MEF UNIVERSITY

SCORING NEIGHBORHOODS FOR LOCATING ATM USING MACHINE LEARNING

Capstone Project

Oğuzhan Yıldırım

İSTANBUL, 2018

MEF UNIVERSITY

SCORING NEIGHBORHOODS FOR LOCATING ATM USING MACHINE LEARNING

Capstone Project

Oğuzhan Yıldırım

Advisor: Asst. Prof. Hande Küçükaydın

İSTANBUL, 2018

MEF UNIVERSITY

Name of the project: Scoring Neighborhoods For Locating ATM Using Machine Learning

Name/Last Name of the Student: Oğuzhan Yıldırım

Date of Thesis Defense: __/__/____

I hereby state that the graduation project prepared by Oğuzhan Yıldırım has been completed under my supervision. I accept this work as a "Graduation Project".

10/09/2018
Asst. Prof. Dr. Hande Küçükaydın

I hereby state that I have examined this graduation project by Oğuzhan Yıldırım which is accepted by his supervisor. This work is acceptable as a graduation project and the student is eligible to take the graduation project examination.

__/__/____

Director
of
Big Data Analytics Program

We hereby state that we have held the graduation examination of _____ and agree that the student has satisfied all requirements.

THE EXAMINATION COMMITTEE

Committee Member	Signature
1. Asst. Prof. Hande Küçükaydın	……..
2. Prof. Özgür Özlük	……..

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University's ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name	Date	Signature

EXECUTIVE SUMMARY

SCORING NEIGHBORHOODS FOR LOCATING ATM USING MACHINE LEARNING

Oğuzhan Yıldırım

Advisor: Asst. Prof. Hande Küçükaydın

SEPTEMBER, 2018, 28 pages

Facility location is a general problem that is important for many different sectors and it is even more important when building the facility costs too much. In this project we analyzed the neighborhoods of Turkey and built two different models to estimate the good and bad neighborhoods for locating an ATM, which has significant costs for banks to build one. We used demographic and socio-economic data of 4,504 neighborhoods in Turkey and built models using Linear Regression and Decision Tree techniques of Machine Learning to find the best neighborhoods for locating a new ATM for a new bank entering the market.

We compared the results of two machine learning methods and the results showed that we can make successful predictions of the neighborhoods by using machine learning methods which are good to locate an ATM without classical optimization techniques that requires complex calculations and machine learning methods.

Key Words: ATM Location, Facility Location, Neighborhood Scoring for ATM

ÖZET

SCORING NEIGHBOURHOODS FOR LOCATING ATM USING MACHINE LEARNING

Oğuzhan Yıldırım

Tez Danışmanı: Yrd. Doç. Dr. Hande Küçükaydın

EYLÜL, 2018, 28 sayfa

Tesis yer seçimi, birçok farklı sektörde var olan genel ve önemli bir sorundur. Eğer kurulmak istenen tesis maliyeti yüksek ve kurması zor / karmaşık bir tesis ise sorun daha da önem kazanmaktadır. Bu projede, Türkiye'nin mahallelerini analiz ettik ve bankalar için oldukça yüksek maliyeti olan "Nereye ATM konulmalı" sorusuna cevap olarak ATM yerleştirmek için iyi ve kötü mahalleleri tahminleyen iki farklı model geliştirdik. Türkiye'deki 4.504 mahallenin demografik, sosyoekonomik ve diğer bazı verilerini kullanarak, sektöre yeni giren bir bankanın hangi mahallelere ATM açması gerektiğini tahminleyen ve Makine Öğreniminin Doğrusal Regresyon ve Karar Ağacı tekniklerini kullanan modeller oluşturduk

.

İki makine öğrenim yönteminin sonuçlarını karşılaştırdık ve gördük ki geleneksel ve karmaşık olan optimizasyon yöntemi yerine makina öğrenim yöntemlerini kullanarak ATM kurmak için iyi olan mahalleler başarılı bir şekilde tahmin edilebilmektedir.

Anahtar Kelimeler: ATM konumlandırma, Tesis yer seçimi, ATM için Mahalle Skorlama

TABLE OF CONTENTS

1. INTRODUCTION

This project gives a summary of the Final Project of Big Data Analytics program of MEF University for 2018. More information can be seen on the source code as comments.

Facility location is a general problem that is important for many different sectors such as health (locating a hospital, pharmacy), energy (locating a gas / electrical charge station), service (locating a restaurant, hairdresser and post-office), shopping & entertainment (locating a Mall, cinema, amusement park). In facility location problems the location which has the highest possibility of getting a high transaction / service number is searched besides some other important aspects like ease of access, ease of building the facility, low cost, security etc.

In this project, we will offer a solution for the problem of locating an ATM for a new company / bank entering into finance sector. Locating an ATM is one of the most important problems for banks because the ATM itself is a high costing technological machine and the only channel that provides physical cash related services without any dedicated staff and maintaining it has various operational costs. In case of a fault, technical problem or in case of ATM running out of money, the bank must be able to send its staff as soon as possible to keep the ATM serving. Besides, as technology improves banking sector evolves from branches to non-branch channels. People can do many of their banking processes through internet or mobile banking, but they still need to use physical money for the rest of their needs and ATM's are the fastest / easiest way for doing that. Although the cost of building an ATM is high, it is always cheaper and easier to build and maintain an ATM instead of a branch. Branches require many employees to be physically at the branch and just serves in specific time periods while ATM's serve 7 / 24 with less staff and effort. In Turkey, there are many reputable banks trying to serve its customer and having correctly located ATM's makes a bank to go one step further among the others in such a highly competitive environment. In addition to these, correctly located ATM's contribute to the revenue as well by not only serving own-customers but also the customers of other banks with a certain amount of fee. Therefore, all of the factors above make locating an ATM to a place that will get a high number of transactions very important for a bank, especially for a new bank entering the sector, to increase its revenue and decrease the costs.

1.1. Literature Review of ATM & Facility Location Problem

Literature review on ATM Location problem shows that it is mostly considered as a classical facility location issue. While there are several approaches, the major approach to solve this problem is considering it as an optimization problem and proposing optimization based methods to find the optimum location. Quadrei and Habib [1] consider locating an ATM as a file server placement problem. They consider it as an optimization problem and they work on a mathematical model which is solved by a genetic algorithm to find out the right place for ATM's with minimum cost. On their work they offer seven different parameters to be considered to solve the problem where type of ATM is one of them, they consider that at different locations the type of ATM needed would be different and the rest of the parameters are: distances between locations, potential locations, maximum number of ATM's per location, cost of distance unit, maximum delay time and weights. Li et al. [2] has another optimization based solution (Particle Swarm Optimization (PSO) algorithm) for the problem but they also make use of the Geographical Information System technology for the solution. They have some basic assumptions such as: assuming that the cost of placing an ATM is the same everywhere and that there is one type of ATM, etc. Their assumptions basically produce a mathematical model that uses the historical data of the ATM's which were already located earlier. Another optimization based solution is offered by Aldajani and Alfares [3]. They consider a problem as to determine the optimum number of ATMs and their optimum locations. They formulated a mathematical model for the optimization problem which has the objective of minimizing the total number of ATMs to cover all customer demands within a given geographical area. Then a novel heuristic algorithm with unique features is developed to solve this problem. Lastly, they used their algorithm (a new heuristic solution that is based on the two-dimensional convolution) to simulate their method and the showed the results. After reviewing the literature, we see that most of the solutions are optimization based solutions but here on this project, we are going to assess the problem from a different point of view and try to solve it by using machine learning methods to find and assign a score for each neighborhood in Turkey that represents how good it is to locate an ATM in a specific neighborhood.

2. ABOUT THE DATA

The data set used for the project is collected by web-crawling after a detailed and multi-step work. The list of districts, counties, provinces and neighborhoods are downloaded from publicly available sites (Ministry of Interior & Postal and Telegraph Corporation). This list is used as the basis to collect the required information for each neighborhood and to create the data set for model development. In the final data set we have demographic, socio-economic and other information of 4,504 neighborhoods of 4 provinces in Turkey. Data collection process is done city by city starting from May 2018 until July 2018. Therefore, value in the data set reflects the situation as of May – June and July 2018 depending on which month that specific city data has been collected. Although we were able to collect most of the neighborhood data for big cities, we had difficulties to find the coordinates and the same set of information for some neighborhoods of rural areas. For this reason, neighborhoods which have null data are assumed to have the average of their related district / county.

The data set consists of total of 95 columns in its raw form which includes information such as the number of residences & private businesses, population distributions based on education level, gender, age, income level per house, population density, number of cars owned, number of pharmacies around, sales prices etc. Most of the features in the data set are either continuous or discrete numeric values except a couple of categorical features like ilce_endeks, il_endeks, tr_endeks include text information. Other text features are the name of district, county, province which does not have any information value except specifying the administrative governance unit. There are also features which are numeric but can be converted into categorical feature. For example we have the socio-economic population distribution as 4 different features the number of people in group A, B, C, and D. We can just label each neighborhood as A, B, C or D group instead of considering the population for each group in that neighborhood. Last but not least, there are features showing specific information per person which are expected to be discrete but having continuous value in the data set (i.e. number of pharmacy per person: 3.10347239)

2.1 Collecting Data

The first step of the project was to collect the required data on neighborhoods in Turkey. Data collection is mainly done by web crawling. Different web sites are examined and used to aggregate the data set. First thing we need to know is the list of neighborhoods in Turkey and the web-site of Ministry of Interior and PTT are the sources for that. At the web-site of Ministry of Interior the list of neighborhoods, districts, counties, provinces and their municipalities are available. PTT has also a list of neighborhoods in Turkey on its web-site publicly available that they use during their in-country shipments. Both of the lists are downloaded, examined and compared with each other. We have decided to use the list of PTT since it has a single list and more detailed, showing the connected villages and both the name of neighborhoods and villages at the same time on rural areas. Plus, PTT data also includes the post code designated to each neighborhood which might be good for the further analysis.

The second step was finding out the required data for our list of neighborhoods. After a comprehensive search on the Internet we came up with several sources that provide demographic, socio-economic, economic, educational and other information for neighborhoods in Turkey. However, the problem was that the sources required the coordinates of neighborhoods to provide the necessary information. At this point we used the geocoding APIs of Google and Yandex to convert our list of neighborhood address downloaded from PTT to the list of coordinates of neighborhoods. The reason we used two different (Google & Yandex) APIs is that Google has a limit of 2500 query per day and yandex does not. In addition, while one provides the false coordinates for some neighborhoods the other outputs the correct coordinates. Using two different APIs and reviewing, comparing their results gave the most accurate results. After this step, we managed to create the list of neighborhoods with their coordinates in Turkey and we were ready to use more web-sites and / or APIs to collect more information for modelling.

As the next step, we used several indexing web-sites to collect the detailed information about neighborhoods. By using the APIs which are opened on the Internet we queried 4,504 neighborhoods, 4 provinces of Turkey and managed to collect following information for each neighborhood:

- SOKAK_SAYI: 0,
- ILADI: "İSTANBUL",
- ILCEADI: "KARTAL",
- KOYADI: "MERKEZ",
- YERLEŞİM: null,
- MAHALLEADI: "SOĞANLIK YENİ",
- KONUT_TOPL: 10234,
- YAZLIK: 1,
- OZELISYERI: 941,
- ARSA_DEGER: 460.66666667,
- NÜFUS_TOP: 26212,
- NÜFUS_KAD: 13007,
- NÜFUS_ERK: 13205,
- EĞ_BİLİ: 159,
- EĞ_OKURYA: 513,
- EĞ_YOK_OK: 1867,
- EĞ_İLKOK: 5020,
- EĞ_İLKÖ: 2369,
- EĞ_ORTAOK: 2475,
- EĞ_LİSE: 5379,
- EĞ_LİSAN: 5103,
- EĞ_YÜKSE: 605,
- EĞ_DOKTOR: 88,
- yas_0_4: 2111,
- yas_5_9: 1666,
- yas_10_14: 1760,
- yas_15_19: 1844,
- yas_20_24: 2008,
- yas_25_29: 2765,
- yas_30_34: 2849,
- yas_35_39: 2536,
- yas_40_44: 2233,
- yas_45_49: 1720,
- yas_50_54: 1583,
- yas_55_59: 1048,
- yas_60_64: 770,
- ATM_SAYISI: 22,
- alan: 1.74754109,
- yoğunluk: 14999.3612121,
- M_BEKAR: 5368,
- M_EVLI: 12347,
- M_BOŞANMI: 727,
- M_EŞİ_Ö: 748,
- Agrubu: 3144,
- Bgrubu: 8387,

- Cgrubu: 7208,
- Dgrubu: 7473,
- hane_geliri: 1311,
- hane_geliri_t: 4168.98,
- eczane_kişibaşına: 3.10347239,
- otomobil_kişibaşı: 1304.3326223,
- araç_sayısı_kişibaşı: 1926.57883694,
- konut_satış_1el: 91.42130293,
- konut_satış_2el: 88.12113159,
- yap_kullan_izin_daire: 75.42312131,
- yapı_kullan_alan: 12537.94104277,
- yapı_kullan_izin_değer: 11763519.16284925,
- yapı_kullan_bina: 8.06465713,
- yapı_kullan_ort_daire: 9.35230352,
- yapı_kullan_metre_değer: 938.23372775,
- banka_şube_sayısı: 1.57359164,
- etc_adet: 523,
- etc_yogunluk: 0.01995269,
- üniversiteli_oran: 0.19468183,
- işyeri_nüfus_oran: 0.03589959,
- totalab: 11531,
- yaşlı_oran: 0.02937586,
- tr_indeks: "Orta-Üst",
- il_indeks: "Orta-Üst",
- ilce_index: 10,
- endeksa_deger: 3793.97,
- IsimIl: "İSTANBUL",
- IsimIlce: "KARTAL",
- IsimKoy: "MERKEZ",
- IsimMahalle: "SOĞANLIK YENİ",
- IsimYerlesim: null,
- NufusGenc: 9389,
- NufusOrta: 15504,
- EGITIM_TOP: 23578,
- TOPLAM_KONUT_ISYERI: 11175,
- KonutOran: 92,
- IsyeriOran: 8,
- NufusOranGenc: 36,
- NufusOranOrta: 59,
- NufusOranYasli: 26212,
- NufusOranEgitimBilimi: 1,
- NufusOranEgitimDoktora: 0,
- NufusOranEgitimLisans: 22,
- NufusOranEgitimLise: 23,
- NufusOranEgitimOkurYazar: 2,

- NufusOranEgitimOrtaOkul: 10,
- NufusOranEgitimYok: 8,
- NufusOranEgitimYüksekLisans: 3,
- NufusOranEgitimIlkOkul: 21,
- NufusOranEgitimIlkOgretim: 10

3. PROJECT DEFINITION

3.1. Problem Statement

In Section 1, it is mentioned why locating an ATM to the right place is important for a bank. Building an ATM at a dead or a low utilized location will just produce maintenance costs for the constitution and will decrease the customer number since customers will choose the banks that they can get service more conveniently. Replacing a false located ATM is also another extra cost for the bank both in financial and reputational aspect. The problem of locating ATMs to right places is basically finding the places that will have a high number of utilization which means that the bank may serve a high number of customers as possible. There are many aspects to be considered while choosing the place for an ATM such as the foot traffic density, drive traffic density, security, cost of place, distance, ease of access, etc. The problem is that all of these variables are dynamic which means they change depending on various other factors in time. In our project, we take this problem from the point of view of a new bank entering into the market in Turkey.

3.2. Project Objectives

In this project, we are going to focus on estimating the best neighborhoods for placing ATMs for a new bank entering into Turkish financial sector. Therefore, the objective of this project is to rate neighborhoods in Turkey according to how good they are to build a new ATM based on some assumptions. We assume that the cost, security and availability of all neighborhoods in Turkey are the same and positive. In other words, it is assumed that every neighborhood is secure and there is available spot to build an ATM at each neighborhood.

3.3. Project Scope

The scope of the project covers only the neighborhoods in Turkey and the predictions are done for a non-existing / new coming bank into the sector. Additional consideration of existing ATM locations must be done in order to use this solution for an existing bank in Turkey. In addition to this, the results do not represent a specific point in the neighborhood but it gives an idea whether it is a good neighborhood to locate an ATM. To specify the exact spot additional and more detailed examination needs to be done at

proposed neighbourhoods. Therefore, our solution provides potential good districts to build ATMs in a simple but comprehensive way and it can be used as a valuable input for other detailed researchs on the same issue.

4. METHODOLGY

As stated in previous sections we are going to use machine learning methods instead of an optimization based approach for modelling. We are going to build two different models with Linear Regression and Decision Tree algorithms and compare the results with each other. The data set will be divided into train test sections and another section will be separated for validation. Further sections give detailed information about the methodology followed for developing our models starting from how the data is collected.

4.1. Cleaning the Data Set and Exploratory Data Analysis

We collected a data set of 95 variables and 4,504 observations. When we analyze this data set in a detailed way we realized that it needs to be cleaned up to get ready for model building. Apart from basic cleaning we removed several fields intuitively by evaluating their predictive values and direct high correlations between each other. Moreover, after analyzing the features in a detailed way we realized that some features just causes too much detail and modelling would be easier if we merge them without losing significant information. The following contain actions we have taken to clean and tidy up the data set:

- The rows and columns having all null values are removed.
- Also columns having more than half of the observations null are removed.
- SOKAK_SAYISI had the value of “zero” for all rows. So we decided to remove SOKAK_SAYISI from our data set as well.
- Text fields such as name of the city, county, district are dropped since they do not have any predictive value.
- The field ATM_SAYISI is removed since it is going to be used as target field.
- The field BANKA_ŞUBE_SAYISI is removed since it is obviously highly correlated with the target field.
- We decided to use Female and Male population variables (NÜFUS_KAD and NÜFUS_ERK) and ‘Total Population’ variable (NÜFUS_TOP) is removed since it is obviously highly correlated with the first two.

- We had features showing the population distribution by 5 years of age intervals and we decided to merge them to come up with 4 different age intervals as: “Child population, Youth population, Middle population, Old population”. Therefore following merge operation is done on age intervals:
    - ‘0-4’, ‘5-9’ and ‘10-14’ are merged as child population.
    - ‘15-19’, ‘20-24’, ‘25-29’ and ‘30-34’ are merged as youth population.
    - ‘35-39’, ‘40-44’ and ‘45-49’ are merged as medium population.
    - ‘50-54’, ‘55-59’ and ‘60-64’ are merged as youth population.
- We merged the features that show the population of divorced and couple dead into one variable since they somewhat represents the people who lives single after marriage.

```
data['M_YALNIZ'] = data['M_BOŞANMI'] + data['M_EŞİ_Ö']
```

- We removed the features that show the number of people whose education level is unknown since those are unclear information.
- We decided to reduce education related fields into 3 fields as to be “Primary, Middle and High Education”. For this purpose features showing the population and population density of ‘primary school’, ‘primary education’ and ‘no education’ merged into a single variable as ‘Primary Education’. Similar to this, middle school and high school fields are merged into a single field of ‘Middle Education’. Lastly, undergraduate, masters and doctorate fields are merged to be ‘High Education’.
- We had two separate fields for automobile per person and vehicle per person. Vehicle per person is removed since they would be highly correlated.
- We had income per person and total income per house. We decided to remove income per person since they would be highly correlated.

4.2. Principal Component Analysis

After cleaning the data, we have decided to check out the correlations between remaining variables. For this purpose we created a correlation matrix and examined the matrix to identify highly correlated features. You can see our examination on correlation matrix as a separate excel file as well. Below, on **Figure 1,** you can see the correlation matrix graph of the remaining variables.

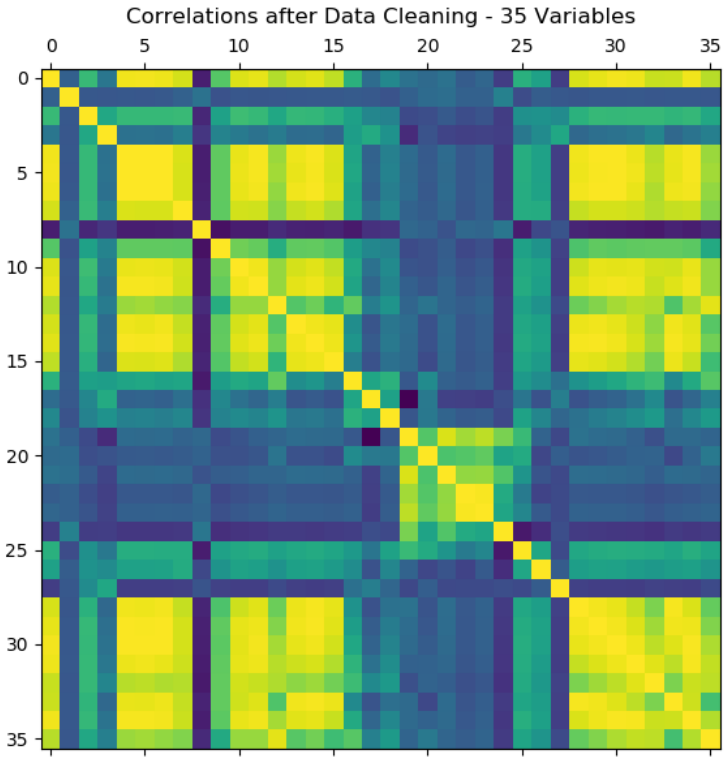

Figure 1 Correlation Matrix After Data Cleaning

On Figure 1 while yellow means that there is a high correlation between two variables as the color goes blue it means that the correlation level is decreasing. Strong blue means there is no correlation between related variables. From Figure 1, we can understand that after 'cleaning' operations we still have a data set of correlated variables. Therefore, we have to find a way to remove / decrease correlation on our data set. That is why, we decided to use *Principle Component Analysis* (PCA) technique to reduce the number of variables and come up with a few number of variables that has strong predictive power but weak correlations between each other. *Principle Component Analysis* is a machine learning and statistical method used for dimensionality reduction. It is a method that uses simple matrix operations from linear algebra and statistics to calculate a projection of original data into the same number or fewer dimensions by keeping the maximum variance on data. PCA is used mainly for two purposes first of which is to reduce the dimensionality of the data set and second, to find out new strong variables. Therefore, after using PCA we are going to not only decrease the number features but also will come up with new predictive feature set where new features called 'Principal Component'. While it is a commonly used technique its main drawback is that it is very

difficult to explain what principal components (the new features) represent since they are the combination of multiple features.

4.2.1 Normalization

We used PCA to project our data into fewer dimensions. However, since PCA is a technique that tries to maximize the variance we had to first normalize our data set between 0 – 1. We used two different methods for normalization. First, we have divided the population based features to total population which resulted in a ratio between 0 – 1. (i.e. or_nüfus_kad = NÜFUS_KAD / NÜFUS_TOP) Second, for the rest of the numeric features we used min – max scaling technique to get values between 0 – 1. Below is the simple formula for min – max scaling [11]:

$$(x - \min(x)) / (\max(x) - \min(x)) \qquad [11]$$

After normalization, we drew the correlation matrix graph again on Figure 2. As expected, Figure 2 shows that there is still correlation but the strength of correlation is less compared to 'prior to normalization' since the scale of features are decreased.

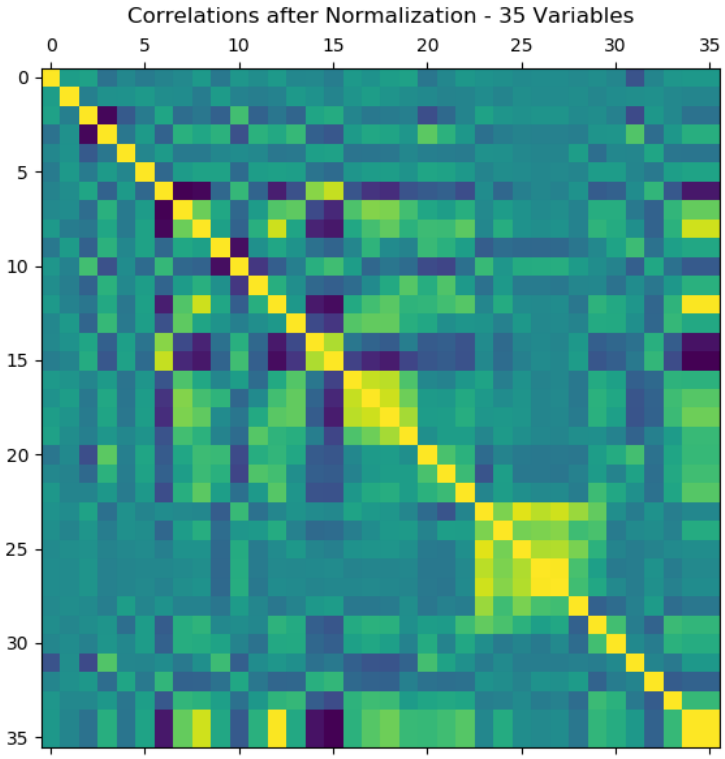

Figure 2 Correlations After Normalization

4.2.2 Detailed PCA

After normalization we are ready to apply PCA. As the first step, we needed to decide the number of features to be used for a successful projection of variances of remaining features. In Figure 3, y-axis represents the 'Cumulative Increase of Explained Variance' and x-axis represents the 'Number of Features'. As we expect, as the feature number increases, total explained variance also increases. When we use all futures projected variance value becomes 1, which means whole variance is covered up. On the other hand, we see that the lift amount of projected variance decreases as the feature number increases. Since our purpose is to cover up maximum variance with minimum number of features we need to decide the number of features. It can be seen that with 6 principal components 85 % of total variance is explained / projected. Therefore, 6 features are enough to cover explained variance significantly. But we decided to project 90 % of variance since it is possible to cover 90 % with only 8 features.

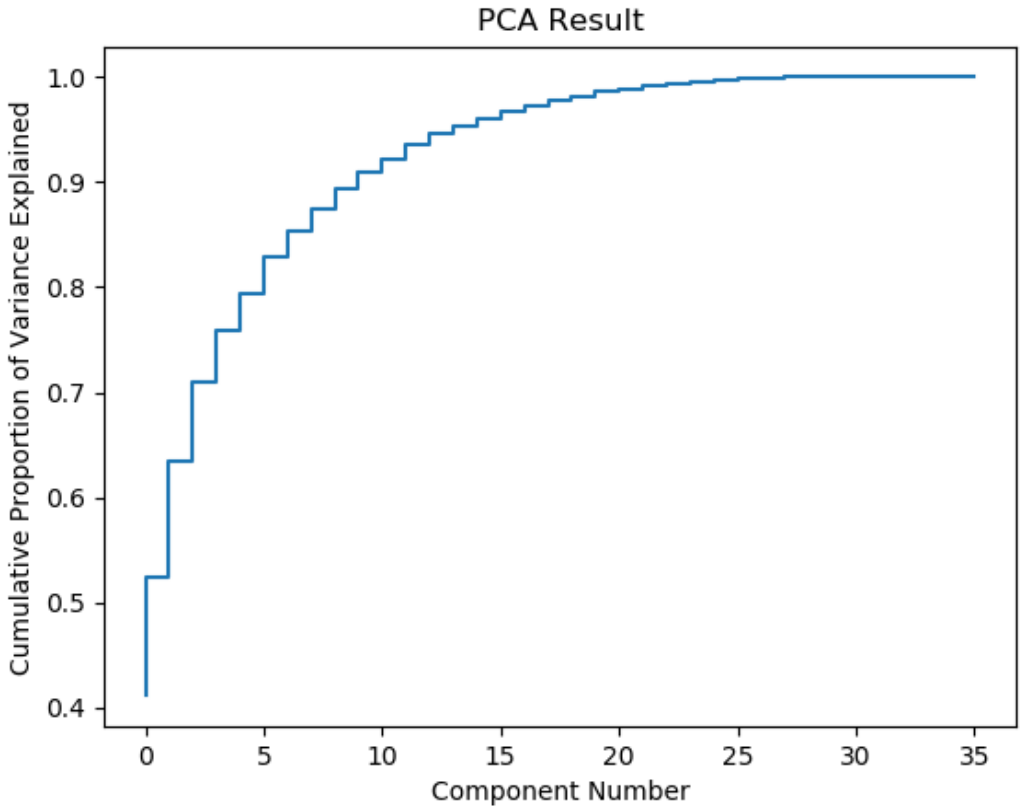

Figure 3 Cumulative Proportion of Variance Explained

Therefore, at the end of *Principal Component Analysis* we came up with 8 principal components (features) which cover the variance of 90 % of the last 35 variables. One disadvantage of PCA is that we cannot name these final 8 features because they are mixture (Eigen-vectors) of many variables. For this reason, we will call them as principal components. As a final step, we wanted to visualize the correlations between principal components to compare them with previous 35 variables' correlations. We expect to see no or very low correlations among principal components since during PCA correlations are expected to be eliminated significantly. Below, Figure 4 shows the correlation matrix plot of these final 8 principal components and we see that all are blue / purple since they have no / very low correlations between each other.

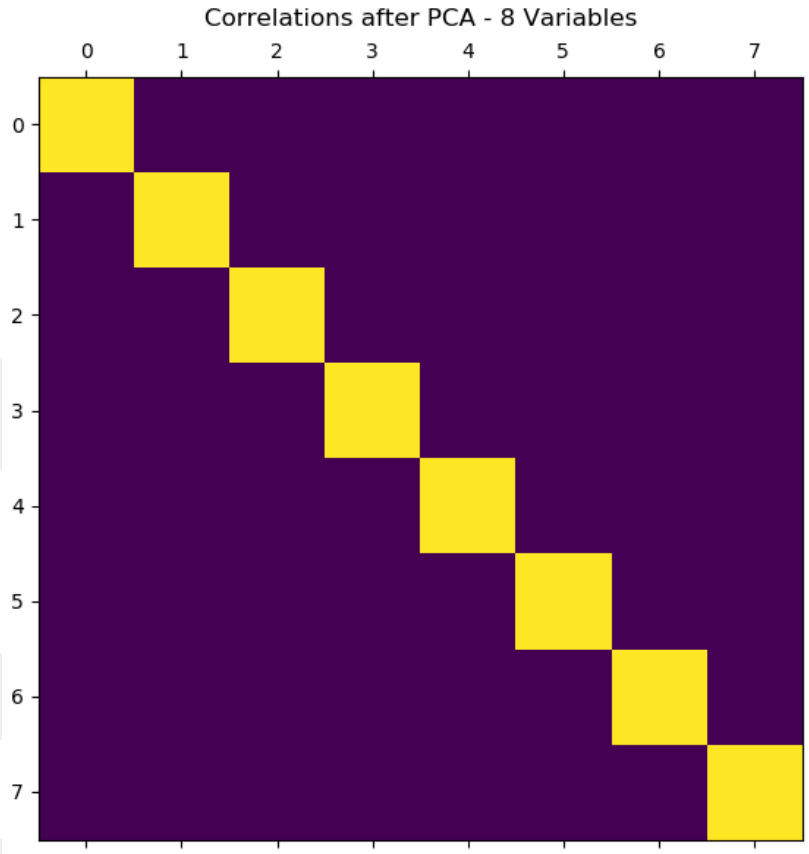

Figure 4 Correlation Matrix of Principal Components

4.3. Target Definition and Train & Test Split

We used ATM_SAYISI as target variable and decided to define neighborhoods which have more than certain number of ATMs as good neighborhoods and the rest as bad neighborhoods. Most private banks in Turkey does not build an ATM in areas where they do not see profitability but in such areas government banks build at least 1 ATM with non-profit purposes. We wanted to use this basic information to determine our target threshold and decided to declare it as 3 since if there are at least 3 ATMs in a neighborhood it means that ATMs are not dwelling there for just serving purpose but also since its profitable. Otherwise 1 or at most 2 ATMs would be enough to serve people living in the neighborhood. With this assumption, if we want to check out the number of goods and bads we came up with the following histogram plot:

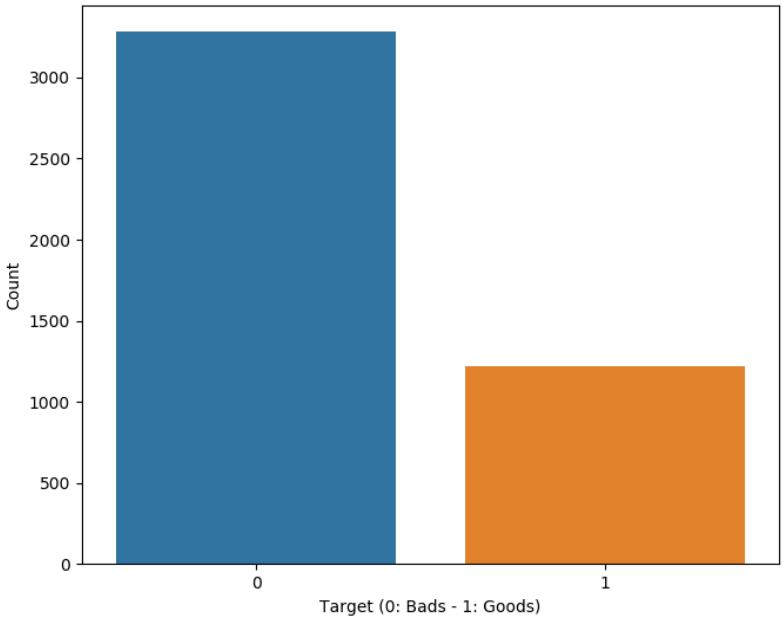

Figure – 5 Target Distribution

Aftter deciding the target we have divided the data into train and test data sets with a ratio of 0.8 train and 0.2 test set. While doing this we assigned random indices to each observation and picked train & test sets randomly among those observations that we have randomly indexed. This prevents possible biased train or test sets which would further cause overfitting models and biased predictions. The results explained in the RESULTS section are modelling results for this randomly prepared train and test data sets consisting of the mixture of neighborhoods belonging to İstanbul, Ankara, İzmir, and Adana.

5. RESULTS

We used three different techniques to create three different machine learning models on our train data set and tested the results on our test data set. The techniques used are Binary Logistic Regression, Decision Tree and Support Vector Machine. Further sections will compare and discuss the results and performances of these techniques.

5.1. Logistic Regression Train & Test Data Results

Since we have a binary target we decided to run first a Logistic Regression to predict whether a neighborhood is good for locating an ATM or not. Logistic Regression is a predictive analysis tool that machine learning borrowed from statistics to predict a binary target based on the probability scores. It explains the relationship between one dependent binary variable (target) and one or more independent variables (features) of different types. By using the relationships, Logistic regression tries to come up with a first degree equation and assigns the coefficients of the equation. Below is an example logistic regression equation:

$$y = \frac{e^{b_0 + b_1 x}}{1 + e^{b_0 + b_1 x}}$$

This equation is used to find a probability score for each observation and based on a cutoff value. Then the probability score is used to classify the observation as good or bad. On the equation 'y' represents the target variable, 'x' represents the input variable, 'e' is the Euler's number. Finally b0 is the constant and b1 is the coefficient of each input variable which all together make up the target variable. In our case, 'y' is the probability of having more than 2 ATMs and we have 8 different 'x' since we have 8 different features on our final data set that we will run Logistic Regression on. Our purpose is to find right coefficients (b0 and b1) for each feature that will create 'y' all together. After fitting our model on Python 3.6 using Logistic regression on train data set we made predictions on train and test data set and came up with following results:

- Train and Test set accuracy score: 0.87 – 0.88
- Confusion matrix of test set shows that we predicted 576 of 651 Bads correctly and 217 of 250 Goods correctly. Please check out Appendix A for description of 'Confusion matrix':

[[576, 75]

[33 217]]

- You can see the rest of the metrics below based on confusion matrix. Please check out Appendix A for description of metrics.

	precision	recall	f1-score	support
0	0.95	0.88	0.91	651
1	0.74	0.87	0.80	250
avg / total	0.89	0.88	0.88	901

A well-known valid metric to measure the goodness of binary classification models is ROC curve (AUC – Area Under Curve). The ROC curve is a fundamental tool for diagnostic test evaluation. In a Receiver Operating Characteristic (ROC) curve the true positive rate (Sensitivity) is plotted in function of the false positive rate (100-Specificity) for different cut-off points. Each point on the ROC curve represents a sensitivity/specificity pair corresponding to a particular decision threshold. A test with perfect discrimination (no overlap in the two distributions) has a ROC curve that passes through the upper left corner (100% sensitivity, 100% specificity). Therefore the closer the ROC curve is to the upper left corner, the higher the overall accuracy of the test (Zweig & Campbell, 1993). Figure 6 shows us below an example of a ROC curve that is statistically good enough.

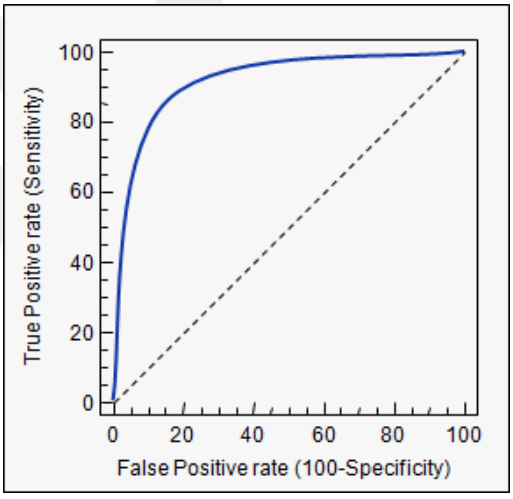

Figure 6 Example of a ROC curve

Below you can see the ROC curve of the logistic regression classifier for test data. The dotted line represents the ROC curve of a purely random classifier; a good classifier ROC plot stays as far away from that line as possible.

- Train - Test ROC scores: 0.83 – 0.88

Table 1 shows that the generally accepted goodness of models based on ROC curve score are as following:

ROC Curve Score	Goodness Category
.90 - 1	Excellent (A)
.80 - .90	Good (B)
.70 - .80	Fair (C)
.60 - .70	Poor (D)
.50 - .60	Fail (F)

Table 1 Classification of Models Based on ROC Curve Scores

Figure 7 below shows us that both train and test models have a good ROC curve. We can see that the curve is close to upper left corner which means we can use the Logistic Regression model to get significantly satisfactory number of True Positive predictions while having significantly satisfactory number of False Positive predictions.

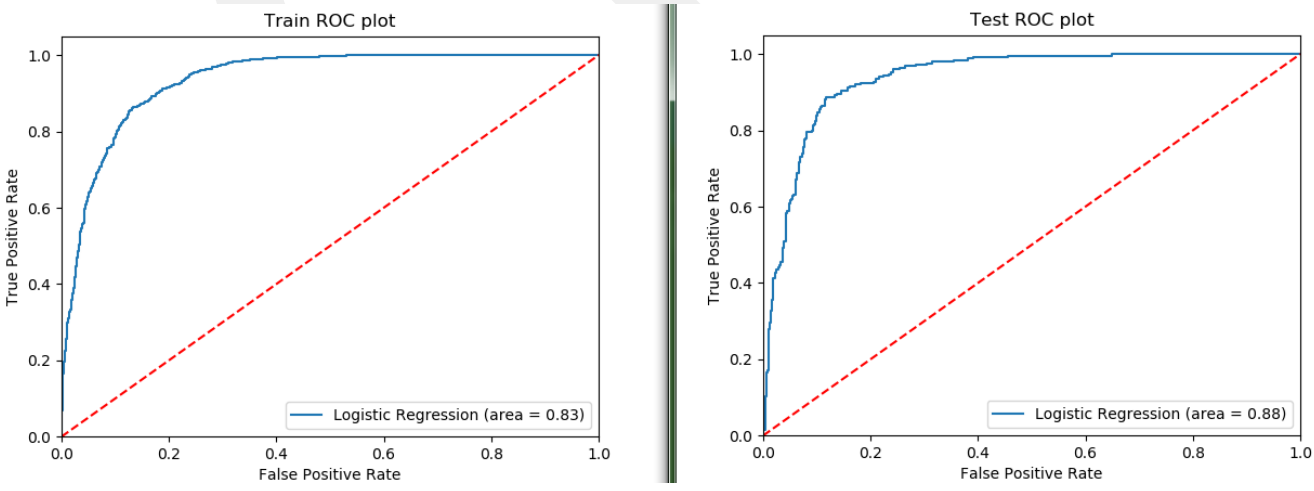

Figure 7 ROC & AUC for Logistic Regression Train and Test Predictions

5.2. Decision Tree Train & Test Data Results

Next we used Decision Tree algorithm to fit our model on the same train and test data sets. Results are as follows:

- Train - Test set accuracy scores: 0.86 – 0.86
- Confusion matrix of test set shows that we predicted 599 of 651 Bads correctly and 176 of 250 Goods correctly:

  [[599 52]

  [74 176]]
- You can see the rest of the metrics below based on confusion matrix:

	precision	recall	f1-score	support
0	0.89	0.92	0.90	651
1	0.77	0.70	0.74	250
avg / total	0.86	0.86	0.86	901

- Train and Test ROC scores: 0.81 – 0.81 (Below you can see the plots)

Figure 8 below shows us that both train and test models have a good ROC curve. 0.81 is slightly over the generally accepted value of 0.80. Again we see that we can use the Decision Tree model to get significantly enough number of True Positive predictions while having significantly enough number of False Positive predictions.

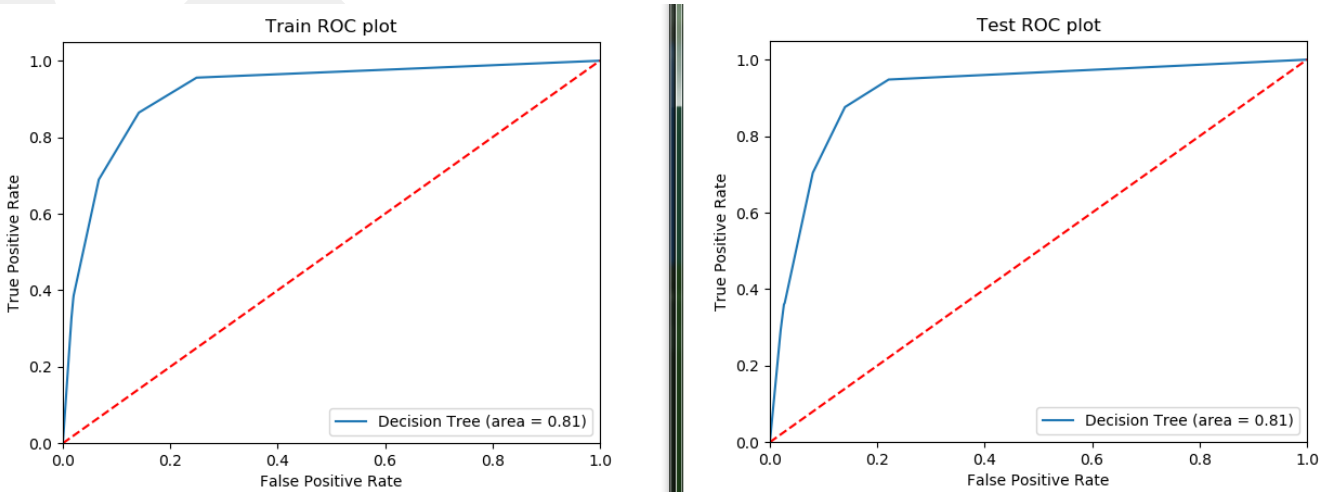

Figure 8 ROC & AUC for Decision Tree Train and Test Predictions

5.3. Support Vector Machine Train & Test Data Results

Finally we train our model by using Support Vector Machine algorithm with linear kernel on the same train and test sets. Here are the results:

- Train - Test set accuracy scores: 0.87 – 0.86

- Confusion matrix of test set shows that we predicted 573 of 651 Bads correctly and 221 of 250 Goods correctly:

  [[573 78]

  [29 221]]

- You can see the rest of the metrics below based on confusion matrix:

	precision	recall	f1-score	support
0	0.95	0.88	0.91	651
1	0.74	0.88	0.81	250
avg / total	0.89	0.88	0.88	901

- Train and Test ROC scores: 0.84 – 0.88 (Below you can see the plots)

Figure 9 below shows us that both train and test models have a good ROC curve on SVM model as well. Test ROC score of 0.88 is even very close to the generally accepted "Excellent" value of 0.90. Again this shows that we can use the model to get significantly enough number of True Positive predictions while having significantly enough number of False Positive predictions.

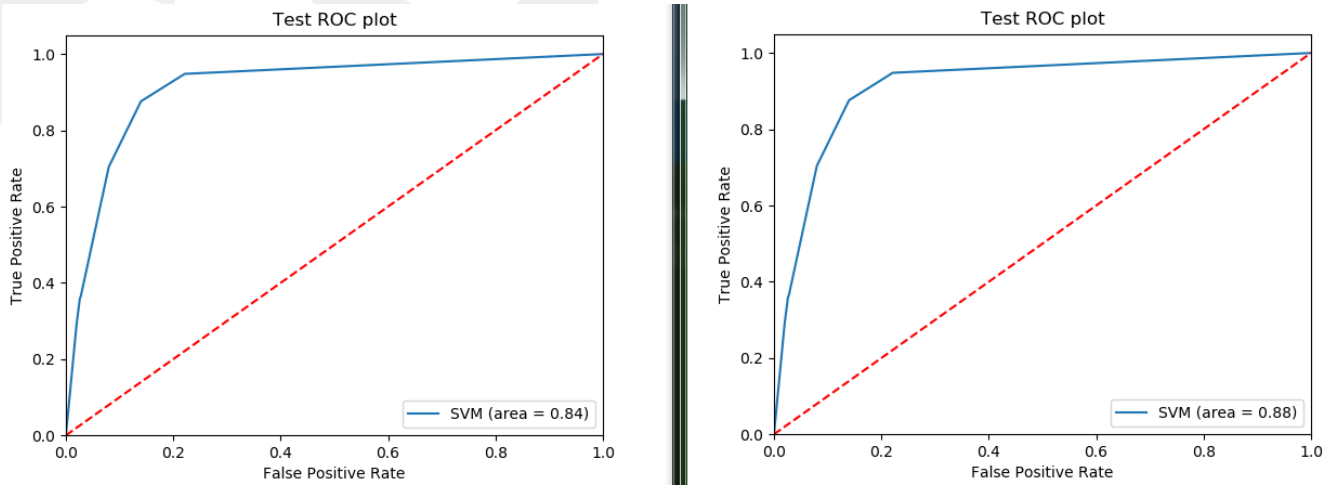

Figure 9 ROC & AUC for SVM Train and Test Predictions

Let us start with comparing the accuracy score of our models. Table 2 shows the train and test accuracy scores and ROC curve scores for each model. Accuracy score is simply the number of correct predictions over all predictions. We can see that all of our models train and test accuracies are close to each other and each model have a good performance. This shows that none of our models over fit and although they are very close to each other, the best score belongs to Logistic Regression. Another point of view is that when we check out the y.means() of test set it is 0.29 which tells that even if we labeled all neighborhoods as "Bad" we would have an accuracy of 71 %. However, our models have the accuracy of 86 % to 88 % which shows that they are doing better.

Although it gives an idea about how good are our predictions, depending solely on accuracy score is not a good practice of model performance evaluation since it does not show anything about correctly predicted Goods / Bads.

	Logistic Regression	**Decision Tree**	**Support Vector Machine**
Accuracy Score	0.87 – 0.88	0.86 – 0.86	0.87 – 0.86
ROC	0.83 – 0.88	0.81 – 0.81	0.84 – 0.88

Table 2 Accuracy and ROC Scores of Models

Table 2 second row depicts the ROC curve scores of our train and test data sets. We can see that all our models have a ROC which is in "B - Good" category but Logistic Regression and SVM models have significantly better (0.88) ROC compared to Decision Tree (0.81).

One last but not least evaluation metric we examined is the Recall of Goods and Bads, which shows the ratio of correctly predicted Goods over all actual Goods and the

same for Bads. Table 3 gives a summary of Recalls for each model. The first thing we see that Decision Tree model is very good at predicting Bads while it is bad at predicting Goods. (0.92 – 0.70) For Logistic Regression and SVM, Recalls for Goods and Bads are (almost) the same. (0.88) At this point our conclusion is that we can use an ensemble method of Decision Tree and Logistic Regression / SVM. If we consider the neighborhoods that Decision Tree tells us Bad as Bad and for the rest of the neighborhoods we depend on the results of Logistic Regression / SVM would give us the best results.

	Logistic Regression	**Decision Tree**	**Support Vector Machine**
Recall Goods - Bads	0.87 – 0.88	0.70 – 0.92	0.88 – 0.88

Table 3 Recall Comparison of Models

Table 4 gives some examples of predictions of model developed by SVM algorithm.

City	**County**	**Neighborhood**	**Prediction**	**Actual**
İZMİR	SEFERİHİSAR	ORHANLI	Bad	Bad
İZMİR	URLA	HACI İSA	Good	Good
İSTANBUL	ÇEKMEKÖY	SOĞUKPINAR	Bad	Good
ANKARA	KIZILCAHAMAM	PAZAR	Bad	Bad
ADANA	SEYHAN	ÇINARLI	Good	Good

Table 4 SVM Prediction Examples

5.4. Conclusion

In this project first we have collected socio-economic, education and demographic information about neighborhoods of several cities in Turkey by crawling the web and latitude longitude information. Then we cleaned the data and applied feature selection, dimensionality reduction (PCA) techniques by considering the correlations between variables and insight in order to build machine learning algorithms to find the Good and Bad neighborhoods to locate an ATM.

We tried Logistic Regression, Decision Tree and Support Vector Machine algorithms on final data set and observed that in overall all three algorithms resulted in significantly good enough models to predict if a neighborhood is Good or Bad to locate an ATM. However while Decision Tree is the best in predicting the Bad neighborhoods, Logistic Regression and SVM is better in predicting the Good neighborhoods. Therefore, it is concluded that we can use machine-learning techniques to predict the neighborhoods to locate an ATM. But the best performance is received ensembling different models developed by different techniques.

This work depends on solely neighborhood data and does not give a point estimate for locating an ATM. To improve this, further street information can be collected at the neighborhoods estimated as Good by this model. In other words, the outputs of this model can be used as input by a separate street data based machine learning model or any other optimization based model in order to eliminate Bad neighborhoods and focus on Good neighborhoods.

REFERENCES

[1] A. Quadrei, S. Habib, "Allocation of heterogeneous banks' Automated Teller Machines", In: 2009 First International Conference on Intensive Applications and Services, April 21-25, 2009, pp.16-21, Valencia

[2] Y. Li, H. Sun, C. Zhang, G. Li, "Sites selection of ATMs based on particle swarm optimization", In: 2009 International Conference on Information Technology and Computer Science, July 25-26, 2009, pp.526-530, Kiev.

[3] M.A. Aldajani, H.K. Alfares, "Location of banking Automatic Teller Machines based on convolution", Computers & Industrial Engineering, vol. 57, pp.1194-1201, 2009.

[4] https://machinelearningmastery.com/

[5] https://stats.stackexchange.com/

[6] http://www.statisticssolutions.com/

[7] http://blog.exsilio.com/

[8] https://towardsdatascience.com/

[9] https://www.medcalc.org/manual/roc-curves.php

[10] ftp://statgen.ncsu.edu/pub/thorne/molevoclass/AtchleyOct19.pdf

[11] http://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html

[12] https://www.mathsisfun.com/

[13] http://setosa.io/ev/principal-component-analysis/

APPENDIX A

Confusion Matrix: The matrix that shows the actual goods and bads vs predicted goods and bads. In other words, a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known.

	Predicted class		
Actual Class		Class = Yes	Class = No
	Class = Yes	True Positive	False Negative
	Class = No	False Positive	True Negative

True Positives (TP) - Correctly predicted goods (positive observations) which means that the value of actual class is yes and the value of predicted class is also yes.

True Negatives (TN) - Correctly predicted negative values which means that the value of actual class is no and value of predicted class is also no.

False Positives (FP) – When actual class is no and predicted class is yes.

False Negatives (FN) – When actual class is yes but predicted class in no.

Accuracy score: The ratio of correctly predicted observations to the total observations. Accuracy = (TP+TN) / (TP+FP+FN+TN)

Precision: The ratio of correctly predicted goods (positive observations) to the total predicted goods (positive observations). Precision = TP / (TP+FP)

Recall (Sensitivity): The ratio of correctly predicted goods (positive observations) to the all observations. Recall = TP / (TP+FN)

F1 Score: The weighted average of Precision and Recall. Therefore, this score takes both false positives and false negatives into account.

F1 Score = 2*(Recall * Precision) / (Recall + Precision)

APPENDIX B

```
import csv
import numpy as np
import pandas as pd

#Get and arrange header names for data set
header = pd.read_csv('C:/Users/TOSHIBA/Documents/MEF/Project/header.txt',
header=None, encoding = "utf-8") #Read headers
names = []
for i in range(0, header.__len__()):
    row = __builtins__.str(header.iloc[i:i+1,:1])
    row = row.split(":")[0]
    names.append(row.split("\n")[1].split(" ")[2])
    i += 1

#Get the data set
data = pd.read_csv('C:/Users/TOSHIBA/Documents/MEF/Project/data.csv',
header=None, encoding = "ISO-8859-9", names=names)
#validation_data = pd.read_csv('data_adana_hatay.txt', header=None,
encoding = "ISO-8859-9", names=names)
#validation_data = pd.read_csv('C:/Users/TOSHIBA/Documents/MEF/BDA-502
Intr to Machine Learning/Project/data_adana.txt', header=None, encoding =
"ISO-8859-9", names=names)

# Get
data['rand_index'] = pd.DataFrame(np.random.choice(4504, replace=False,
size=4504))
data.sort_values('rand_index', inplace=True)
data.describe()

data.shape

data = data.dropna(axis=0, how="all") # remove all NA rows
data = data.dropna(axis=1, how="all") # drop all NA rows
#validation_data = validation_data.dropna(axis=0, how="all") # remove all
NA rows
#validation_data = validation_data.dropna(axis=1, how="all") # remove all
NA cols
# validation_data = validation_data.dropna(axis=1, how="all", thresh=1) #
??? remove any column that have more than half of the observations NA
# validation_data = validation_data.dropna(axis=0, how="all", thresh=1) #
??? remove any column that have more than half of the observations NA
data = data.drop('SOKAK_SAYI', axis=1) # drop numeric columns having all
values "0"

#identify numeric and categoric features
num_cols = []
cat_cols = []
i = 0
for i in range(0, names.__len__()):
    try:
        row = __builtins__.str(data[names[i]].dtype)
        if(row == "float64"):
            num_cols.append(names[i])
        elif(row == "object"):
            cat_cols.append(names[i])
    except:
        continue
```

```
# Tranfer two possible target fields
target = pd.DataFrame()
target['ATM_SAYISI'] = data['ATM_SAYISI']
data = data.drop('ATM_SAYISI', axis=1)

# Remove branch number since it's highly correlated with target value
data = data.drop(                    , axis=1)

# Remove name fields since they don't have a predictive value and keep
the rest
for row in cat_cols:
    if(row != 'tr_indeks' and row != 'il_indeks'):
        data = data.drop(row, axis=1)

# Combine age intervals
data['yas_cocuk'] = data['yas_0_4'] + data['yas_5_9'] + data['yas_10_14']
data['yas_genc'] = data['yas_15_19'] + data['yas_20_24'] +
data['yas_25_29'] + data['yas_30_34']
data['yas_orta'] = data['yas_35_39'] + data['yas_40_44'] +
data['yas_45_49']
data['yas_yasli'] = data['yas_50_54'] + data['yas_55_59'] +
data['yas_60_64']
data = data.drop('yas_0_4', axis=1)
data = data.drop('yas_5_9', axis=1)
data = data.drop('yas_10_14', axis=1)
data = data.drop('yas_15_19', axis=1)
data = data.drop('yas_20_24', axis=1)
data = data.drop('yas_25_29', axis=1)
data = data.drop('yas_30_34', axis=1)
data = data.drop('yas_35_39', axis=1)
data = data.drop('yas_40_44', axis=1)
data = data.drop('yas_45_49', axis=1)
data = data.drop('yas_50_54', axis=1)
data = data.drop('yas_55_59', axis=1)
data = data.drop('yas_60_64', axis=1)

# Combine divorced and couple dead on a single field
data['M_YALNIZ'] = data['M_BO      ] + data[         ]
data = data.drop([           ,           ], axis=1)

# Combine first education related fields on 1 field
data[        ] = data[           ] + data[           ] + data[          ] +
data[          ]
data['NufusOranEgitimIlk'] = data['NufusOranEgitimOkurYazar'] +
data['NufusOranEgitimIlkOkul'] + data['NufusOranEgitimIlkOgretim'] +
data['NufusOranEgitimYok']
data = data.drop([           ,           ,           ,           ],
axis=1)
data = data.drop(['NufusOranEgitimOkurYazar', 'NufusOranEgitimIlkOkul',
'NufusOranEgitimIlkOgretim', 'NufusOranEgitimYok'], axis=1)

# Combine middle education related fields on 1 field
data[         ] = data[           ] + data[         ]
data[                      ] = data['NufusOranEgitimLise'] +
data['NufusOranEgitimOrtaOkul']
data = data.drop([           ,           ], axis=1)
data = data.drop(['NufusOranEgitimOrtaOkul', 'NufusOranEgitimLise'],
axis=1)
```

```
# Combine high education related fields on 1 field
data[        ] = data[          ] + data[          ] + data[           ]
data[                    ] = data['NufusOranEgitimLisans'] +
data[                           ] + data['NufusOranEgitimDoktora']
data = data.drop([          ,           ,           ], axis=1)
data = data.drop(['NufusOranEgitimLisans',                                ,
'NufusOranEgitimDoktora'], axis=1)

# Remove indexes since they are not on district based
data = data.drop(['tr_indeks', 'il_indeks', 'ilce_index',
'endeksa_deger'], axis=1)
#validation_data = validation_data.drop(['tr_indeks', 'il_indeks',
'ilce_index', 'endeksa_deger'], axis=1)


data = data.drop('etc_adet', axis=1)

################ PCA ####################
from sklearn.decomposition import PCA

# Before PCA we need to normalise the values
# First create a new data frame X to store the normalised values
norm = pd.DataFrame()
norm[             ] = data[          ] / data[          ]
norm[            ] = data[          ] / data[          ]

norm['or_konut'] = data['KONUT_TOPL'] / (data['KONUT_TOPL'] +
data['YAZLIK'] + data['OZELISYERI'])
norm[            ] = data['OZELISYERI'] / (data['KONUT_TOPL'] +
data['YAZLIK'] + data['OZELISYERI'])
norm[           ] = data['YAZLIK'] / (data['KONUT_TOPL'] + data['YAZLIK']
+ data['OZELISYERI'])

norm[          ] = data[        ] / (data[         ] + data['E      ] +
data[         ] + data[        ])
norm[          ] = data[        ] / (data[         ] + data[        ] +
data[         ] + data[        ])
norm[          ] = data[        ] / (data[         ] + data[        ] +
data[         ] + data[        ])
norm[          ] = data[        ] / (data[         ] + data[        ] +
data[         ] + data[        ])

norm['or_bekar'] = data['M_BEKAR'] / (data['M_BEKAR'] + data['M_EVLI'] +
data['M_YALNIZ'])
norm['or_evli'] = data['M_EVLI'] / (data['M_BEKAR'] + data['M_EVLI'] +
data['M_YALNIZ'])
norm[          ] = data['M_YALNIZ'] / (data['M_BEKAR'] + data['M_EVLI']
+ data['M_YALNIZ'])

norm['or_agrubu'] = data['Agrubu'] / (data['Agrubu'] + data['Bgrubu'] +
data['Cgrubu'] + data['Dgrubu'])
norm['or_bgrubu'] = data['Bgrubu'] / (data['Agrubu'] + data['Bgrubu'] +
data['Cgrubu'] + data['Dgrubu'])
norm['or_cgrubu'] = data['Cgrubu'] / (data['Agrubu'] + data['Bgrubu'] +
data['Cgrubu'] + data['Dgrubu'])
norm['or_dgrubu'] = data['Dgrubu'] / (data['Agrubu'] + data['Bgrubu'] +
data['Cgrubu'] + data['Dgrubu'])
```

```
norm[          ] = data['yas_cocuk'] / data[           ]
norm[         ] = data['yas_genc'] / data[           ]
norm['or_orta'] = data['yas_orta'] / data[           ]
norm[          ] = data['yas_yasli'] / data[           ]

norm[            ] = data['ARSA_DEGER']
norm[                  ] = data[                  ]
norm[                  ] = data[                  ]
norm[                ] = data[                ]
norm[                ] = data[                ]
norm['yap_kullan_izin_daire'] = data['yap_kullan_izin_daire']
norm[                  ] = data[                  ]
norm[                      ] = data[                      ]
norm[                  ] = data['                 ]
norm[                      ] = data[                      ]
norm[                       ] = data[                       ]
norm['etc_yogunluk'] = data['etc_yogunluk']
norm['alan'] = data['alan']
norm[          ] = data[          ]
norm['hane_geliri'] = data['hane_geliri']
norm[                ] = data['hane_geliri_t']

#validation_data = norm[0:1000]
#X = norm[1000:]
X = norm

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
# Create train test data
x_train, x_test = train_test_split(X, test_size=0.2, random_state=0)
# Replace NAN with average
x_train = x_train.fillna(x_train.mean())
x_test = x_test.fillna(x_test.mean())
#validation_data = validation_data.fillna(validation_data.mean())

# Scale train test data for PCA
scaler = MinMaxScaler()
x_train = scaler.fit_transform(x_train)
x_test = scaler.fit_transform(x_test)
#validation_data = scaler.fit_transform(validation_data)

# Prepare targets
# Set districts having at least 3 ATMs as Goods and the rest as Bads
target['target_al_3'] = 0
for i in range(0, X.shape[0]):
    if(pd.to_numeric(target['ATM_SAYISI'].iloc[i])>2):
        target['target_al_3'].iloc[i] = 1

# Create train test target
#y_validation = target['target_al_3'][0:1000]
#train_test_target = target['target_al_3'][1000:]

y_train, y_test = train_test_split(target['target_al_3'], test_size=0.2,
random_state=0)
import seaborn as sns
# Check out the distribution of Goods and Bads
sns.countplot(target['target_al_3'])

pca = PCA(random_state=0, whiten=True)
```

```
pca.fit(x_train)
exp_var_cum = np.cumsum(pca.explained_variance_ratio_)

import matplotlib.pyplot as plt
# With 6 principal components we get the 85 percent of variance
# With 8 principal components we get the 90 percent of variance
plt.step(range(exp_var_cum.size), exp_var_cum)
plt.title("PCA Result")
plt.xlabel('Component Number')
plt.ylabel('Cumulative Proportion of Variance Explained')
# Let's go with 8 principal components
pca = PCA(n_components=8, random_state=0, whiten=True)
pca.fit(x_train)
#train.describe()
x_train_pca = pca.transform(x_train)
x_test_pca = pca.transform(x_test)
#validation_pca = pca.transform(validation_data)

# Let's see how discriptive PC1 and PC2 on below plot
plt.scatter(x_train_pca[:,0], x_train_pca[:,1], c=np.array(y_train),
cmap='prism', alpha=0.4)
plt.title("Classification based on PC1 & PC2")
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Componene 2')
########################################

# Remove below features since we already have normalised values of these
features
data =
data.drop([                          ,                    ,'totalab',
,'TOPLAM_KONUT_ISYERI','KonutOran','IsyeriOran','NufusOranGenc','NufusOra
nOrta','NufusOranYasli','NufusGenc','NufusOrta'], axis=1)
data =
data.drop(['NufusOranEgitimBilimi','NufusOranEgitimIlk',
rta',                    ,'EGITIM_TOP','hane_geliri',
  ], axis=1)

# Set districts having at least 5 ATMs as Goods and the rest as Bads
# After exploration of data I decided to use at least 3 ATMs as Goods and
that's why the other possible targets are commented out
#target['target_al_5'] = 0
#for i in range(0, data.shape[0]):
#    if(pd.to_numeric(target['ATM_SAYISI'].iloc[i])>4):
#        target['target_al_5'].iloc[i] = 1

################################################
########## Start Building Model ################
################################################
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
from sklearn import metrics

logreg = LogisticRegression()

#  Fit the model and make predictions
logreg.fit(x_train_pca, y_train)
predictions = logreg.predict(x_test_pca)
```

```
# Use score method to get accuracy of model
train_score = logreg.score(x_train_pca, y_train)
print("Train set accuracy: {:.2f}".format(train_score))
test_score = logreg.score(x_test_pca, y_test)
print("Test set accuracy: {:.2f}".format(test_score))

print(y_test.mean())
# The mean of y_test (target test) is 0.28 which shows that if always
said No we'd come up with an accuracy of 0,72 and our model has a better
score of 0,88

#  Here is the ROC Curve of the model
#logit_roc_auc_validation = metrics.roc_auc_score(y_validation,
logreg.predict(validation_pca))
# ROC plot for Train set
logit_roc_auc_train = metrics.roc_auc_score(y_train,
logreg.predict(x_train_pca))
fpr, tpr, thresholds = metrics.roc_curve(y_train,
logreg.predict_proba(x_train_pca)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc_train)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Train ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
# ROC plot for Test
logit_roc_auc_test = metrics.roc_auc_score(y_test,
logreg.predict(x_test_pca))
fpr, tpr, thresholds = metrics.roc_curve(y_test,
logreg.predict_proba(x_test_pca)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Logistic Regression (area = %0.2f)' %
logit_roc_auc_test)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Test ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()

# Confusion Matrix
cm = metrics.confusion_matrix(y_test, predictions)
print("Confusion matrix of test data predictions: ")
print(cm)
print("Percentage of known Bads: " + str(cm[0][0] / (cm[0][0] + cm[0][1])
* 100))
print("Percentage of known Goods: " + str(cm[1][1] / (cm[1][0] +
cm[1][1]) * 100))

# Precision, Recall, f1-score, support
```

```
print(metrics.classification_report(y_train,
logreg.predict(x_train_pca)))
print(metrics.classification_report(y_test, predictions))


# Print coefficients
for i in range(0, x_train_pca.shape[1]):
    print("PC" + str(i) + ": " + str(logreg.coef_[0][i]))

xpca = pd.DataFrame(x_train_pca, range(3603), ['PC1', 'PC2', 'PC3',
'PC4', 'PC5', 'PC6', 'PC7', 'PC8'])
xpca['target'] = y_train
xpca = xpca.drop('target', axis=1)
data = data.drop('rand_index', axis=1)

plt.matshow(data.corr())
plt.title("Correlations after Data Cleaning - 35 Variables")
plt.matshow(X.corr())
plt.title("Correlations after Normalization - 35 Variables")
plt.matshow(xpca.corr())
plt.title("Correlations after PCA - 8 Variables")


########## DECISION TREE MODEL ########
dectree = DecisionTreeClassifier(criterion="gini", random_state=0,
max_depth=3, min_samples_leaf=5)
dectree.fit(x_train_pca, y_train)

predictions = dectree.predict(x_train_pca)
metrics.accuracy_score(y_train, predictions)
metrics.roc_auc_score(y_train, predictions)

dectree_roc_auc_train = metrics.roc_auc_score(y_train,
dectree.predict(x_train_pca))
fpr, tpr, thresholds = metrics.roc_curve(y_train,
dectree.predict_proba(x_train_pca)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='Decision Tree (area = %0.2f)' %
dectree_roc_auc_train)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Train ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()

predictions = dectree.predict(x_test_pca)
metrics.accuracy_score(y_test, predictions)
metrics.roc_auc_score(y_test, predictions)
metrics.confusion_matrix(y_test, predictions)
metrics.classification_report(y_test, predictions)
# ROC plot for Test
dectree_roc_auc_test = metrics.roc_auc_score(y_test,
dectree.predict(x_test_pca))
fpr, tpr, thresholds = metrics.roc_curve(y_test,
dectree.predict_proba(x_test_pca)[:,1])
```

```
plt.figure()
plt.plot(fpr, tpr, label='Decision Tree (area = %0.2f)' %
dectree_roc_auc_test)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Test ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()

################ SVM ##############################
from sklearn import svm

mysvm = svm.SVC(kernel="linear", C=1.0)
mysvm.fit(x_train_pca, y_train)

predictions = mysvm.predict(x_train_pca)
metrics.accuracy_score(y_train, predictions)
# ROC plot for Test
svm_roc_auc_train = metrics.roc_auc_score(y_train,
mysvm.predict(x_train_pca))
#fpr, tpr, thresholds = metrics.roc_curve(y_train,
mysvm.predict_proba(x_train_pca)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='SVM (area = %0.2f)' % svm_roc_auc_train)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Test ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()


predictions = mysvm.predict(x_test_pca)
metrics.confusion_matrix(y_test, predictions)
metrics.classification_report(y_test, predictions)
# ROC plot for Test
svm_roc_auc_test = metrics.roc_auc_score(y_test,
mysvm.predict(x_test_pca))
#fpr, tpr, thresholds = metrics.roc_curve(y_test,
mysvm.predict_proba(x_test_pca)[:,1])
plt.figure()
plt.plot(fpr, tpr, label='SVM (area = %0.2f)' % svm_roc_auc_test)
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Test ROC plot')
plt.legend(loc="lower right")
plt.savefig('Log_ROC')
plt.show()
```