EPFL
ENAC TRANSP-OR
Prof. M. Bierlaire
Gael Lederrey & Nikola Obrenovic

Decision Aids
Spring 2018

DATA MINING I - PRACTICE - EXERCISES

We are the 1st December 2012. Mr. Jenkins, a well-known business man, comes to your office because you are an expert in the field of Machine Learning. Mr. Jenkins would like to start investing in the company named Capital Bikeshare. This company is providing Bike sharing systems in Washington D.C.. However, he's not sure if it will be profitable for him or not. After discussing with the business team, he knows that the company earns some money for every hour with more than 80 bikes used. Therefore, he asked some kind Data Scientist to provide him some data[1] . They were able to provide him the data from the 1st of January 2011 up to the 30th of November 2011. The file they provided is named `train_bike_sharing.csv`. It contains 16'637 data points with the following columns (in order):

1. `season`: season (1:springer, 2:summer, 3:fall, 4:winter)
2. `yr`: year (0:2011, 1:2012)
3. `mnth`: month (1 to 12)
4. `hr`: hour (0 to 23)
5. `holiday`: whether a day is a holiday (1) or not (0).
6. `weekday`: day of the week
7. `workingday`: whether this day is neither weekend nor holiday (1) or it falls in these two categories (0).
8. `weathersit`: Weather situation:
   - 1: Clear, Few clouds, Partly cloudy, Partly cloudy
   - 2: Mist + Cloudy, Mist + Broken clouds, Mist + Few clouds, Mist
   - 3: Light Snow, Light Rain + Thunderstorm + Scattered clouds, Light Rain + Scattered clouds
   - 4: Heavy Rain + Ice Pallets + Thunderstorm + Mist, Snow + Fog
9. `temp`: Normalized temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-8, t_max=+39
10. `atemp`: Normalized feeling temperature in Celsius. The values are derived via (t-t_min)/(t_max-t_min), t_min=-16, t_max=+50
11. `hum`: Normalized humidity. The values are divided to 100 (max)
12. `windspeed`: Normalized wind speed. The values are divided to 67 (max)
13. `class`: Whether it's a profitable hour (1) or a non-profitable hour (0).

[1] The original data can be found here: https://archive.ics.uci.edu/ml/datasets/bike+sharing+dataset

In addition, Mr. Jenkins was able to create the data for December 2012. The file he provided is named `test_bike_sharing.csv`. *(You also have the* `class` *column in the test data. However, Mr. Jenkins will know if you cheated!)* In the end, Mr. Jenkins would like to know how many hours will be profitable for the company in December 2012.

Your tasks is to test two different classifiers, Logistic Regression and Decisions Tree, using the functions you coded last week and some functions already implemented in Matlab. You have to find the best features, find a way to make the algorithms better (if possible), and provide good insight about your predictions. But before starting your exercises, Mr. Jenkins has two advice for you:

- *Google is your best friend when you're writing code!*
- *If you understand well these exercises, the Data Mining exercise in the project should not be a problem for you.*

Exercise 1 - Logistic Regression

In this exercise, you will have to use the code you wrote for the Logistic Regression last week. The files `computeCostAndGrad.m`, `sigmoid.m`, and `predict.m` are provided in the folder `functions`. Mr. Jenkins was kind enough to provide you a Matlab file named `logreg.m` where the data is loaded. He also suggests some features you could use: `weathersit`, `temp`, `hum`, and `windspeed`. He also suggest you some metrics he would like to see: The Confusion Matrix and the values of Precision and Recall. Happily, your boss already wrote a Matlab function for the Confusion Matrix that is provided in the file `confusionMatrix.m` in the folder `function`. However, you will have to compute the values of Precision and Recall.

Here are your tasks step-by-step:

1. Go back to last week's exercise set and have a look at the file `exercise1.m`. In this file, you will find how to use Logistic Regression to train a model and predict new labels.

2. In the file `logreg.m`, you have train a model with Logistic Regression.

3. If previous step works flawlessly, you should see the Confusion Matrix. Discuss within your group about the accuracy of this first model (is it good? or bad? and what are the different information provided by the Confusion Matrix). The Wikipedia article on the Confusion Matrix can be very helpful: https://en.wikipedia.org/wiki/Confusion_matrix. In our case, the positive class is *Earning Money* and the negative class is *Not Earning Money*, *i.e.* loosing money.

4. Now, you need to display the values for Precision and Recall. You can implement this in the file named `precision_recall.m` in the folder `function`.

   Hints:

   - *The function* `confusionmat` *from Matlab will provide you the numbers inside the Confusion Matrix.* **WARNING:** *This function requires line vectors. Therefore, make sure you're passing line vectors and not column vectors.*
   - *From this Confusion Matrix, extract the values for True Positive, True Negative, False Positive, and False Negative.*
   - *You can find the definition of Precision and Recall in the Wikipedia article on the Confusion Matrix.*

5. What are the differences between Accuracy, Precision, and Recall? And why are we using these different metrics?

6. (*Optional but interesting*): You can now try different features to see if you can get a better model. For this, you only need to change the column numbers in lines 18-19 in the file `logreg.m`.

Exercise 2 - Decision Tree

In this exercise, you will do the same thing as in Exercise 1 except that you will use a different algorithm: Decision Tree. Mr. Jenkins provided you a file named `decisiontrees.m` in which he's loading the data and suggesting to use the same feature as in Exercise 1.

Here are your tasks step-by-step:

1. You need to train a Decision Tree. Matlab already has a built-in function named `fitctree`. I strongly advise you to have a look at this function on Matlab's website and maybe find some easy-to-understand example on Internet.

2. (*Optional*): Once your tree is trained, you have a look at it by using the function `view(tree, 'mode', 'graph');`. It can be interesting to see how it is built, especially for debugging purpose and feature importance.

3. Now, you need to predict the new data with your tree. Once it's done, you can use the Confusion Matrix and the Precision and Recall values to compare the Decision Tree to the Logistic Regression model you obtained in the previous exercise.

   Hint: I don't give you the name of the function to predict. But a simple Google Search should help you to find it. ;-)

4. (*Optional but interesting*): As for the Logistic Regression, you can try different features and see how your tree performs.

5. One recurrent problem with Decision Trees is that they tend to over-fit the data. Do you have any intuition why? (Having a look at the tree can be useful here).
   Usually, we try to prune the tree to reduce the over-fitting. In order to do that, you need to find the best level of your tree, *i.e.* find the level at which the tree is not over-fitting and gives good results on test data. To get such a level, we can use Cross-Validation. Have a look at the built-in function `cvLoss` in Matlab to find how to get the best level using this function. Then, to prune it, you can use the function `prune` in Matlab. Does the pruning improve your results?

Bonus Exercise - Feature Engineering

This exercise is given as a bonus exercise but this where the fun part begins. Indeed, once you've tested the available features for your model, you may find the best model. And if you're still not satisfied with it, you can now play with the features. What we do here is called *Feature Engineering* because the idea is to create new features based on the original ones you had.

Since training the Logistic Regression is faster than the Decision Tree, I advise you to use it. But you can also do the same thing with the Decision Tree. In the file `feature_engineering.m`, I show you how to add columns to `X_train` and `X_test` on lines 21-22. In this case, I add a column of ones. Indeed, Logistic Regression works sometimes better with the addition of an intercept. (If you're using the Decision Trees algorithm, this feature is useless.)

Now, you can simply be creative and try to come up with the best model possible. Here are your tasks step-by-step:

1. Add your favorite algorithm in the file `feature_engineering.m`. (Simply copy-paste it)

2. Now, you can engineer some new features. Here are some ideas:

   - You can change the **normalization** of your data. For example, you can change the normalization of the humidity such that it's between 0 and 1.

- You can apply some function to your data. For example, you can take the square root of one of the features.
- You can linearly combine two (or more) features. You can, for example, do a multiplication element-wise between two columns.
- etc.

The idea here is to be creative but not excessive. Indeed, it's possible that adding the temperature to the power of 4 will increasing your accuracy by 0.1%. But does it really make sense? (Can you still interpret your features? Is the improvement good enough to keep this new feature? etc.)

3. Be creative and have fun!