CSCI334 Project Description
Machine Learning & Data Mining: Spring 2024
Proposal Due: March 23, 2024

For your course project, you will deepen you exploration of data mining with real-world data. There are four main options:

a. Class Kaggle Data:

   1. Choose one of the following dataset in our class Kaggle competition: CSCI334 Project 2024 competition:

   - Pima Indians Diabetes Data (https://www.kaggle.com/uciml/pima-indians-diabetes-database)
   - Spambase Data Set (https://archive.ics.uci.edu/ml/datasets/Spambase)
   - Labelled Faces in the Wild(LFW) Dataset(https://www.kaggle.com/jessicali9530/lfw-dataset)
   - Boston House dataset (https://www.kaggle.com/datasets/altavish/boston-housing-dataset/data)
   - Titanic: Machine Learning from Disaster (https://www.kaggle.com/c/titanic)

   2. Explore the dataset and related algorithm/code, choose your task then implement it, specify what's your observation, modification/improvement, and present the comparisons of your implementation and those you reference to.

b. Personal Photo Album Mining

   1. Given the personal photo album folder, classify all the faces in the photos. Here are several sample codes:

   - https://realpython.com/blog/python/face-recognition-with-python/
   - https://github.com/bytefish/facerec
   - https://www.youtube.com/watch?v=fRiCOxJtsQQ
   - Apply face recognition functions from sk-learn sample code: https://scikit-learn.org/stable/auto_examples/applications/plot_face_recognition.html

c. External Kaggle: Participate in a different Kaggle competition of your choice. Please note that some competitions have significant data processing components (extracting features from images, etc.) so make sure you can make some progress easily before you commit to one of these. Here are some references:

   - https://www.kaggle.com/code/alexisbcook/getting-started-with-kaggle-competitions
   - https://www.datacamp.com/blog/kaggle-competitions-the-complete-guide

d. If you have a personal project you would like to do instead, please speak to me about it after class; this is also valid but should not overlap with projects for other classes.

Step 1: Individual or Form teams.

You can choose to do by yourself or find a partner who share your interests and with whom you will directly collaborate.

There is no barrier to collaboration on this project, so feel free to talk to other teams about what they are doing, how they are doing it, and how well it works . A significant component of good performance can boil down to feature design and choices, so you may want to talk to other teams about their choices and how it affects performance. You are also free to use online code, or other sources (if so, please indicate in your project proposal and project report) However, please make sure that your own entries are developed by you and your team member. The purpose of this project is to give you practical experience, so it will not be helpful if you just follow instructions from a friend or online resource.

Step 2: Download the data and data preprocessing.

a. Choose the dataset you are interested, download it and import it in a good way that NumPy or your python code can manage.

b. Data exploring on the data: include basic statistics, scatter plot, histogram, probability density distribution, or any visualization if needed, etc.

c. Data preprocessing: checking outlier, missing values, discretization if needed, normalization if needed, feature selection(using PCA, etc.) if needed, etc.

Step 3: Choose a technique (or more) to explore.

Your project will consist of learning several predictors for the Kaggle data (if you choose option a or c), as well as an ensemble "blend" of them, to try to do as well as possible at the prediction task. Specifically, learn *at least three* (more is good) different types of models; suggestions include:

1. **K-Nearest neighbor.** Note that a KNN model on these data will need to overcome two issues: the large number of training & test data, and the data dimension. As noted in class, Distance based methods often do not work well in high dimensions, so you may need to perform some kind of feature selection process to decide how many, and which features to include. Similarly, you will need to reduce the number of training data somehow, either by subsampling, clustering, or selecting "useful" examples to retain. Finally, the right "distance" for prediction may not be Euclidean in the original feature scaling (these are raw numbers); you may want to experiment with scaling/normalizing features differently, or even learning the distance function.
2. **Linear models, linear regression.** linear methods are very fast but may not have enough model complexity to provide a good fit. For such learner, you may need to generate good feature systematically (polynomial, etc.), using clustering, or more.
3. **Kernel methods:** SVMs, libSVM, or RBF kernel. However, like KNN, these methods often do not scale well with dimension (at least for the RBF kernel) or number of training data, and you will need to do something about these issues.
4. **Decision tree, random forest.**
5. **Naïve Bayes classifier.**
6. **Boosted learners.** Use AdaBoost, Gradient Boosting, or another boosting algorithm, to train a boosted ensemble of some base learner (perceptron, decision stump, or Gaussian Bayes classifier).
7. **Neural network**. The key for learning a NN model on these data will be to ensure that your model is well optimized. You should monitor its performance, preferably on both training & validation data, during backpropagation, and verify that the training process is working properly

and converging to a reasonable performance value (e.g., comparably to other methods). Start with few layers and moderate numbers of hidden nodes per layer; within these settings you can work to make sure your model is training adequately.

8. **K mean clustering**
9. **Other.** You tell me: apply another class of learners, or a variant or combination of methods like the above. You can use existing libraries or modify course code; just be sure to understand the model you are applying, and why it may work well.

Step 4: Build your learners.

Use your selected focus, along with the techniques we have developed so far in class, to construct predictive models for your target(s) or discover the underlying structure.
Be aware of the positive and negative aspects of the learners we have discussed. For example, nearest neighbor methods can be very powerful, but can also be very slow for large data sets; a similar statement applies to dual-form SVMs. For such learners, dealing with the large data set may be a significant issue perhaps you could reduce the data in some way without sacrificing performance? On the other hand, linear methods are very fast, but may not have enough model complexity to provide a good fit. For such learners, you may need to try to generate better features, etc.

Step 5: Evaluate

Try every possible model with every possible parameter setting use validation data, or cross-validation, to assess which models are better.

Step 5: Write it up

Your team will produce a single write-up Jupyter Notebook report including all codes and source links (as reference).

The report should describe the problem you chose to tackle and the methods you used to address it, including which model(s) you tried, how you trained them, how you selected any parameters they might require, and how they performed in on the test data. Consider including tables of performance of different approaches, or plots of performance used to perform model selection (i.e., parameters that control complexity).

The report also includes the individual contribution of each team member. For example, who was responsible for which aspects (which learners, etc.), and how the team as a whole puts the ideas together.

Each member must submit the final project (including report, dataset and other python files, package needed) to the GitHub classroom and submit the repo's URL to the Oaks to get credit.

Notes: You are free to collaborate with other teams, including sharing ideas and even code, but please document where your predictions came from. For example, for any code you use, please say in your report who wrote the code and how it was applied (who determined the parameter settings and how, etc.) Collaboration is particularly true for learning ensembles of predictors: your teams may each supply a set of predictors, and then collaborate to learn an ensemble from the set.

Requirements / Grading

I am looking for several elements to be present in any good project. These are:

(a) Exploration of at least one or two techniques on which we did not spend significant time in class. For example, using neural networks, support vector machines, random forests or boosted learner are great ideas; if you do this, explore in some depth the various options available to you for parameterizing the model, controlling complexity, etc. (This should involve more than simply varying a parameter and showing a plot of results.) Other options might include feature design or optimizing your models to deal with special aspects of the data (large numbers of zeros in the data; possible outlier data; etc.). Your report should describe what aspects you chose to focus on.

(b) Performance validation. You should practice good cost function and use validation or cross-validation to assess your models' performance, do model selection, combine models, etc. Try plot ROC and calculate AUC to demonstrate your performance.

(c) Adaptation to under- and over-fitting. Machine learning is not very one size fits all: it is impossible to know for sure what model to choose, what features to give it, or how to set the parameters until you see how it does on the data. Therefore, much of machine learning revolves around assessing performance (e.g., is my poor performance due to under fitting, or over fitting?) and deciding how to modify your techniques in response. Your report should describe how, during your process, you decided how to adapt your models and why.

(d) You report/code can be rerun without errors on the instructor's side.