EE 559 **Course Project Assignment** Post: Fri., 4/7/2023
Jenkins/Chugg Due: Tues., 5/2/2023, 11:59 PM

Introduction

The goal of this project is to develop one (or more) machine learning systems that operate on given real-world dataset(s). In doing so, you will apply tools and techniques that we have covered in class. You may also confront and solve issues that occur in practice and that have not been covered in class. Discussion sessions and past homework problems will provide you with some tips and pointers for this; also internet searches and piazza discussions will likely be helpful.

Projects can be individual (one student working on the project) **or a team** (2 students working together on one project).

You will have significant freedom in designing what you will do for your project. You must cover various topics that are listed below (as "Required Elements"); the methods you use, and the degree of depth you go into each topic, are up to you. And, you are encouraged to do more than just the required elements (or to dive deeper than required on some of the required elements).

Everyone will choose their project topics based on the three topic datasets listed below.

Collaboration and comparing notes on piazza may be helpful, and looking for pertinent information on the internet can be useful. However each student or team is required to do their own work, coding, and write up their own results.

Topic datasets:

There are 3 topic datasets to choose from:

(i) Credit card default
Problem type: classification

(ii) Mushroom
Problem type: classification

(iii) NBA Players
Problem type: regression or classification

These datasets are described in the appendix, below.

Note: for each topic dataset, you are required to use the training and test sets provided on piazza, except where stated otherwise. Some of the features have been preprocessed (e.g., normalization, transformation, deletion, noise removal or addition). Additionally, we want everyone to use the same training set and test set, so that everyone's system can be compared by the same criteria.

Which topic datasets can you use?

Choose any one dataset. For most datasets, you can vary the difficulty level by what problems you define, and by how much depth or extent you pursue. Note the difficulty level estimates given with each dataset description. Part of your project grade will depend on the workload, which includes the amount of work and the difficulty of the problem.

Team projects are expected to do more work than individual projects, so for team projects aim toward a higher overall difficulty level/work load. Individual projects can be more moderate in their goals, but you are still encouraged to pursue directions that you are interested and go beyond the minimal requirements.

Ideas for extending each of the topics, or pursuing more depth, are given in the dataset descriptions.

Computer languages and available code

You must use Python as your primary language for the project. You may use Python and its built-in functions, NumPy, scikit-learn, pandas, and matplotlib. Additionally, you may use imblearn (imbalanced-learn) functions for undersampling or oversampling of your data if that would be useful for your project; and you may use a function or class for RBF network implementation. For neural networks in addition to scikit-learn, you may use keras, tensorflow, or PyTorch. Please note that this is not a class on deep learning systems, so deep neural networks can be tried as a comparison but are not a primary topic of EE 559.

Within these guidelines, you may use any library functions or methods, and you may write your own code, functions, and methods. Please note that for library routines (functions, methods, classes) you use, it is your responsibility to know what the routine does, what its parameters mean, and that you are setting the parameters and options correctly for what you want the routine to do.

If you have parts of your project that uses some techniques outside of EE 559 topics, for those parts you may where necessary use other libraries as appropriate. Please include a readme file that states what libraries need to be imported to run your code, and describe clearly in your report the algorithms you used.

Use of C/C++ is generally discouraged (for your own time commitments and because we didn't cover it in relation to ML). However, there could be some valid reasons to use C/C++ for some portions of your code, e.g. for faster runtime. If you want to use it, we recommend you check with the TAs or instructor first.

Be sure to state in your project report what languages and toolboxes/libraries you used; what you coded yourself specifically for this class project; and of course any code you use from other sources must be credited (referenced) as such.

Required elements

- **The items below give the minimal set of items you are required to include in your project.** Note that you are welcome and encouraged to do more than the minimal required elements (for example, where you are required to use one method, you are welcome to try more than one method and compare the results). Doing more work will increase your workload score, might increase your interpretation score, and might improve your final system's performance.

- **EE 559 content**
  - The majority of your work (> 50%) must use algorithms or methods from EE 559 (covered in any part of the semester).
  - You may also (optionally) try algorithms and methods that were not covered in EE 559 for comparison; and describe the method and results in your report.
- **Consider preprocessing: use if appropriate** [Discussion 7, 8, 9]
  - Tip: you might find it easier to let Python (using pandas) handle csv parsing.
  - Normalization or standardization. It is generally good practice to consider these. It is often beneficial if different features have significantly different ranges of values. Normalization or standardization doesn't have to be all features or none; for example, binary variables are typically not standardized.
  - For classification problems, if the dataset is significantly unbalanced, then some methods for dealing with that should be included. If it's moderately unbalanced, then it might be worth trying some methods to see if they improve the performance. Some approaches to this are done by preprocessing of the data.
  - Representation of categorical (ordinal or cardinal) input data should be considered. Non-binary categorical-valued features usually should be changed to a different representation.
- **Consider feature-space dimensionality adjustment: use if appropriate**
  - You can use a method to try reducing and/or expanding the dimensionality, and to choose a good dimensionality. Use the d.o.f. and constraints as an initial guide on what range of dimensionality to try.
  - In addition to feature-reduction methods we covered in EE 559, feel free to try others.
- **Cross validation or validation**
  - Generally it's best to use cross validation for choosing parameter values, comparing different models or classifiers, and/or for dimensionality adjustment. If you have lots of data and you're limited by computation time, you might instead use validation without cross-validation.
- **Training and prediction**
  - Individual projects should try at least 3 different classification/regression techniques that we have covered (or will cover) in class; it is recommended for these 3 techniques to cover 3 of these 4 types: non-probabilistic (distribution-free), support vector (classification or regression), neural network, and probabilistic (statistical). Team projects should cover at least 4 classification/regression techniques, preferably one from each of the 4 types listed, if they all fit the problem type you are solving. Beyond this, feel free to optionally try other methods.

    Note that the required trivial and baseline systems don't count toward the 3 or 4 required classifiers or regressors, unless substantial additional work is done to optimize it to make it one of your chosen systems.

- **Proper dataset (and subset) usage**
  - Final test set (as given), training set, validation sets, cross validation.
- **Interpretation and analysis**
  - Explain the reasoning behind your approach. For example, how did you decide whether to do normalization and standardization? And if you did use it, what difference did it make in system performance? Can you explain why?
  - Analyze and interpret intermediate results and final results. Especially, if some results seem surprising or weren't what you expected. Can you explain (or hypothesize reasons for) what you observe?
  - (Optional) If you hypothesize a reason, what could you run to verify or refute the hypothesis? Try running it and see.
  - (Optional) Consider how you would change or amend your approach if you had more data or less data. Or, what else could be done to potentially improve the performance. Suggest this in your report and justify why it could be helpful.
- **Reference systems and comparison**
  - At least one trivial system and one baseline system are required. Each dataset description states what to use or what is recommended for these systems.
  - Run, and then compare with, the baseline system(s). The goal is to see how much your systems can improve over the baseline's system performance.
  - Also run, and compare with, the trivial system. The trivial system doesn't look at the input values $\underline{x}$ for each prediction; its comparison helps you assess whether your systems have learned anything at all.
- **Performance evaluation**
  - Report on the cross-validation (or validation) performance (mean and standard deviation); it is recommended to use one or more of the required performance measures stated in your dataset's description (in the Appendix, below). If you use other measure(s), justify in your report why.
  - Report the test-set performance of your final (best-model) system.
    - For your final-system test-set performance, you may use the best parameters found from model selection, and re-train your final system using all the training data to get the final weight vector(s).
    - Report on all required performance measures listed in the dataset description.
    - If you also tried models outside the scope of EE 559, include you best performing EE 559 system here, as described above. If an outside-of-EE 559 system was your best performing system overall, then also report on its performance. Clearly indicate which results are from which system.
  - You may also report other measures if you like. Use appropriate measures for your dataset and problem.

- **Written final report and code**
  - Submit by uploading your report and code in the formats specified below, to D2L.

General Tips

1. Be careful to keep your final test set uncorrupted, by using it only to evaluate performance of your final system(s).
2. If you find the computation time too long using the provided dataset(s), you could try the following: check that you are using the routines and code efficiently, consider using other classifiers or regressors, or down-sample the training dataset further to use a smaller *N* for your most repetitive work. In the latter case, once you have narrowed your options down, you can do some final choices or just your final training using the full (not down-sampled) training dataset.
3. Wherever possible, it can be helpful to consider the degrees of freedom, and number of constraints, as discussed in class. However, this is easier to evaluate for some classifiers than for others; and yet for others, such as SVM and SVR, it doesn't directly apply.

Grading criteria

Please note that your project will not be graded like a homework assignment. There is no set of problems with pre-defined completion points, and for many aspects of your project there is no one correct answer. The project is open-ended, and the work you do is largely up to you. You will be graded on the following aspects:

- Workload and difficulty of the problem (effort in comparison with required elements, additional work beyond the required minimum, progress in comparison with difficulty of the problem);
- Approach (soundness and quality of work);
- Performance of final system on the provided test set;
- Analysis (understanding and interpretation);
- Final report write-up (clarity, completeness, conciseness).

In each category above, you will get a score, and the weighted sum of scores will be your total project score.

For team projects, both members of a team will usually get the same score. Exceptions are when the quantity and quality of each team member's effort are clearly different.

Final report [detailed guidelines (and template) to be posted later]

In your final report you will describe the work that you have done, including the problem statement, your approach, your results, and your interpretation and understanding.

Note that where you describe the work of others, or include information taken from elsewhere, it must be cited and referenced as such; similarly, any code that is taken from elsewhere must be cited in comments in your code file. Instructions for citing and referencing will be included with the final report instructions. Plagiarism (copying information from elsewhere without crediting the source) of text, figures, or code will cause substantial penalty.

For team projects, each team will submit one final report; the final report must also describe which tasks were done by each team member.

You will submit one pdf of your (or your team's) report, one pdf of all code, and a zip file with all your code as you run it. .

Please note that your **report** should include all relevant information that will allow us to understand what you did and evaluate your work. Information that is in your code but not in your report might be missed. The purpose of turning in your code is so that we can check various other things (e.g., random checks for proper dataset usage, checking for plagiarism, verifying what you coded yourself (as stated in the report), and running the code when needed to check your reported results).

Appendix: Overview of Project Datasets

For all datasets

Use only the provided training and test sets on D2L, except where stated otherwise. In the provided datasets, some of the features may have been preprocessed, and thus are incompatible with the feature representation of the dataset that is posted on the UCI website. Except where stated otherwise, use only the provided (D2L) datasets for your course project work, so that everyone on a given topic is graded on the same datasets.

1. Default of credit card clients dataset

Description written by: *Pranav.*

Summary description

This dataset contains credit card payment information of six months (April 2005 – September 2005) of 30,000 customers from Taiwan. The goal is to predict whether the customer defaults on payment or not. This is a binary classifier task.

Problem type: Classification.

Data Description

The dataset contains 23 features. They are:

1. Amount of the given credit
2. Gender
3. Education
4. Marital status
5. Age
6. Repayment status in September 2005
7. Repayment status in August 2005
8. Repayment status in July 2005
9. Repayment status in June 2005
10. Repayment status in May 2005
11. Repayment status in April 2005
12. Amount of bill statement in September 2005
13. Amount of bill statement in August 2005
14. Amount of bill statement in July 2005
15. Amount of bill statement in June 2005
16. Amount of bill statement in May 2005
17. Amount of bill statement in April 2005
18. Amount paid in September 2005

19. Amount paid in August 2005
20. Amount paid in July 2005
21. Amount paid in June 2005
22. Amount paid in May 2005
23. Amount paid in April 2005

Refer to the UCI website [1] for more details regarding these features. The output classes are: 0 (NO default payment) and 1 (default payment).

Note: Please use the dataset csv files provided by us (and not the one on the UCI repository). It is the same as the one on the UCI repository, except for the train/test split and we have removed the customer ID column. We have already split the dataset into train and test sets for you.

Required performance measures to report.

- F1-score
- Accuracy
- Confusion matrix

Required reference systems and analysis.

You must code, evaluate and report the test performance of the following reference systems.

- **Trivial system:** A system that outputs class assignments (S_0, S_1) at random with probability N_0/N and N_1/N, respectively; N_i is the number of training data points with class label S_i, and N is the total number of training data points.
- **Baseline system:** Nearest means classifier.

In addition to the requirements given in the Project Assignment, you are encouraged to discuss which features are more important over the other, and any patterns (such as if it is more likely that married individuals default more often than unmarried individuals). For group projects, you are expected to show results with and without compensating for class imbalance (at least one approach).

Tips

Following are general guidelines to consider while working with this dataset. They may or may not necessarily improve the model performance.

Class imbalance

The dataset is quite imbalanced with more than 75% of the data points belonging to class 0 (NO default). A variety of approaches can be used to accommodate this: subsample the majority class; augment the minority class; or use weighted loss function or weighted data points (for some classifiers). These approaches will be covered in discussion session.

As an example, if you augment class 1 to make the dataset more balanced, then keep the following in mind. Be careful when dealing with categorical features such as gender, education, and marital status. The augmented data points shall not have values of these features different from the ones in the original dataset (while this is not a hard constraint, doing so may generate significant outliers and affect the classification performance). Also, be careful that these approaches, if applied, are used only on the training dataset and the test set should not be altered (even if it is imbalanced).

Data cleaning

If you do a closer data inspection, you will find that some categorical features have more categories than the description in [1]. For example, for the ***education*** feature, there are four categories as per [1] – graduate school (1), university (2), high school (3), and others (4). However, in the given dataset, you will notice that some data points have values 0, 5, and 6 for ***education.*** Since we do not have information as to what these undefined categories mean, you may do two things. One way is to keep these features as they are

. Other option is to reassign them to some of the known categories. Intuitively, it makes sense to assign all the new categories to category 4 (others). However, you may assign them to other categories if you see some pattern with respect to other features for these data points. Again, all the analysis needs to be performed only using the training dataset. Then, the established rule should be applied to modify these features in the test set. The same is true for ***marital status***. Be sure to include in your report, whether any data cleaning method was adopted and what was the motivation behind selecting a particular approach.

You will also notice that some features do not exactly match with the description on the UCI webpage [1]. For example, for features 6–11, as per [1], the features range from -1 (pay duly) to 9 (payment delay for 9 months and above). However, if you compare it with the same feature values from the given csv files, you will notice that the range for these features is -2 to 8. Note that the original dataset file from [1] also has this discrepancy. Please keep it as it is (and just assume in mind that they are offset by 1, meaning -2 in the given dataset means paid duly, while 8 means payment delay for 9 months and above).

Feature transformation / selection

As a general feature engineering process, you may use several techniques discussed in the class such as feature normalization, non-linear feature transformation and feature selection, wherever appropriate.

One idea for doing more than required for this dataset (and possible consideration for a group project) is to study the effect of number of months on the classification. In particular, you may only use the data for three (instead of six) months – July, August, and September 2005. Can your model still be reliable and achieve similar results as before? Is any machine learning algorithm more sensitive to number of months over the other? If so, why? It may happen that the performance drops with each deleted month for all the algorithms you use. This is fine. Even if this is the case, show all your results and summarize your observations.

Estimated difficulty level (1-5, 1 is easy, 5 is difficult):

3-4 (moderate to challenging)

Reference

[1] https://archive-beta.ics.uci.edu/dataset/350/default+of+credit+card+clients

2. **Mushroom dataset**

Description written by: *Thanos.*

Summary description

This dataset includes 61069 mushrooms of 173 different species (353 mushrooms per species). The dataset does not provide information regarding the species of each data point (mushroom). Each mushroom is identified as edible or poisonous.

Problem type: Classification.

Data Description

The dataset contains 15 features. They are:

1. cap-diameter
2. cap-shape
3. cap-surface
4. cap-color
5. does-bruise-bleed
6. gill-attachment
7. gill-spacing
8. gill-color
9. stem-height
10. stem-width
11. stem-color
12. has-ring
13. ring-type
14. habitat
15. season

24.

25. Refer to the UCI website [1] for more details regarding these features. The output classes are: p (poisonous) and e (edible).

Note: Please use the information on the UCI website as reference only. Use the dataset csv files provided by us (and not the one on the UCI repository as we have performed some required preprocessing such as dealing with missing values, etc.). We have already split the dataset into train and test sets for you.

Required performance measures to report.

- F1-score
- Accuracy

Required reference systems and analysis.

You must code, evaluate and report the test performance of the following reference systems.

- **Trivial system:** A system that outputs class assignments (S_0, S_1) at random with probability N_0/N and N_1/N, respectively; N_i is the number of training data points with class label S_i, and N is the total number of training data points.
- **Baseline system:** Nearest means classifier.

You should be expecting to see improved performance of even simple models compared to the trivial and baseline systems. You are further encouraged to perform some feature importance analysis (such as which features are important on the predictive task) and analyze potential data patterns (such as red colored cap mushrooms with bell cap-shape and fibrous cap surface tend to be poisonous. Refer to Feature Engineering below for more explanation and examples.

Tips

Following are general guidelines to consider while working with this dataset. They may or may not necessarily improve the model performance.

Feature Engineering

Most of the features of the dataset are categorical (such as cap-shape: bell, conical, convex or flat) and cannot be directly used as inputs to machine learning models (since they are not numerical). We can use such features to create extra features on both training and test datasets. The new features can reflect statistics of the original numerical features and can potentially detect patterns of poisonous or edible mushrooms and simplify the classification task.

We can use the categorical features to group all the data points with the same categorical feature value (i.e., all the mushrooms with orange cap color) and calculate statistics of the numerical data corresponding to each group (i.e., average cap-diameter of all the mushrooms with orange cap color). Then in the new feature, all data points of this group (i.e., mushrooms with orange cap-color) are assigned that calculated statistic. This could be used as an alternative to one-hot encoding of the feature.

In general, we can calculate features such as: Average/min/max/median cap-diameter/stem-height/stem-width of mushrooms of cap-shape 'x'/'f'/'p'/'b'/'c'/'s'/'o'. In this case you will create 4*3*6 = 72 new features. You can further combine more categorical features such as minimum stem-height of mushrooms of cap-shape /'p'/'b'/ and stem-color /'y'/'b'/.

The following command calculated the average (mean) of the numerical data only by grouping the mushrooms by cap-color (df is the original dataset and mean_df is the dataset of the means per group):

```
mean_df = df.groupby(['cap-color'], as_index=True).mean(numeric_only =
True)
```

Feature Selection or Reduction

After the feature engineering step, the dataset will have a large number of features, which can slow down model runtime or possibly worsen model performance. It is important to remove some features (or reduce dimensionality of feature space some other way) and keep $\boldsymbol{D'} < \boldsymbol{D}$ number of features that are more useful for the prediction task:

1. A simple way of doing this is to measure each feature's correlation (such as with Pearson correlation coefficient) with the class label and then use the D' number of features with the highest correlation.
2. Another way of doing this is using a simple model (such as a linear model) to predict poisonous or edible mushrooms using one feature at a time and then use the D' number of features with the highest performance.
3. Refer to the Lectures 19 – 20 (either section) and Discussion 12 notes for other methods for feature selection and dimensionality reduction.

Estimated difficulty level (1-5, 1 is easy, 5 is difficult):

- 2 for classification task with one hot encoding for converting categorical data to numerical.
- 3-4 for classification task with more extensive feature engineering methods.

3. NBA Players Dataset

Description written by: *Chengwei Wei*

Description

The NBA Players Dataset contains data on each player who has been part of an NBA team's roster over the past two decades. The dataset captures demographic variables such as age, height, weight, and place of birth, as well as biographical details such as the team played for, draft year and round. In addition, it includes basic box score statistics such as games played, average number of points, rebounds, assists, etc. You have freedom to explore the dataset and design the machine learning tasks you are interested in.

Suggested project tasks:

1. **Classification or Regression tasks:** Create or choose one or several classification or regression tasks on this dataset.
   Regression Tasks:
   - Predict players' pts, rebs, and ast (average number of points, rebounds, and assists) this season based on the statistics in previous seasons.
   - Given players' height or weight or other demographic attributes, predict their performance (average pts, rebs, and ast of all seasons).

   Classification Tasks:
   - Draft round and draft number prediction. (For draft number, you may want to combine into several categories, like 1-5 as class "very early", 6-10 as class "early", etc. Or, pose the draft number prediction as a regression problem.)
2. **Data Exploration:** In addition to a classification or regression task, you could explore the dataset by clustering data points and interpreting each cluster. For example, you could average the statistics of all seasons for a player and use the average statistics to cluster

players into different groups. Then, try to identify the similar attributes (such as height or rebounds number) of players in the same group.

Data Description

The full dataset [1] is posted on Kaggle. In detail you have:

1. player_name: Name of the player
2. team_abbreviation: Abbreviated name of the team the player played for (at the end of the season)
3. age: Age of the player
4. player_height: Height of the player (in centimeters)
5. player_weight: Weight of the player (in kilograms)
6. college: Name of the college the player attended
7. country: Name of the country the player was born in (not necessarily the nationality)
8. draft_year: The year the player was drafted
9. draft_round: The draft round the player was picked
10. draft_number: The number at which the player was picked in his draft round
11. gp: Games played throughout the season
12. pts: Average number of points scored
13. rebs: Average number of rebounds grabbed
14. ast: Average number of assists distributed
15. net_rating: Team's point differential per 100 possessions while the player is on the court
16. oreb_pct: Percentage of available offensive rebounds the player grabbed while he was on the floor
17. dreb_pct: Percentage of available defensive rebounds the player grabbed while he was on the floor
18. usg_pct : Percentage of team plays used by the player while he was on the floor
19. ts_pct: Measure of the player's shooting efficiency that takes into account free throws, 2 and 3 point shots
20. ast_pct: Percentage of teammate field goals the player assisted while he was on the floor
    season NBA season

Required training and testing sets

There is no train-test-split in the original dataset.

We will provide train/test splits in 2 ways:

- Training and testing sets split 80/20 random but stratified according to player (training set has 80% of stats for each player; testing set has 20%).
- Testing set consists of all data points from the 2 most recent seasons; training set consists of data points from all previous seasons back to 1996-1997 season.

If either of these splits fit the problem you have defined, then use the appropriate training and testing sets we have provided for that problem. If neither fits the problem you have defined, then create your own training and testing datasets appropriate to that problem; one training set (for you to use as you like for training, validation, cross-validation, etc.); one test set to use on your final model, to estimate its performance on unknowns. Be sure to state in your report how you have created your training and testing sets.

Suggested Reference Systems

You can code, evaluate and report the test performance of the following reference systems. Both trivial and baseline systems are required; please use the suggested systems below if they are appropriate to the problem(s) you have defined; if they are not, then define your own reference systems and explain the rationale in your report.

For Regression

- Trivial System:
  - A system that always outputs the mean output value $\bar{y}$ from the training set.
- Baseline systems:
  - 1NN
  - Linear Regression (no regularization)

For Classification

- Trivial System:
  - A system that randomly outputs class labels with probability based on class priors (priors calculated from the training set). Run the trivial system at least 10 times and take the average of the results as the final performance of the trivial system.
- Baseline system:
  - Nearest Means

Suggested performance measures to report

For Regression

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared (R^2)

For Classification:

- Accuracy
- Macro f1-score*
- Confusion matrix

*Comment: for multiclass problems, you will have a precision, recall, and f1 score for each of the classes. Macro f1-score is the mean of f1 score over all classes.

Overall Tips (apply to both classification and regression)

(i) Consider converting the data into a more usable form, e.g. non-binary categorical variables are best converted to binary one-hot variables.

(ii) For the systems you choose, design, and optimize, you may need to do data cleaning, and try feature normalization, nonlinear transformation, feature selection, and other data preprocessing techniques and data engineering techniques that you have learned in EE559.

Estimated difficulty level (1-5, 1 is easy, 5 is difficult)

Regression:

1) Predict players' performance (e.g. pts, rebs, and ast (average number of points, rebounds, and assists)) this season based on the statistics in previous seasons. (4-5)
2) Predict players' performance given their height, weight, age, and/or other demographic attributes. (3-4)

Classification:

1) Draft round and draft number prediction (3)

References

[1] Dataset information: https://www.kaggle.com/datasets/justinas/nba-players-data