**Report based on researches conducted on Prostitution legality dataset**

Bohdan Darmits

**Preprocessing:**

The main Idea was to perform different Data Processing methods and compare the results by using them for predictions. Also, I tried to represent various methods of prediction.

Firstly, Data Cleaning and Missing Data Imputation was done. Since I had few datasets with different number of rows, many data was wrong and uncompleted at this stage due to the large amount of missing data. Fortunately, all, wrong data and non-standard representation of the same data disappeared at this stage. Then a few cells were left empty. To fill them, the Substitution method was used. I found almost all needed data in internet and filled the gaps. The data I was not able to find was replaced by None values. I decided to replace nones with the mean value. There is a list of other methods of imputing missing values, such as: Hot deck imputation – when user needs to find all the sample subjects who are similar variables, then randomly choose one to replace the missing one; Cold deck imputation – made by systematically chosen value from an individual who has similar values on other variables; Regression imputation – prediction of value based on other variables; Stochastic regression imputation – same prediction plus random residual value. All these methods were investigated and tried. The conclusion, made after this step is that there is a very small number of Nones, and these methods doesn’t make big influence on further steps. That’s why, work done next was based on data where gaps are filled with mean values. In the end, I created 2 copies of my dataset. The first one, where I changed missing values on mean values and the second one for comparing and testing further predictions on dirty data to check if my data processing was done correctly.

The final dataset was a result of combination of 4 datasets and an information collected from additional resources. This is Data Integration. At this part, all required work was done. Firstly, three main techniques types of data mapping were tried. Manual Data Mapping, Semi Automated Data Mapping and Automated Data Mapping. For this internet resources and SQL tables were used. Secondly task – schemas matching. At this step five approaches were discovered: Database Schema Matching Using Machine Learning with Feature Selection - this method improves on its predecessors by including one-to-one attribute matching rather than just matching one attribute with a set of possible attributes, it still has the same problem that it does not consider the possibility of one attribute matching to a set of attributes; Semantic Integration in Heterogenous Databases using Neural Network - this method implemented schema matching using Machine Learning approach. Known as SemaInt. Provides with a similarity mapping of each attribute in one schema with a set of attributes in another, it does not consider the fact one might map to a set of others as well; Corpus-based Schema Matching - this method considers one to one matching of attributes and cannot make mappings like one to many or many to one; Generic Schema Matching with Cupid - a technique of matching which is schema based and not instance based. In the proposed method, heirarchical schemas are represented as trees and nonheirarchical schemas are generalied as graphs. This method maintains a thesaurus for finding linguistic similarity and also makes use of information like schema structure and relation of attributes with each when assigning scores; iMAP:Discovering Complex Semantic Matches between Database Schemas - iMAP is a new method of performing both one to one and one to many schema matching by converting the matching problem to a search problem in a relatively large search space of all possible schema mappings. For efficient searching custom searchers based on concatenation of text, arithmetic operations over numeric attributes etc. are used, and scoring each match to find the best possible matchings. Since the searchers are customized over type of data, they only search through a subset of search space and by this reducing system complexity. This method achieves one to many mapping. It requires a domain expert for creating custom searchers specific to a particular type of database. The method also makes use of only the data contained in the tables and not the schema names themselves. Within Data Integration stage, finding redundant attributes was done by correlation and the covariance using information from lectures - Pearson’s product moment coefficient for numeric data and covariance for numeric data. There was no need of χ2 test because it is used when data in nominal. And Detecting Tuple Duplication and Inconsistency was done by discovering and testing different methods, such as editdistance.py lib from lecture, string comparison algorithm, Q-grams.

The third step, to talk about is Data Transformation. All methods from lecture and a few from additional sources were discovered. The biggest attention was given to Rank Transformation and BoxCox transformation. Rank is good to obtain a variable that is will behave like a normally distributed one. BoxCox has similar intentions. It aims to transform a continuous variable into an almost normal distribution. It doesn’t like negative values. To apply algorithm to them we should modify the formula a bit. These last two methods were implemented and tested in my work.

How do I unify and scale data? Data Normalization – the answer and the forth step. Insertion, Updation and Deletion problems – is the list of what might happen without proper normalization. In notebook a few ways of Data Normalization were applied. A normalization similar to MinMax but realized only by formula. A MinMax function taken from .py library. Min-max normalization has only one significant con - it does not handle outliers. If set will have 99 variables in one range and one variable out of it – this one will have a great affect on transformation especially if it is much bigger than others. Z-score normalization is next our step. Is should be applied when the minimum or maximum values of attribute are not known and as a result the min-max normalization is infeasible, and when the presence of outliers can bias the min-max normalization. The next normalization way, that was implemented – is Decimal Scaling Normalization. This one is generally used in data mining. But shows good results wherever there is a need to normalize data from disparate sources.

Noisy data finding. To find outliers, the LOF algorithm. was implemented. Local outlier factor algorithm idea is based on local density idea, where we locally take n nearest neighbors and use their distance to estimate density. After this we compare densities of an object and its neighbors to identify points that have lower density that their neighbors. And these points are outliers.

**Prediction:**

The prediction goal was to try to predict whether the prostitution in country is legal or not, and also answer on question: Does the values such as gdp, hpi, literacy lever, etc. have influence on acceptance of prostitution?

For this task, the prostitutionStatus column in dataset was changed by the following principle: If prostitution is Illegal or corresponding country uses Swedish model – 1, all others countries – 0, because the accept prostitution in different types.

Predictions were made using Linear regression, KNN, PCA, LDA.

Linear regression – is used to show relations between one dependent variable and one or few independent. Our dependent variable was prostitution Status, independent – all others, except country and region. Linear regression is good when the relationship to between covariates and response variable is known to be linear. A clear disadvantage is that Linear Regression simplifies many problems. And that’s why can’t give a clear result with high accuracy. Very often covariates and response variables don’t exhibit a linear relationship.

KNN - supervised machine learning algorithm. It calculates the distance of a new data point to all other training data points. The distance can be of any type like Euclidean or Manhattan. Then the algorithm selects the K-nearest data points, where K can be any integer. Finally, it assigns the data point to the class to which the majority of the K data points belong. Pros: Since the algorithm requires no training before making predictions, new data can be added during work; It is lazy learning algorithm and therefore requires no training prior to making real time predictions. That’s why this algorithm is vey fast. Cons: The KNN algorithm doesn't work well with high dimensional data because with large number of dimensions, it becomes difficult for the algorithm to calculate distance in each dimension; The KNN algorithm has a high prediction cost for large datasets. This is because in large datasets the cost of calculating distance between new point and each existing point becomes higher; KNN algorithm doesn't work well with categorical features since it is difficult to find the distance between dimensions with categorical features.

We don’t use categorical values and our dataset is very small (100 rows) that’s why I assume that KNN will show better results compared to MLR, in all cases for my dataset.



Uncleared data – data that wasn’t processed by normalization technics, with missing values left, replaced by 0.

Min Max Normalization 1 – min max implemented by formula of min max respectively.

Min Max Normalization 2 – min max take from .py library

Z score, Decimal Scaling – refer to corresponding Normalization algorithms.

It is obvious that rank transformation gives same results for all sets because normalization doesn’t change rank of element, but I decided to test all options, as we can see, the results are pretty much the same.

Assumption about KNN vs MLR vas correct, KNN shows way better result in all cases. As we can also see MLR gives the smallest accuracy for Uncleared Data. It says that data preprocessing was done correctly.

In average z-score normalization shows best result on such dataset. Min-Max – the lowest result, because of the outliers , that minmax can’t handle.

PCA - Principal Component Analysis is a technique that is used to transform high dimensional data to low dimensional by selecting the most important features that capture maximum information about the dataset. The training time of the algorithms reduces significantly with smaller number of features. But it is mandatory to normalize features before applying PCA

LDA – Linear discriminant analysis is one more technique for reducing dimension level of the set. Unlike PCA, LDA tries to reduce dimensions of the feature set while retaining the information that discriminates output classes. LDA tries to find a decision boundary around each cluster of a class. Then LDA projects the data points to new dimensions in a way that the clusters are as separate from each other as possible and the individual elements within a cluster are as close to the centroid of the cluster as possible. The new dimensions are ranked on the basis of their ability to maximize the distance between the clusters and minimize the distance between the data points within a cluster and their centroids. These new dimensions form the linear discriminants of the feature set. In simple words, we just need to calculate the separability between different classes, calculate the variance in all classes, construct the lower-dimensional space that maximizes separability and minimizes variance…

PCA is an unsupervised algorithm. It ignores class labels and wants to find the principal components that maximize variance in a given set of data. Linear Discriminant Analysis, on the other hand, is a supervised algorithm that finds the linear discriminants that will represent those axes which maximize separation between different classes.

After applying these two algorithms to the sets, I will try to make predictions by using random forest classification algorithm.

Random Forest Algorithm: Pick N random records from the dataset. Build a decision tree based on these N records. Choose the number of trees you want in your algorithm and repeat previous steps.

In case of a regression problem, for a new record, each tree in the forest predicts a value for Y (output). The final value can be calculated by taking the average of all the values predicted by all the trees in forest. Or, in case of a classification problem, each tree in the forest predicts the category to which the new record belongs. Finally, the new record is assigned to the category that wins the majority vote. Pros: The random forest algorithm is not biased. It works well when dataset has both categorical and numerical features. The random forest algorithm also works well when data has missing values or it has not been scaled well.



PCA also shows responsibility of each principal component for variance:

MinMax: [0.24954494 0.17015967 0.16877637 0.16877637 0.15797572 0.08476693

0.]

MinMax Transformed: [0.43477592 0.17327978 0.16605036 0.10174327 0.05919596 0.03663477 0.02831995]

Z score: [0.32977919 0.1739652 0.14606763 0.12415746 0.0851884 0.08122084

0.05962127]

Z score Transformed: [0.43574895 0.18501489 0.16204613 0.10015973 0.05429565 0.0366148 0.02611983]

From this information we can say the percent of the classification information contained in set of corresponding features.

From second table we can see that PCA helps us to get accuracy almost as good as KNN. And PCA after Transformation gives way better results. Around 90% in all cases. This is the best result I got during my experiments.

To get better results we need make bigger dataset, and also fix problems with outliers.