CMPS142, Machine Learning
March 23, 2017
Alina Syrtsova
asyrtsov@ucsc.edu

Project: Classifying Survey Results

Team members:

I did not have team members for this project because I hadn't asked anyone by the time the project proposal was due, so I worked on it alone. I am an undergraduate and a senior Computer Science major. I fulfilled all the roles necessary for this project, with the great help of the scikit-learn machine learning library for python.

Dataset:

The data used in this project was found on kaggle.com, in a dataset called "Young People Survey." It was conducted by a group of statistic students at the Comenius University in Bratislava, with the respondents being the students and their friends, all aged 15 to 30. The survey consists of 150 questions, which could be split into eight categories: music preference, movie preferences, hobbies and interests, phobias, health habits, personality traits/views on life/opinions (the largest category), spending habits, and demographics.

139 of the questions were asking the participant to rate whether they enjoy/agree with or dislike/disagree with the the subject of that question, on the discrete scale of 1 to 5; the other 11 questions were categorical, though all those outside of the demographics section could still be scaled, for example: never drank/ social drinker/ drink a lot. Notably, the data contained missing values where participants skipped questions. There was a total of 1010 responses.

Problem and Hypothesis:

I was curious to see how well the gender of the participants can be predicted by their answers to the survey's questions, and what kind of personality trends can be extracted from the data. More concretely, I set out to answer the following questions:

1. How strongly are the recorded preferences and personality traits correlated with either male or female gender? What is the best level of accuracy I can get?

2. Which machine learning methods work best to answer the first question, and what does this imply about the data?

3. Using the method which gives us the best result for predicting gender, what kind of accurate prediction rates can we get with the other binary features provided in the demographics section? Specifically, these are: Is the participant left-handed or Right-handed? Are they an only child (y/n)? Did they spend most of their childhood in a city or a village?

Methodology I: Procedure

1. **Data cleaning**:

    a. Load the csv file from Kaggle, casting to values to floats or strings, and feed into my function *clean_data*.

    b. Inside *clean_data*, parse through all the rows and their elements. Those that are already valid values 1 through 5, add into the row as is. When the parser reaches the column which contains gender, add 1 for “male” or “0” for female into the list of y’s.

    c. When one of the categorical responses is reached, it is given to a method which converts it into a value 1 through 5, if it matches a valid question and response. These had to be hard-coded, and there are 17 of them (in response to five questions). Those strings that answer demographic questions are replaced with “-1”, because they will be cut out later.

    d. In the case that the response was left blank, a “3” was added as a value (since it is the average value on the scale).

    e. Lastly, the rows were added into the X matrix, and the last 10 columns were chopped away because they contained the demographics values.

2. **Use different binary classification models to classify the data**:

    a. Support Vector Machines
        i. Using linear a linear kernel
        ii. Using a polynomial kernel
        iii. Using a RBF kernel
    b. Multi-Layer Perceptron (Neural Net)
        i. Try with ReLU and tanh
        ii. Adjust batch size
        iii. Adjust learning rate
        iv. Play around with other hyperparameters
    c. Gaussian Process Classifier
        i. RBF kernel

3. **Improve accuracy, and repeat best methods from part 2**

    a. Principal Component Analysis
    b. Cross-Validation

4. **Report the accuracy**

    a. Write results of predictions into a text file.

Methodology II: Rationale

1. **Data cleaning:**

    a. Some parts of my implementation here were not ideal. Specifically, adding a 3 in for all missing values might have caused some inaccuracy for the data. If I had

more time, I would have averaged the values of responses belonging to the correct category (male or female), and replaced missing values with those, so that their effect on the model would be minimized. Another possibility would have been to delete those rows altogether, but I did not want to risk discarding too large a portion of my samples, when I already had relatively few.

b. Similarly, my conversion from categorical answers in the form of strings to discrete values of 1,3, or 5 could have been approached in a different way, either by having those questions discarded, or plotted onto a continuous value on the 1-to-5 scale, when not divisible by 5. As it was, I condensed some groups to have the same value (such as “tried smoking” and “former smoker” both being converted into 3’s, while “never smoked” and “current smoker” turned into 1’s and 5’s.

2. **Binary Classification Models**

    a. I chose to use a Support Vector Machine as my first method because it works well for a dataset of my size, and is a good method to use for binary classification.

    b. A feed-forward Neural Net was my second choice as a classifier, since neural nets are known to be some of the best methods for predicting complex functions, and I did not expect my data to necessarily work well with a linear model.

3. **Improving Accuracy**

    a. PCA. I expected Principal Component Analysis to have a positive effect on my predictions, because my data has 140 features. Many of them likely have low variance, so my models could benefit from collapsing certain features together.

    b. Trying out different methods and manipulating my best model.

Implementation outcome:

Framework:

I used the scikit-learn library in doing the machine learning part of my work. Specifically, *sklearn.nueral_network.MLPClassifier* and *sklearn.svm.SVC*. Both of these have default settings, which I used to get my initial results, but there are also a lot of hyperparameters which could be adjusted. My programming environment was mostly Sublime Text and Windows Command Line.

Previous work:

Since I found the data on a website that is specifically meant for machine learning projects, there are already several available studies of it. Most of them do not focus on gender, although a few do. However, I did not use any of them as reference for my work, except perhaps the

charts in Miroslav Sabo's "Analysing gender differences" kernel (which gave me a sense of which features might provide the most information in my model).

Challenges:

Starting out with the project and trying to process the data was the most challenging part for me. Because I was relatively unfamiliar with python, I had a lot of problems with syntax, not knowing the right methods, and getting confused with datatypes. There was a long period when I thought my predictions for some reason couldn't get better than 50-60%, but shortly before the due date it turned out that my y values and X rows were not lining up correctly.

Also, the limited time given to work on this project restricted the amount of things was able to try out. Although at first I spent a lot of time trying to perfect my work, near the end I realized I did not have time to try out all the different learning and optimization algorithms available to me.

Experiment Results:

1. **Support Vector Machines**

RBF Kernel	Linear Kernel	Polynomial Kernel, k=3
SVM **Train: 200 / 200 = 1.0000** **Test: 715 / 804 = 0.8893** **SVM** **Train: 298 / 300 = 0.9933** **Test: 626 / 704 = 0.8892** **SVM** **Train: 398 / 400 = 0.9950** **Test: 541 / 604 = 0.8957** **SVM** **Train: 498 / 500 = 0.9960** **Test: 444 / 504 = 0.8810**	**SVM** **Train: 200 / 200 = 1.0000** **Test: 655 / 804 = 0.8147** **SVM** **Train: 300 / 300 = 1.0000** **Test: 595 / 704 = 0.8452** **SVM** **Train: 400 / 400 = 1.0000** **Test: 521 / 604 = 0.8626** **SVM** **Train: 500 / 500 = 1.0000** **Test: 417 / 504 = 0.8274**	**SVM** **Train: 200 / 200 = 1.0000** **Test: 695 / 804 = 0.8644** **SVM** **Train: 300 / 300 = 1.0000** **Test: 603 / 704 = 0.8565** **SVM** **Train: 400 / 400 = 1.0000** **Test: 526 / 604 = 0.8709** **SVM** **Train: 500 / 500 = 1.0000** **Test: 426 / 504 = 0.8452**
Best in Testing: 0.8957	**Best in Testing: 0.8626**	**Best in Testing: 0.8709**

The Radial Basis Function kernel worked best for the SVM. And, as expected, linear did worse. Usually the advantage to linear is that it is faster, but that was not a concern for me since my dataset was small.

2. **Multi-Layer Perceptron**

ReLU activation	Sigmoid activation	Sigmoid activation

Solver: 'adam'	Solver: 'lbfgs'	Solver: 'stochastic gd'
MLP **Train: 200 / 200 = 1.0000** **Test: 709 / 804 = 0.8818** **MLP** **Train: 300 / 300 = 1.0000** **Test: 621 / 704 = 0.8821** **MLP** **Train: 400 / 400 = 1.0000** **Test: 539 / 604 = 0.8924** **MLP** **Train: 488 / 500 = 0.9760** **Test: 438 / 504 = 0.8690**	**MLP** **Train: 200 / 200 = 1.0000** **Test: 710 / 804 = 0.8831** **MLP** **Train: 300 / 300 = 1.0000** **Test: 630 / 704 = 0.8949** **MLP** **Train: 400 / 400 = 1.0000** **Test: 539 / 604 = 0.8924** **MLP** **Train: 500 / 500 = 1.0000** **Test: 439 / 504 = 0.8710**	**MLP** **Train: 116 / 200 = 0.5800** **Test: 477 / 804 = 0.5933** **MLP** **Train: 186 / 300 = 0.6200** **Test: 407 / 704 = 0.5781** **MLP** **Train: 247 / 400 = 0.6175** **Test: 346 / 604 = 0.5728** **MLP** **Train: 316 / 500 = 0.6320** **Test: 277 / 504 = 0.5496**
Best in Testing: 0.5642	**Best in Testing: 0.8949**	**Best in Testing: 0.5933**

Interestingly, there turned out to be a lot of variation in how well the Neural Networks performed. While the ReLU/adam iteration and Sigmoid/SGD iteration gave results that were barely better than the guaranteed 50% (since it is a binary problem), the iteration using the logistic activation function and the 'lbfgs' solver (which, according to the sklearn documentation, is "an optimizer in the family of quasi-Newton methods") was a close second behind the RBF kernel.

3. Trying out other things

PCA, components reduced from 140 to 50	Gaussian Process Classification	Logistic Regression L2 Regularizer
SVM **Train: 200 / 200 = 1.0000** **Test: 646 / 804 = 0.8035** **SVM** **Train: 300 / 300 = 1.0000** **Test: 564 / 704 = 0.8011** **SVM** **Train: 400 / 400 = 1.0000** **Test: 491 / 604 = 0.8129** **SVM** **Train: 500 / 500 = 1.0000** **Test: 407 / 504 = 0.8075**	**GPC** **Train: 200 / 200 = 1.0000** **Test: 628 / 804 = 0.7811** **GPC** **Train: 300 / 300 = 1.0000** **Test: 515 / 704 = 0.7315** **GPC** **Train: 400 / 400 = 1.0000** **Test: 466 / 604 = 0.7715** **GPC** **Train: 500 / 500 = 1.0000** **Test: 374 / 504 = 0.7421**	**LR** **Train: 200 / 200 = 1.0000** **Test: 692 / 804 = 0.8607** **LR** **Train: 300 / 300 = 1.0000** **Test: 607 / 704 = 0.8622** **LR** **Train: 400 / 400 = 1.0000** **Test: 522 / 604 = 0.8642** **LR** **Train: 500 / 500 = 1.0000** **Test: 426 / 504 = 0.8452**
Best in Testing: 0.8129	**Best in Testing: 0.7811**	**Best in Testing: 0.8642**

To my surprise, using Principal Component Analysis did not improve my prediction, but made it worse. Not show here, I tried it with values as low as 2, and as high as 100. Having 2 dimensions brought the success rate down to the 50's, while having more components only made it approach the accuracy rate predicted with all 140 features.

Cross Validation on the RBF Kernel SVM produced:
[0.9141791 0.91791045 0.86940299]
[0.92340426 0.91489362 0.86752137]
[0.91089109 0.88059701 0.87064677]
[0.90532544 0.83928571 0.89820359]

(The four rows are for different training/testing set sizes) This shows that the best accuracy I can aim for with my RBF SVM model is in the low 90's.

4. **Predicting other traits**

Left or Right Handed?	**Only child? Y/N**	**Grew up in city or village?**
SVM **Train: 186 / 200 = 0.9300** **Test: 727 / 807 = 0.9009** **SVM** **Train: 280 / 300 = 0.9333** **Test: 632 / 707 = 0.8939** **SVM** **Train: 373 / 400 = 0.9325** **Test: 542 / 607 = 0.8929** **SVM** **Train: 466 / 500 = 0.9320** **Test: 454 / 507 = 0.8955**	**SVM** **Train: 193 / 200 = 0.9650** **Test: 599 / 808 = 0.7413** **SVM** **Train: 289 / 300 = 0.9633** **Test: 526 / 708 = 0.7429** **SVM** **Train: 390 / 400 = 0.9750** **Test: 449 / 608 = 0.7385** **SVM** **Train: 484 / 500 = 0.9680** **Test: 368 / 508 = 0.7244**	**SVM** **Train: 199 / 200 = 0.9950** **Test: 564 / 806 = 0.6998** **SVM** **Train: 294 / 300 = 0.9800** **Test: 500 / 706 = 0.7082** **SVM** **Train: 391 / 400 = 0.9775** **Test: 423 / 606 = 0.6980** **SVM** **Train: 486 / 500 = 0.9720** **Test: 362 / 506 = 0.7154**
0.9009	**0.7429**	**0.7154**

These interesting results imply that being right or left handed is correlated with certain interests and personalities as much as gender is. The environment one was raised in and only-child-status also have some correlation with the personalities of the young people, although not as much as type of handedness and gender.

Evaluation and Conclusions:

1. The best prediction accuracy I was able to get with this data was around 90%. This is a pretty high percentage, but it implies about 10% of those taking the survey do not enjoy the same things, or act the same way, as is typical of others of their gender.

   I expected that reducing the dimensionality of my X matrix would improve results since 140 features is not a small amount. However, Principal Component Analysis actually

made the scores worse, which suggests that the responses for each question weren't very linearly correlated with each other for each responder.

2. The learning model which gave me the best results was the Support Vector Machine with the Radial Basis Function as kernel. A possible reason for this is that the data might have had outliers on one side, which would have weighed the function down in their direction, but seeing as how SVM only cares about support vectors, outliers would not have influenced it if they were correctly classified. However, even if that is not the case, SVMs are a powerful prediction tool, so it makes sense that it worked well on my relatively small dataset, where execution time stayed short.

   As the data was not linearly separable, the SVM was soft-margin and used a penalty term for the errors. The default penalty term for the sklearm method was 1.0. Making it slightly higher or lower did not improve results.

3. Since it was easy to make the necessary alterations in my code, I tried out m models with targets other than gender. It is curious to see that right or left handedness can be predicted as well as gender based on the data. Only-child status prediction is also 24% better than it would be if there was no relationship between that and personality/ interests. I am surprised to see

Future work:

Potential future work on the project can include breaking up the data into the sections given on the questionnaire, to see whether there is an especially high correlation between gender and any of the sections in particular. Also, processing the data in a different way may have led to better results, so that would also be a good thing to try. There are also many more accuracy-improving methods available in the scikit-learn library, and they might be able to lower error rate more.

What I learned:

I gained a lot of experience with Python working on this project. I am now much more familiar with how to do basic things like manipulate arrays and read/write files from the script.

Also, I enjoyed familiarizing myself with the scikit learn machine learning library. Although I did not get to go in depth exploring its functionality during the course of this project, I am now familiar with just how useful the library is. There are tons of optimization methods available to try out, as well as tutorials and guides. I found the flowchart they provided for choosing an algorithm especially interesting. Based on the characteristics of my question, it suggested I use Linear SVC, which turned out to be close to the ideal method I found.

Apart from getting a glimpse into the many different options available for machine learning, I also learned that what I expect to work in theory does not always behave the way I expect. Because of this, it is valuable trying out different things to see if they fit your prediction goals.

Screenshots: