Predicting happiness

Hristo Buyukliev

Sofia University

Economics, 2466

09.02.2017

Table of Contents

1 Introduction

1.1 Problem description

For my Survey Analysis class, I chose to use the World Values Survey (wvs) dataset. More specifically, I chose to predict how happy each interviewee is based on their interview.

1.2 Related literature

I chose the topic of this paper after reading similar analyses from students of Sofia University - [1] and [2]. While very good, I had some issues, mostly related to:

- Data preparation – Filipov made some pretty arbitrary choices, which, while common in the financial industry, are neither intuitive nor mathematically grounded.
- Algorithm comparison – I think that multiple algorithms applied reveal more about the data, and make it less likely to commit a fatal mistake
- Evaluation metric – Ganchev used r^2 which is not appropriate for categorical data, and Filipov used Roc-auc, which is only applicable for binary variables
- Training-validation-test split were not explicitly (or at all) defined.
- Overt focus on specific variables. Fixating on only some variables throws away a lot of information.

2 Exploratory data analysis and cleaning

2.1 Exploratory data analysis

The original data has 411 features. Most of those seem to be categorical, with 394 features having no more than 15 unique values. For all categorical variables, the values ranging from -5 to -1 indicate lack of response for whatever reason. I grouped them all together as missing values.

2.2 Data cleaning

There were some interviewees that didn't answer the question of how happy they were, so I left them out. After that, there were about 90 000 interviews. I shuffled them and left out 30000 for the final test dataset. I didn't explicitly split the training data into training and validation, because I used cross-validation for further experiments.

The categorical variables with more than 15 possible values seem to be open-ended questions with hundred of possible answers – such as country of birth, religion of parents, maternal language and so on. If dummyfied, they would each make hundreds of features. Given that the dataset was not that big, it definitely could not handle that. I could figure out a denser representation – e.g. group countries by continent, then by region and so on – but that seemed like too much work for too little signal.

2.3 Data representation

There are some algorithms that work better when the data satisfies some conditions. For example, KNN classifier works better when the dimensions of the input are lower. Logistic regression does not work well or at all with categorical and ordinal data. Neural networks work better when the input is normalized. For all these reasons, I prepared two transformations on the original dataset.

- First, the original dataset with 411 columns is dummyfied, where each categorical variable is converted to binary variables. The resulting dataset has 2149 columns.
- Secondly, the dummyfied dataset is reduced via PCA to p dimensions.

After those transformations, there exist 3 different representations of the input data. I'll refer to those as X, X_{dummy}, and $X_{PCA\text{-}P}$.

3 Metric selection

In supervised learning there can be many algorithms, all of which try to predict some target *y* given some input *X*. Given their predictions $\hat{y}$, they try to minimize some *distance* or *loss* between *y* and $\hat{y}$.

3.1 Area under curve

Filipov [1] used *area under curve*. This metric is also know as "receiver-operating characteristic" or "roc-auc". It works by plotting the trade-off between true positive rate and false positive rate at different threshold levels. An example usage is shown in figure 1. It's main drawback is that it only works when *y* is binary. Filipov circumvented the problem by only using the respondents which were either very happy or very unhappy. This metric, besides the significant drawback of only using about a third of the data, also artificially inflates the algorithms' performance. For example, a k-neighbors classifier would score over 90% accuracy when predicting only two classes, but when scoring all for classes would barely improve on the benchmark.

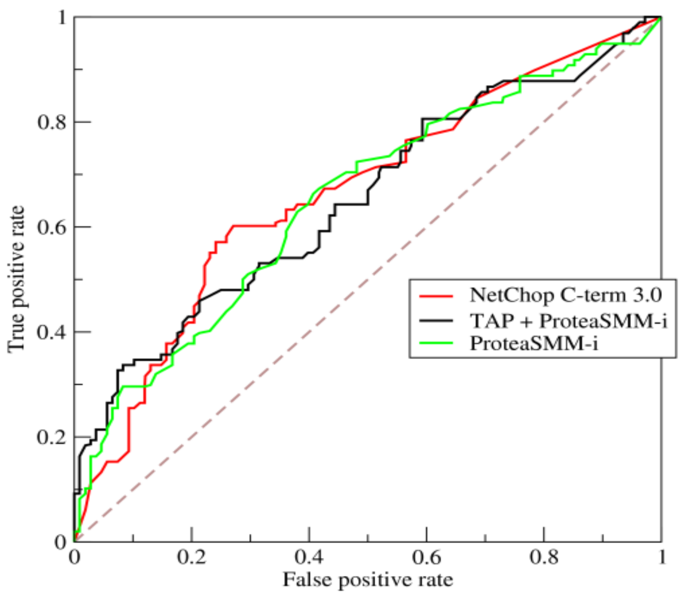

Figure 1: Receiver-operator curve

Roc-auc scoring of some predictions moves between 0.5 (random guess) and 1 (perfect guesses). A scoring below 0.5 is possible, but it means something went very wrong.

3.2 Accuracy

Accuracy is the simplest metric possible when scoring classifiers. It suffers from two fundamental flaws:

- When the data has very unbalanced labels, e.g. 96% of class "A" and 4% of class "B", using raw accuracy encourages most algorithms to always predict "A". Those models are of course useless.
- Accuracy is a "hard" metric, in the sense that a guess is either right or not. For example, if classifier "A" predicts [very happy: 35%, very unhappy: 65%], and classifier "B" predicts [very happy: 5%, very unhappy: 95%], and the true label is "very unhappy", they score the same, even though "B" is much more confident.

On the other hand, accuracy is very intuitive and interpretable. It varies from 0 (worst possible score) to 1 (perfect guesses). A common benchmark is always predicting the most common class, which in the case of 4 classes would yield accuracy of around 0.51.

3.3 Categorical cross entropy

Categorical cross entropy comes from the field of Information Theory. It aims to measure the distance between two probability distributions.

The entropy of a discrete variable *X* under a distribution *p* is defined as:

$$H(X)=\sum p(x_i)\log\frac{1}{p}(x_i)$$

Cross-entropy of two distributions *p* and *q* is defined as:

$$H_p(q)=\sum q(x)\log\frac{1}{p}(x)$$

Notice that cross-entropy is *not* symmetrical. It does alleviate the two main flaws of the accuracy measure.

3.4 Benchmark

When scoring accuracy, the simplest benchmark is to predict the most common class – in this case, “unhappy”. This benchmark would score an accuracy of around 0.51.

4 Algorithms used

4.1 K-nearest neighbors

K-nearest neighbors (KNN) classifiers are among the simplest classifiers possible. The idea is for each interviewee's happiness we want to predict, to take the k other respondents who are the most similar, and average their guesses. KNN algorithms suffer acutely from the curse of dimensionality [3], so I've used them on the PCA-reduced data.

4.2 Logistic regression

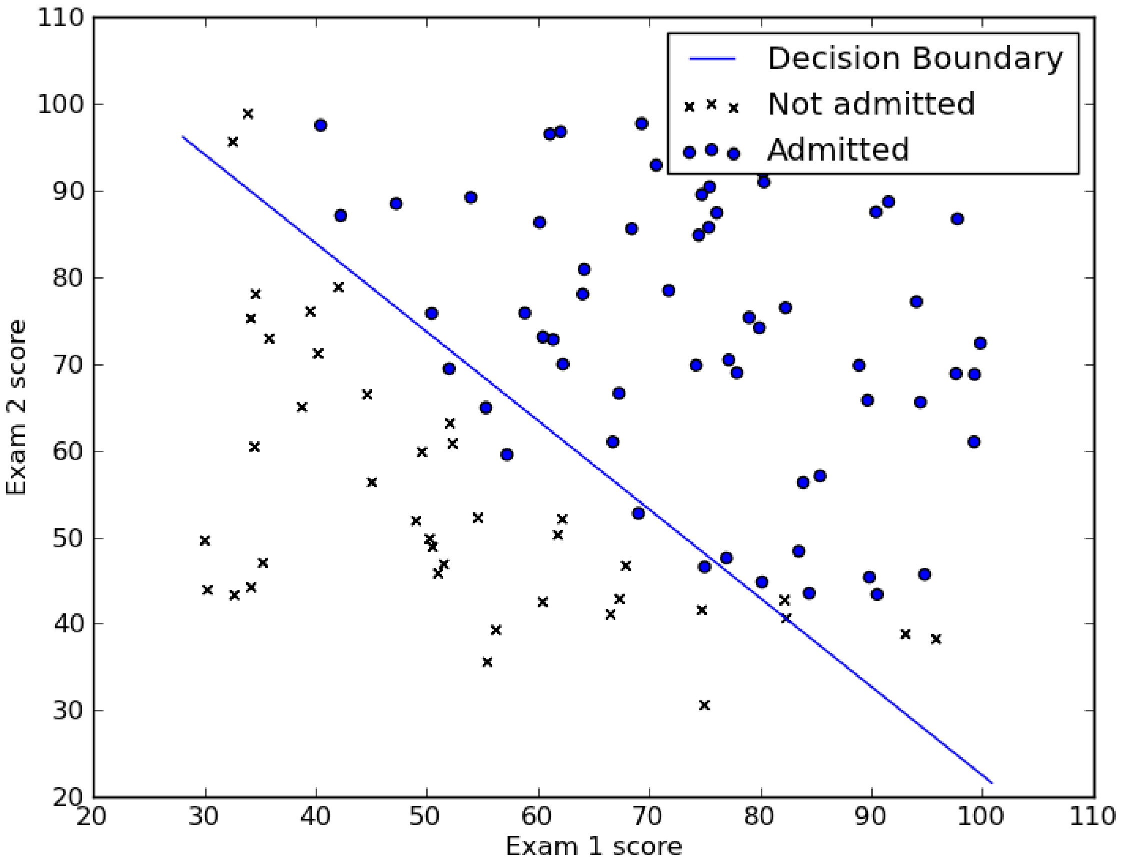

Figure 2: Logistic regression and decision boundary between two classes.

Logistic regression is the same as linear regression, only with some nonlinearity Q at the end. For binary classification, typically the sigmoid is used, and for multiclass classification, a softmax function. In figure 2, the decision boundary of logistic regression on medical data is shown.

4.3 Feed-forward neural networks

Feed-forward neural networks(a.k.a. multilayer perceptrons) are a natural extension to logistic regression. An example structure is shown in figure 3.

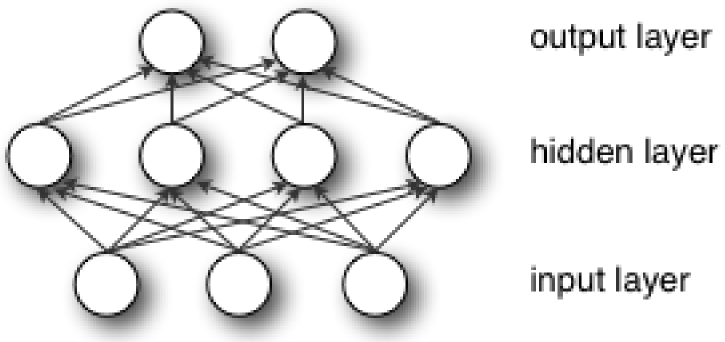

Figure 3: A classical feed-forward network

In practice, feed-forward networks are stacked logistic regression. In practice the nonlinearity (called *activation*) after each hidden layer is usually not sigmoid, as some other nonlinearities are found to work slightly better.

Notice that if a multilayer perceptron has no hidden layers, it is the same as logistic/linear regression. Therefore, the family of functions a perceptron can represent is a superset of the functions regression can. That means that neural networks can model much more complicated relationships in the data. However, they are much more likely to overfit, so typically they require more data.

4.4 Support vector machines

Support vector machines work similarly to logistic regression, although they are not maximizing the posterior class probability, but some margin between *support vectors*. In figure 4 are shown data that must be linearly separated, and the three support vectors. The decision boundary does not need to be linear. In figure 5 are shown different kernels. In the final comparison, I'll use three kernels: linear, polynomial, and radial basis.

Notice that, in contrast to logistic regression, SVMs do not explicitly define probability. For this reason, I'll only mention accuracy in the results section.

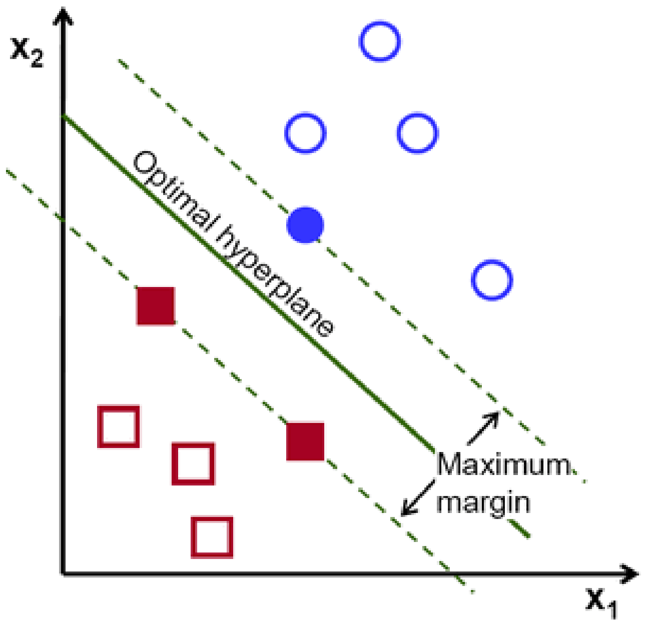

Figure 4: Support vector machines maximize the margin between the decision boundary (dotted line) and the support vectors (bolded points).

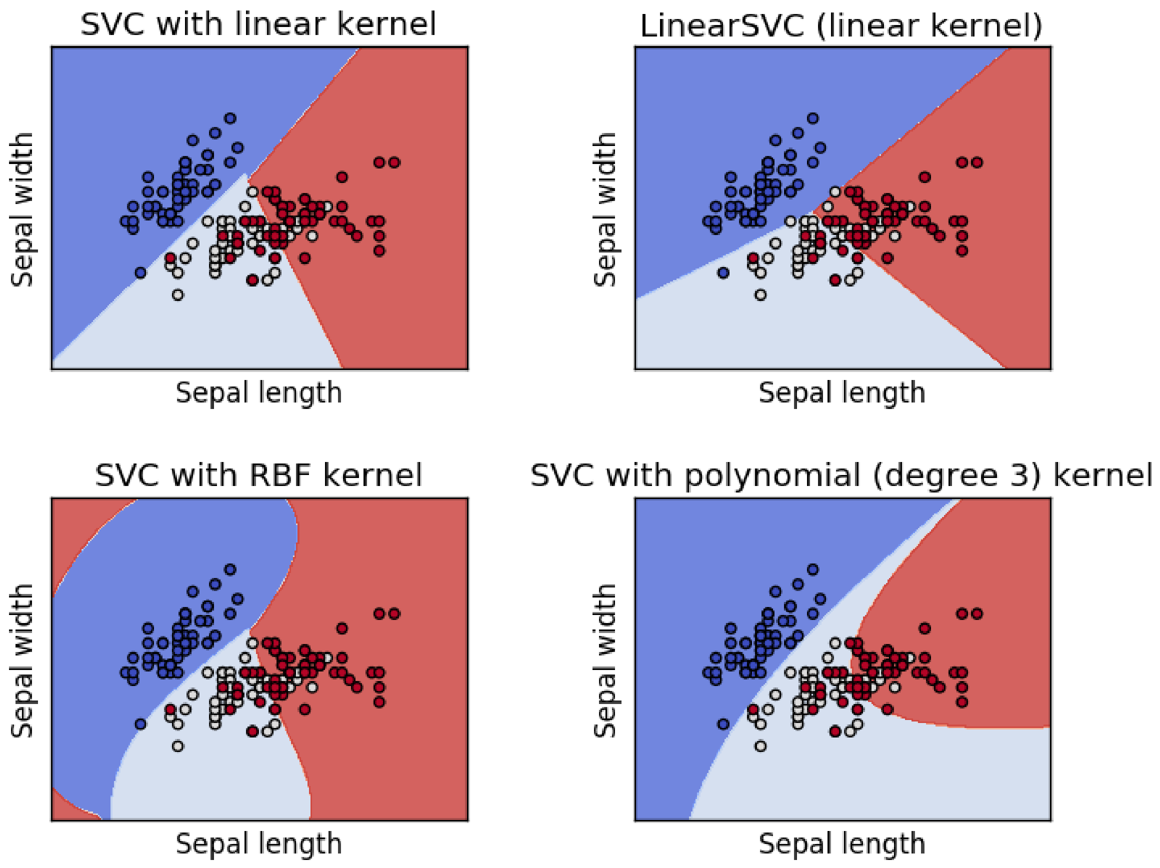

Figure 5: Different kernels' decision boundaries visualized. SVC stands for support vector classifier.

4.5 Decision trees, random forests and boosting

4.5.1 Decision tree classification

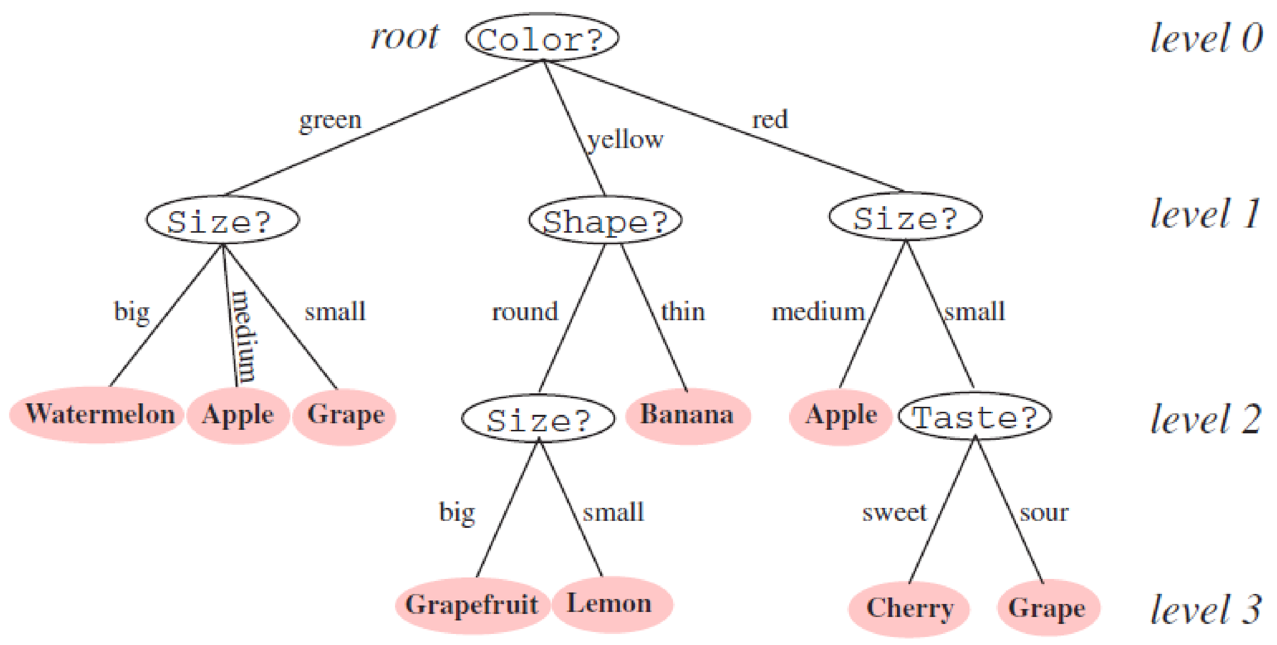

Figure 6: Decision tree for classifying fruit.

Decision trees are a non-parametric model, that aims to infer simple yes-no decision features based on the input data. In it's simplest variant, a decision tree recursively splits the data, so that the sum of entropies of the two resulting subsets is minimized. In figure 6 an example of a decision tree that classifies fruit is given. Decision trees and their extensions naturally work better with categorical data, and performed very well with the happiness dataset.

4.5.2 Random forests

Multiple simple classifiers can be averaged, to achieve higher accuracy (or whatever other metric) than any of the single classifiers. In this case we say that there is an ensemble of classifiers. Random forests create multiple decision trees and average their predictions.

4.5.3 Boosting decision trees

Boosting represents a more complicated form of ensembling. The difference is weak learners (in this case decision trees) are iteratively learned, and after each is wearned, data importance is reweighted, so that hard to learn examples are more important. There exist many variants of boosting algorithms, and here I've used xgboost.

4.6 Naive Bayes classifiers

Naive Bayes assumes all the features are independent (which is why it is called naive) and leverages Bayes' theorem for all of them. As a result, whenever there are many dimensions, the classifier becomes “too confident” -

predicting probabilities too close to either 0 or 1. This means it performs awfully on the crossentropy metric. Another flaw, is that when presented two very similar features – or even the same feature twice – the algorithm does not recognize they are connected. In the case of the WVS dataset, there are multiple questions that are very similar – e.g. “are you religious”, “do you approve of gay marriage/abortion/women's rights”, etc. which are all caused by some underlying causes. Treating all features as independent means that those causes are unfairly weighted. As a result, naïve bayes on X or X_{dummy} often can not outperform even the benchmark.

5 Results and conclusion

As expected, decision trees performed relatively well, given that the data is mostly categorical. Note how single decision tree barely improves on the benchmark, but ensembling multiple trees outperforms all other algorithms by a large margin. Neural networks, of course, beat logistic regression, but adding a layer did not improve performance. This suggests that more data is needed. KNN performed surprisingly well, while Naive Bayes disappointed. In the table below are the results.

Note that in the "train" column, the score is calculated via 3-fold cross-validation on the training data. Neural networks were an exception, because of their slow train-time. The test results were calculated only once, *after* finishing all experiments, doing grid-search on hyperparameters and so on. Bolded are the algorithms that performed best on the train and on the test sets. Test performance seemed pretty consistent with, even slightly better than the training (probably due to the 1.5 times more data).

Just for comparison, Filipov reported .85 AUC score. Xgboost delivered 0.97 right off the bat, with no hyperparameter gridsearch. Granted, I was working on wvs's sixth wave, while he was using the fifth.

The code is uploaded to GitHub.

Algorithm	Dataset	Accuracy – CV	Accuracy - test
KNN, k=50	$X_{PCA\text{-}30}$	0.574	0.580
SVM, kernel=linear	$X_{PCA\text{-}30}$	0.561	0.564
SVM, kernel=poly	$X_{PCA\text{-}30}$	0.571	0.577
SVM, kernel=rbf	$X_{PCA\text{-}30}$	0.580	0.587
Logistic regression	X_{dummy}	0.568	0.570
ANN, 1 hidden layer	$X_{PCA\text{-}30}$	0.611	0.610
ANN, 2 hidden layers	$X_{PCA\text{-}30}$	0.606	0.614
Bernoulli NB	X_{dummy}	0.482	0.477
Gaussian NB	$X_{PCA\text{-}30}$	0.342	0.340
Decision tree	X	0.532	0.536
Random forest	X	0.637	0.643
xgboost	X	**0.644**	**0.645**

Appendix A: References and Bibliography

Bibliography

1: Petar Filipov, Приложение на финансови иконометричнитехники в моделиране на човешкото щастие, 2014

2: Georgi Ganchev, PREDICTING HAPPINESS - Evidence from Bulgaria, 2016

3: Indyk, Piotr, and Rajeev Motwani., Approximate nearest neighbors: towards removing the curse of dimensionality, 1998