EE 559 **Course Project Assignment** Post: 3/29/2024
Jenkins Due: 4/26/2024, 11:59 PM

Introduction

The goal of this project is to develop machine learning systems that operate on your choice of the 3 given datasets. 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.

We also hope that you find this project interesting, invigorating, and/or motivational.

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 3 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) Dry beans dataset
Problem type: classification

(ii) Infrared thermography dataset
Problem type: regression

(iii) Laptop dataset
Problem type: regression

These datasets are described in the appendix, below.

Note: for each topic dataset, you are required to use the training and test sets posted on D2L with this assignment, except where stated otherwise. We have done some work on some of these datasets to help you have a good project experience. Some of the features may have been preprocessed (e.g., normalization, transformation, feature removal or creation). Additionally, we want everyone to use the same training set and test set, so that everyone's system on a given topic can be compared by the same criteria.

Which topic datasets can you use?

Individual projects: choose any one dataset. Team projects: choose either the thermography dataset or the laptop dataset. Note the difficulty level estimates given with each dataset description. For most datasets, you can vary the difficulty level by what problems you define, and by how much depth or extent you pursue. 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 interest you, and to 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, SciPy, scikit-learn, pandas, matplotlib, and Seaborn. 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 optionally be tried as one of your ML models, but should not be the primary focus of your project.

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 for more than just a small portion of your code, please check with a TA or the instructor first.

Be sure to state in your project report what languages and toolboxes/libraries you used, and what you coded yourself specifically for this class project. Coding algorithms yourself will count towards higher workload, but too much self-coding might hinder the amount of time you have to spend on other parts of your project. Also, 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; if so, be sure to describe the method in your report.
- **Consider preprocessing: use if appropriate** [Discussion 10]
  - Tip: you might find it easier to let Python (using pandas) handle csv parsing.
  - Normalization or standardization. This 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.
  - If there are any missing entries to the data, a technique to resolve this must be used.
- **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 hyperparameter 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 ML models (classification/regression techniques); it is recommended that these 3 ML models cover 3 of these 4 types: non-probabilistic (distribution-free), support vector (classification or regression), neural network, and probabilistic (statistical). Team projects should try at least 4 ML models (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 ML models, 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 your baseline system(s) first, and then compare your chosen ML models when you run them. 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); use at least the required performance measures stated in your dataset's description (in the Appendix, below).
  - 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, and justify in your report why. 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 (e.g., .py, .ipynb, files).

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., checks for proper dataset usage where warranted, checking for plagiarism, verifying what you coded yourself (as stated in the report), and running the code when needed to check your reported results).

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.

Each dataset has a range for its "estimated difficulty level". The lower number represents basic work to get reasonable results, but not much extra. The upper number represents significant amount of extra work such as trying more feature engineering, more sophisticated techniques for preprocessing or feature reduction, or diving into more detail about the dataset or application area to better tailor your approaches, etc. Datasets that have a larger range have more opportunity to add things for a higher difficulty or workload.

Dry Bean Classification dataset

Description written by: *Jiageng Zhu.*

Summary description

This dataset includes 13611 grains from 7 different classes of dry beans.

Problem type: Classification.

Data Description [1]

The dataset contains 16 features. They are:

Area
Perimeter
MajorAxisLength
MinorAxisLength
AspectRation
Eccentricity
ConvexArea
EquivDiameter
Extent
Solidity
Roundness
Compactness
ShapeFactor1
ShapeFactor2
ShapeFactor3

ShapeFactor4

Required performance measures to report.

- Classification accuracy
- F1-score
    - Macro-averaged F1 score
    - Micro-averaged F1 score
- Confusion matrix

Required reference systems and analysis.

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

- **Trivial Solution:** A system that outputs class assignments (S_0, $S_{1...}$ S_6) at random with probability N_0/N and N_1/N, etc.; 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).

Tips

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

Data Balance

The dataset is not a balanced dataset, i.e., the number of samples of different classes are not the same. Because of that, model performance may be compromised. To mitigate this issue, you can try:

1. Remove some samples from classes whose number is larger than others
2. Synthesize some additional minority-class samples (for example, Synthetic Minority Over-sampling Technique)
3. Refer to Discussion 10 for other suggestions

Data Augmentation

Adding noise to data can help to model robustness, simulating the effect of small variations that could occur in real-world data. This can be as simple as adding a small random value to each feature in your dataset, ensuring the noise is in a realistic range that does not change the meaning of the data.

Remove Outliers

Removing outliers in training data is often considered a necessary step in data preprocessing for several reasons. These outliers can be due to variability in the measurement or experimental errors. In some cases, they might be genuine observations that are significantly different from the rest of the data.

Data Preprocessing and Feature Engineering

The scale of each feature is different. Therefore, consider normalizing the features before training the model.

For additional work: You might consider using nonlinear combinations of features or using a nonlinear transformation to improve performance. You could follow this up with dimensionality reduction to see which feature combinations are most pertinent for prediction.

Feature Selection or Reduction

After the feature engineering step, the dataset might have a large number $\boldsymbol{D'}$ of features, which can slow down model runtime or possibly worsen model performance. It might be helpful to remove some features (or reduce dimensionality of feature space some other way) and keep $\boldsymbol{D'' < D'}$ 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 $\boldsymbol{D''}$ 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 the output class using one feature at a time and then use the $\boldsymbol{D''}$ number of features with the highest performance.
3. Other techniques (better performance but more computation than 1 and 2 above) include sequential forward (and sequential backward) feature selection.
4. Refer to EE 559 lectures for other dimensionality reduction techniques.

Estimated difficulty level (1-5, 1 is easy, 3 is moderate, 5 is difficult):

3 - 3.5.

For example, the higher difficulty level could include more feature engineering and data augmentation; and/or better efforts to resolve data imbalance.

Reference

[1] https://www.kaggle.com/datasets/nimapourmoradi/dry-bean-dataset-classification/data

Infrared Thermography Temperature dataset

Description written by: *Jintang Xue.*

Summary description

The Infrared Thermography Temperature Dataset contains temperatures read from various locations of infrared images about patients, with the addition of oral temperatures measured for each individual. The 33 features consist of gender, age, ethnicity, ambient temperature, humidity, distance, and other temperature readings from the thermal images. The dataset is intended to be used in a regression task to predict the oral temperature using the environment information as well as the thermal image readings.

Problem type: Regression.

Data Description

The dataset contains 1020 data points and 33 features. The features are:

1. T_offset_1
2. Max1R13_1
3. Max1L13_1
4. aveAllR13_1
5. aveAllL13_1
6. T_RC_1
7. T_RC_Dry_1
8. T_RC_Wet_1
9. T_RC_Max_1
10. T_LC_1
11. T_LC_Dry_1
12. T_LC_Wet_1
13. T_LC_Max_1
14. RCC_1
15. LCC_1
16. canthiMax_1
17. canthi4Max_1
18. T_FHCC_1
19. T_FHRC_1
20. T_FHLC_1
21. T_FHBC_1
22. T_FHTC_1
23. T_FH_Max_1
24. T_FHC_Max_1
25. T_Max_1
26. T_OR_1
27. T_OR_Max_1

28. Gender
29. Age
30. Ethnicity
31. T_atm
32. Humidity
33. Distance

Refer to the UCI website [1] for more details regarding these features. You may also check the two related papers for more information about the features [2, 3]. The output target is oral temperature measured in monitor mode (aveOralM). Please ignore the aveOralF in the original dataset, as it is a second output quantity which is a less precise measurement of the oral temperature so we will ignore it.

Note: Please use the dataset .csv files provided by us (and not the one on the UCI repository). It is the same as the corresponding one on the UCI, except we have removed the aveOralF, Cosmetics, time, and date columns as they are not needed in the project. And we have already split the dataset into train and test sets for you. In this project, we only focus on the FLIR data for both groups 1 and 2.

Required performance measures to report.

- Mean Absolute Error (MAE)
- Mean Squared Error (MSE)
- Root Mean Squared Error (RMSE)

Required reference systems and analysis.

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

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

In addition to the requirements given in the Project Assignment, you are encouraged to discuss which features are more important than the others and any patterns (such as if the forehead temperature is more important than the canthus temperature). For group projects, you are expected to explore more background information based on the provided papers and develop more advanced approaches to deal with the data (at least one approach).

Tips

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

Preprocessing and Feature Engineering

Most of the features of the dataset are numerical (such as the temperatures) and can be directly used as inputs to machine learning models. Note that there are 4 rounds for each temperature measured. You need to apply proper ways to preprocess the data (such as averaging the 4 rounds). You may also use such features to create extra features on both training and test datasets. There are also some text features in the dataset (such as gender, age, ethnicity, etc.). You may need to convert them into numerical values in proper ways (such as one-hot coding).

You can also use the categorical features to group all the data points with the same categorical feature value (i.e., all the female individuals) and calculate statistics of the numerical data corresponding to each group (i.e., the average/min/max/median of forehead center temperature of all the female individuals). Then in the new feature, all data points of this group are assigned that calculated statistic. This could be used as an alternative to the one-hot encoding of the feature. You might also consider the advantages and disadvantages of this method vs. one-hot encoding.

There are some missing values in the dataset. Please make sure to check missing values in the pre-processing stage and use proper ways to handle them.

Feature Selection or Reduction

After the feature engineering step, the dataset might have a large number $\boldsymbol{D'}$ of features, which can slow down model runtime or possibly worsen model performance. It might be helpful to remove some features (or reduce dimensionality of feature space some other way) and keep $\boldsymbol{D'' < D'}$ 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 known output $y(x_n)$ and then use the $\boldsymbol{D''}$ 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 the output class using one feature at a time and then use the $\boldsymbol{D''}$ number of features with the highest performance.
3. Other techniques (better performance but more computation than 1 and 2 above) include sequential forward and sequential backward feature selection.
4. Refer to EE 559 lectures for other dimensionality reduction techniques.

Estimated difficulty level (1-5, 1 is easy, 3 is moderate, 5 is difficult):

3.25 – 4.5.

Reference

[1] https://archive.ics.uci.edu/dataset/925/infrared+thermography+temperature+dataset

[2] https://physionet.org/content/face-oral-temp-data/1.0.0/

[3] https://www.mdpi.com/1424-8220/22/1/215

Laptop Prices dataset

Description written by: *Sagar Sudhakara and Shipeng Liu*

Summary description

Predict laptop prices based on their specifications. This laptop dataset is a detailed collection that looks at different features of laptops like screen size, CPU, RAM, storage, graphics card, operating system, weight, and price. It helps analyze how these specs affect laptop prices across various brands and models. The dataset is a great resource for understanding how the details of a laptop, from its screen to its internal components, can influence how much it costs. By examining how these factors relate to price, users of this dataset can get important insights into what makes laptops more or less expensive, helping them make informed decisions or analyses about these devices.

Problem type: Regression.

Data Description

The dataset contains 1300 data points and 9 features. The features are:

- Company: Brand Name
- TypeName: Laptop Category
- Inches: Screen Size
- Screen Resolution: Screen Resolution
- Cpu: Central Processing Unit
- Ram: Random Access Memory
- Memory: Storage Capacity
- Gpu: Graphics Processing Unit
- OpSys: Operating System

Refer to the Kaggle website [1] for more details regarding these features. The output values are the price of laptop.

Note: You are required to use the dataset csv files provided by us (and not the one on the Kaggle repository).

Required Reference Systems

Code, evaluate and report the test performance of the following reference systems. Both trivial and baseline systems are required.

Trivial system

- A system that always outputs the mean output value $\bar{y}$ from the training set.

Baseline systems

- 1NN
- Linear Regression (no regularization)

Suggested performance measures to report.

- Root Mean Squared Error (RMSE)
- Mean Absolute Error (MAE)
- R-squared (R2)

Tips for preprocessing and feature engineering

(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) Some of the features contain a variety of text. You will need to convert the data to numerical form to make it usable. For example, if there are a only a moderate number of different text entries for a feature, then you could treat the different entries as categories. Then a standard techniques for converting categorical features can be used.

(iii) Some of the features might be split into multiple features using one-hot encoding combined with other methods. For example, the CPU feature might be represented using one-hot for CPU type (e.g., one feature for "Intel Core i3"), but the value of the hot feature could be a numeric attribute like processor speed (instead of just a value of 1).

(iv) 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.

(v) See Discussion 10 for more information on feature preprocessing.

Estimated difficulty level (1-5, 1 is easy, 3 is moderate, 5 is difficult)

3.5 – 4.25.

Reference

[1] https://www.kaggle.com/datasets/ara001/laptop-prices-based-on-its-specifications/data