INFX574 Problem Set 4

Your name:

Deadline: Wed, May 16, 5:30

Introduction

This problem set is long but I hope it is still doable :-) It revolves around classification, logistic regression and regularization. At the end, you are asked to construc a ROC curve.

Please submit a) your code (notebooks) *and* b) the results in a final output form (html or pdf). I recommend to use *scikit-learn.linear_model*.

You are welcome to answer some of the questions on paper but please include the result as an image in your final file. Note that you can easily include images in both notebooks and .rmd—besides of the code, both are just markdown documents.

Working together is fun and useful but you have to submit your own work. Discussing the solutions and problems with your classmates is all right but do not copy-paste their solution! Please list all your collaborators below:

1.
2. ...

Wisconsin Breast Cancer Dataset

You will work with Wisconsin Breast Cancer Dataset (WBCD), available at UCI Machine Learning Repository. You can download it from the internet but rather use the files *wdbc.csv.bz2* and *wdbc_doc.txt* from canvas (under *files/data*) where I have added the variable names to the data. The first file is the csv with variable names, the second one a brief description of the data.

The data includes diagnosis of the tumor with "M" meaning cancer (malignant) and "B" no cancer (benign), and 10 features, describing physical properties of cell nuclei from biopsy samples. Each feature is represented three times, once for mean, once for standard error, and once for the worst values. Your task is to predict diagnosis based on this data.

1 Explore the data

As the first step, explore the data.

1. Load the data. You may drop *id* or just ignore it in the rest of your analysis.
2. Create a summary table where you show means, ranges, and number of missings for each variable. In addition, add correlation between the diagnosis and the corresponding feature. You may include more statistics you consider useful.
3. Graphical exploration. Make a number of scatterplots where you explore the relationship between features and the diagnosis.

Note: you may attempt to display all 435 possible combinations as a scatterplot matrix, but that will most likely be just unreadable. Choose instead a few cases with higher correlation. Avoid overwhelming your reader with tens of similar figures.

2 Decision Boundary

The first task is to plot the decision boundary by kNN and by logistic regression. You will also play a little with feature engineering.

If you are uncertain what is decision boundary, I recommend to consult James et al. (2015) book Section 2. For instance, Figure 2.13 on p38 depicts a 2D classification case where certain X_1, X_2 values are classified as orange and others as blue. Decision boundary is the dashed winding line that separates these two regions.

There are two broad strategies to plot it. In any case, you have first to estimate (train) your model. Thereafter you have to predict the classes (cancer/no cancer here) on a regular dense grid that covers the parameter space (this is the small blue/orange dots on figure 2.13). Afterwards you can either plot your predicted values with a certain color code, or alternatively, say, set predicted $M = 1$ and predicted $B = 0$, and make a contour plot for a single contour at level 0.5. You may also combine these both methods.

We ignore training/testing/overfitting issues for now.

2.1 kNN Case

First, let's explore the decision boundary using kNN.

Pick two features. I recommend to use a few that show relative strong correlation with diagnosis. Feel free to use a combination you already plotted above.

1. Predict the diagnosis on a grid (say, 100x100) that covers the range of the explanatory variables. Use kNN with $k = 3..7$ (pick just one value). This gives you 100x100 predicted diagnoses.

   Note: if your features are of very different scale, you should either scale these into a roughly equal scale, or use a metric that does this with you. Consult James et al. (2015, p 217).

2. Plot the actual data and the decision boundary on the same plot. Ensure that actual observations and predictions are clearly distinguishable, and that one can easily understand the color code.

3. Describe your observations. How good is kNN in picking up the actual shape? Does it also pick up noise?

   Note: unless you do cross-validation, you cannot know if the model picks up noise (overfit). Here I just ask your best judgement, not any formal analysis.

2.2 Logistic Regression

Now repeat the process above by logistic regression. Pick the same features as what you used for k-NN above.

1. Fit a logistic regression model with these two features.

2. Predict the diagnosis on a similar grid...

3. ...and create a similar plot.

4. Describe your observations. How does the result for kNN compare to that for Logistic Regression?

2.3 Feature Engineering

So far you were using just two of the existing features in the data. However, now let's create a few more.

1. Use these two features to compute some new ones. Let's denote your original features by x and y. Examples you may create include: $x^2, y^2, x \cdot y, \mathbb{1}(x > 5), \mathbb{1}(y < 1) \cdot x^2, \log x \ldots$ You can use all sorts of mathematical operations as long as a) you only use x and y, not other features, and b) the original and the engineered features remain linearly independent (they are, unless you create features like $\alpha \cdot x + \beta \cdot y$).

2. Fit a logistic regression model. However, this time pick both x, y, and some of your engineered features.

3. Create the decision boundary plot.

4. Comment on the shape of the boundary. What do you think, how well can you capture the actual boundary? What about overfitting? (Again, I ask about your judgement, not about any formal analysis).

5. Repeat the exercise a few times where you pick/engineer different new features, and try to get as reasonable boundary as you can.

   As above, I am asking "reasonable" boundary in the sense of your best judgement. No actual cross validation is necessary.

Note: I have mixed experience with `scikit-learn.linear_model.LogisticRegression`. It occasionally appears the default convergence tolerance is far too big, and the default *liblinear* solver too imprecise. Setting tolerance to 10^{-12} and solver to *lbfgs* improved the results for me, but did not make it flawless.

3 Use the full data

Finally, we'll get serious. We'll use full data set in the logistic regression, include regularization, and cross-validate our results.

3.1 Cross Validation

Here your task is to Write code to do n-fold cross validation. Note: I expect you to implement CV here, not to use any canned version. Consult James et al. (2015, Section 5.1.3).

1. You should do k-fold CV ($k \geqslant 10$) with the following tasks:

   (a) Fit logistic regression model on training data using all existing features. You may also add your engineered features if you wish.

   (b) Calculate accuracy, precision and recall on the validation data.

2. Report the average accuracy, precision, and recall over your CV runs.

3.2 Regularization

Your (almost) last task is to find the best regularization parameter using CV algorithm you created above. Consult James et al. (2015, Section 6.2) for introduction to regularization methods (ridge and lasso). I expect you to play with regularization parameters in existing software, such as *scikit-learn.linear_model*, not to implement it yourself.

1. Pick a type of regularization (lasso, ridge, elastic net).

2. Create a wide list of regularization parameters. And I mean *wide*, for instance ranging from 10^{-6} to 10^6. Pick a number of values λ inside this range.

3. For each λ in this range, repeat the CV process in exercise 3.1 above. Compute accuracy, precision, recall.

   Warning: this may be slow, so I recommend you to start with only a few values. When you algorithm works well, increase the number of values.

4. Report the results as a function of λ. This may be in the form of a table or a graph.

5. Report the best regularization parameters, and the best results.

4 ROC curve: which estimator is the best

Your last task is to create the ROC curve of several estimators. Consult James et al. (2015, p 148) for what is the ROC curve. You have to compare a) k-NN, b) Naive Bayes, and c) logistic regression estimators. In case of k-NN, you should pick a few different k values (say, 1, 5 and 25). In case of Naive Bayes, you may use use *sklearn.naive_bayes.GaussianNB*. Feel free to use it "as is", no calibration necessary.

1. Split your data into testing and training sets (you may use *sklearn.model_selection.train_test_split*), say 1/3 for testing. Use all the features in the original data.

2. Using your training data, estimate (a few) k-NN models, Naive Bayes model, and logistic regression.

   Notes: in case of k-NN, you should use scaled features. In case of logistic regression, use the optimal regularization parameter value you found in 3.2.

3. For each of the models, predict the probabilites on the test data.

   Note: you have to use *predict_proba*, not *predict* for logistic and NB regression.

4. Pick a number of thresholds between 0 and 1 (for instance, 0, 0.1, 0.2, ...). For each model and each threshold, treat your predictions to be 1 if it it's probability is at least as big as the threshold.

5. Based on these predictions, compute false positive rate and true positive rate.

6. Plot these rates for each threshold and model.

7. Comment your results. Which model is the best?

References

James, G., Witten, D., Hastie, T., Tibshirani, R., 2015. An Introduction to Statistical Learning with Applications in R. Springer.