 different tables (each table for each cluster), and observed data within each cluster. We can conclude the following:

- Cluster 0 has 27 observations. In this cluster 6 communities have IKEA store, while 21 cities do not have. Huddinge is a community with the highest sales index and revenue, but there is no IKEA store. Therefore, we think that **Huddinge** would be a good place for IKEA store. Moreover, Norrköping would be a good community to consider for IKEA store in the same cluster because it has the highest population in this cluster that ikea is not located on it. However, most likely the reason why there is no store in this community, because it is close to Linköping, which has already IKEA store. Generally speaking, if we would have to choose one cluster to open IKEA stores, then it would be cities without stores in this cluster. In addition to Huddinge, we can consider **Falkenberg** as the second option with high revenue.
- Cluster 1 has only Stockholm community with IKEA store.
- Cluster 2 has 74 observations and one community has an IKEA store. **Trollhättan** community without IKEA store has high sales index, revenue and population. If we want to consider population, Botkyrka has the highest population in this cluster without ikea, but if in cluster 0, we consider Huddinge as a city to be located ikea on it, it is better we ignore Botkyrka, because this city is near to Huddinge. **Nyköping** could be a good option with higher revenue and population after **Trollhättan**.
- Cluster 3 has 8 observations and one community has an IKEA store that is Haparanda community. Haparanda is close to Finland boarder that is why it has IKEA store. In this cluster, Strömstad has highest revenue, but if in cluster2 we consider **Trollhättan**, is located near **Trollhättan,** hence we can consider **Arvika** as a community that should be located ikea on it.
- Cluster 4 has only Malmö community with IKEA store.
- Cluster 5 has 8 observations and one community has an IKEA store that is Helsingborg community. **Gotland** in this cluster has the highest revenue and population and there is any community which has ikea near Gotland .
- Cluster 6 has 88 observations and one community has an IKEA store that is Älmhult community. This cluster has communities with the lowest sales index, revenue and population compared to other clusters. but we can consider **Varberg** with the highest revenue in this cluster or **Haninge** as a community with the highest population in this cluster where ikea is not located on it.
- After analyzing the data for all the communities in each cluster, we can say that **Huddinge** in cluster 0 is the most suitable community for new IKEA store, followed by **Falkenberg**. In other clusters we can select **Trollhättan** , **Arvika, Gotland** and **Varberg.** We select these communities because these have high sales index, revenue and population. We have border feature in the data but we do not know how far is border from and which cities. If we have that data then that would be good to take this feature seriously in our decision about opening new IKEA in these Communities.

Task2:

Introduction:

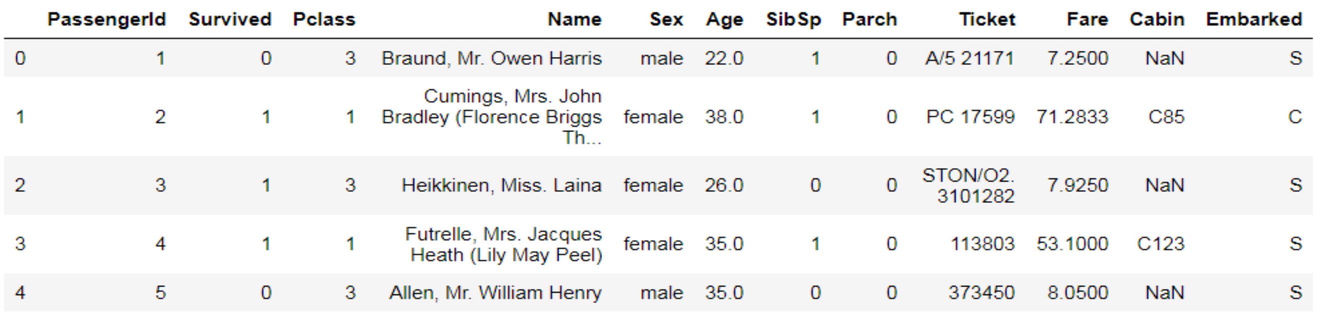
Decision Trees can help a lot when we need to understanding the data. Decision Trees are also very useful for exploring your data before applying other algorithms. They're helpful for checking the quality of engineered features and identifying the most relevant ones by visualizing the resulting tree. The main downsides of Decision Trees are their tendency to over-fit, their inability to grasp relationships between features, and the use of greedy learning algorithms (not guaranteed to find the global optimal model). Using them in a Random Forest helps mitigate some of this issues. After this short introduction to Decision Trees and their place in Machine Learning, let's see how to apply them for the Titanic challenge. First, we're going to prepare the dataset and discuss the most relevant features. We'll then find the best tree depth to avoid over-fitting, generate the final model, and explain how to visualize the resulting tree.

Data Description:

After we have loaded dataset in Python data frame, the first step was to understand data, and to check if data is structured and to check for missing values. The results are presented in the table 1.

Features Name	Missing Value	Relevant for model
PassengerId	No	No
Pclass	No	Yes
Name	No	No
Sex	No	Yes
Age	Yes	Yes
SibSp	No	No
Parch	No	Yes
Ticket	No	No
Fare	No	Yes
Cabin	Yes	No
Embarked	Yes	Yes

For the Titanic challenge we need to guess whether the individuals from the *test* dataset had survived or not. But for our current purpose let's also find out what can the data tell us about the shipwreck with the help of a Classification Tree. we can see that our dataset needs some treatment. The class Survived is already in binary format so no additional formatting is necessary, but features like Name, Ticket or Cabin need to be adapted for the problem we're trying to solve, and we can also engineer some new features by merging or regrouping existing ones.

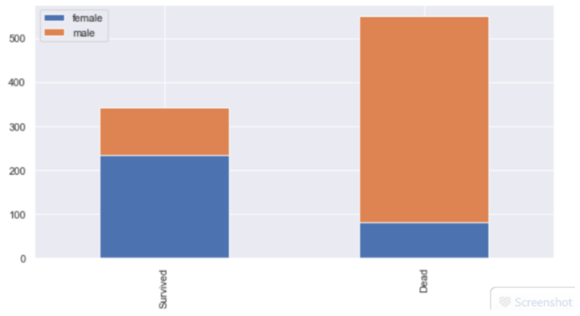


	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

Data Dictionary:

- Survived: 0 = No, 1 = Yes
- pclass: Ticket class 1 = 1st, 2 = 2nd, 3 = 3rd
- sibsp: # of siblings / spouses aboard the Titanic
- parch: # of parents / children aboard the Titanic
- ticket: Ticket number
- cabin: Cabin number
- embarked: Port of Embarkation C = Cherbourg, Q = Queenstown, S = Southampton

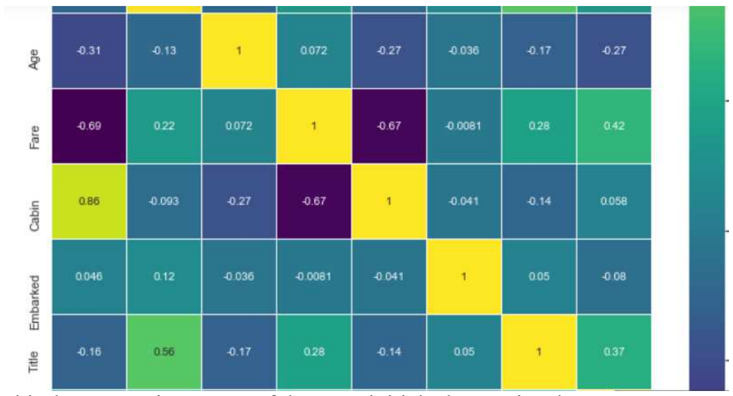
By visualization through bar chart we will see:



The Chart confirms Women more likely survived than Men, Other Charts confirm 1st class more likely survived than other classes and 3rd class more likely dead than other classes, a person boarded with more than 2 siblings or spouse more likely survived and without siblings or spouse more likely dead. The Chart confirms a person a boarded with more than 2 parents or children more likely survived and a person boarded alone more likely dead. A person boarded from C slightly more likely survived and person a boarded from Q more likely dead. The Chart confirms a person aboarded from S more likely dead. The next step is feature engineering. Feature engineering is the process of using domain knowledge of the data to create features (feature vectors) that make machine learning algorithms work. feature vector is an n-dimensional vector of numerical features that represent some object. Many algorithms in machine learning require a numerical representation of objects, since such representations facilitate processing and statistical analysis. We use feature engineering to split the title from the name,and removing unnecessary feature like Name from dataset. Converting male and female group to dummy variables. Smoe age is missing , hence title's median age for missing Age. Then Binning/Converting Numerical Age to Categorical Variable feature vector map: child:0 young:1 adult:2 mid-age:3 senior: 4. And for missing value in Embark, we use more than 50% of 1st class are from S embark more than 50% of 2nd class are from S embark more than 50% of 3rd class are from S embark, and fill missing Fare with median fare for each Pclass. Our clean data set is equal to:

	Pclass	Sex	Age	Fare	Cabin	Embarked	Title	FamilySize
0	3	0	1.0	0.0	2.0	0	0	0.4
1	1	1	3.0	2.0	0.8	1	2	0.4
2	3	1	1.0	0.0	2.0	0	1	0.0
3	1	1	2.0	2.0	0.8	0	2	0.4
4	3	0	2.0	0.0	2.0	0	0	0.0
5	3	0	2.0	0.0	2.0	2	0	0.0
6	1	0	3.0	2.0	1.6	0	0	0.0
7	3	0	0.0	1.0	2.0	0	3	1.6
8	3	1	2.0	0.0	2.0	0	2	0.8
9	2	1	0.0	2.0	1.8	1	2	0.4

Our dataset is now much cleaner than before, with only numerical values and potentially meaningful features. Let's now explore the relationship between our variables by plotting the Pearson Correlation between all the attributes in our dataset (credit to Anisotropic for this beautiful plot):



This heat map is very useful as an initial observation because you can easily get an idea of the predictive value of each feature. In this case, *Sex* and *Title* show the highest correlations (in absolute terms) with the class (*Survived*): 0.54 and 0.49 respectively. But the absolute correlation between both is also very high (0.86, the highest in our dataset), so they are probably carrying the same information and using the two as inputs for the same model wouldn't be a good idea. High chances are one of them will be used for the first node in our final decision tree, so let's first explore further these features and compare them.

Finding best tree depth with the help of Cross Validation:

After exploring the data, we're going to find of much of it can be relevant for our decision tree. This is a critical point for every Data Science project, since too much train data can easily result in bad model generalization (accuracy on test/real/unseen observations). Over-fitting (a model excessively adapted to the train data) is a common reason. In other cases, too much data can also hide meaningful relationships either because they evolve with time or because highly correlated features prevent the model from capturing properly the value of each single one. In the case of decision trees, the 'max_depth' parameter determines the maximum number of attributes the model is going to use for each prediction (up to the number of available features in the dataset). A good way to find the best value for this parameter is just iterating through all the possible depths and measure the accuracy with a robust method such as Cross Validation. Cross Validation is a model validation technique that splits the training dataset in a given number of "folds". Each split uses different data for training and testing purposes, allowing the model to be trained and tested with different data each time. This allows the algorithm to be trained and tested with all available data across all folds, avoiding any splitting bias and giving a good idea of the generalisation of the chosen model. The main downside is that Cross Validation requires the model to be trained for each fold, so the computational cost can be very high for complex models or huge datasets.

```
Max Depth  Average Accuracy
        1          0.782285
        2          0.799189
        3          0.818190
        4          0.826030
        5          0.841748
        6          0.827154
        7          0.821635
        8          0.824981
        9          0.817179
```

The best max_depth parameter seems therefore to be 5 (84.2% average accuracy across the 10 folds), and feeding the model with more data results in worst results probably due to over-fitting. We'll therefore use 5 as the max_depth parameter for our final model.