CSC 503/SENG 474: Assignment 1

Due on: Sunday, Jun 6th at 23:59 PST
Where: Brightspace (`https://bright.uvic.ca/d2l/home/136102`)

Instructions:

- You must complete this assignment *entirely* on your own. In other words, you should come up with the solution *yourself*, write the code *yourself,* conduct the experiments *yourself*, analyze the results *yourself*, and finally, write it all solely by *yourself.* The university policies on academic dishonesty (a.k.a. cheating) will be taken very seriously.

- This does not mean that you need to go to a cave and self-isolate while preparing the assignment. You are allowed to have high-level discussions with your classmates about the course material. You are also more than welcome to use Piazza or come to office hours and ask questions. If in doubt, ask!— we are here to help.

- If you are still stuck, you can use books and *published* online material (i.e., material that has a fixed URL). However, **you must *explicitly* credit all sources**. **You are also not allowed to copy-paste online materials**. Woe to you if we catch you copy-pasting the uncredited sources!

  - Why "if stuck"? Assignments are designed to develop your practical ML skills and make you strong. If you do the assignments well, the project will feel like a piece of cake. So, give your best. But, on the other hand, do not waste a whole week on a single question: if you are stuck on a question for a few days, ask (us) for help!

- If you cannot make it until the deadline, you can use **a maximum of two grace days** per assignment. They are not free, though: **each grace day comes with the 25% mark penalty** (so submitting on Monday evening would reduce your score by 25%; submitting on Tuesday would further reduce it by 50%). No other accommodations will be provided unless explicitly approved by the instructor at least 7 days before the deadline.

- **These assignments are supposed to be really hard! Start early!** You will need *at least* two weeks to complete them!

  - If you do not feel challenged enough, please let me know, and I'll think of something.

- Remember: **you will need to gather at least one-third of all points during the assignments to pass the course. If you don't, you will get an F!**

- Make sure to follow the technical requirements outlined below. TAs have the full power to take 50% off your grade if you disregard some of them.

- Be sure that your answers are clear and easy for TAs to understand. They can penalize you if your solutions lack clarity or are convoluted (in a non-algebraic way), even if they are nominally correct.

- We will try to grade your assignments within seven (7) days of the initial submission deadline.

- If you think there is a problem with your grade, you have one week to raise concern after the grades go public. Grading TAs will be holding office hours during those seven days to address any such problems. After that, your grade is set in stone.

Technical matters:

- You must type up your analysis and solutions *electronically* and submit them as a self-containing Jupyter notebook. Jupyter notebooks can contain code, its output, and images. They can also be used to type math and proofs in LaTeX mode.
  - You ***must*** use LaTeX mode to type formulas. Typing `a^2=sqrt(3)+b1` is a pretty good way to lose 50% of your grade for no good reason.
- Each problem should be submitted as a separate file.
- Each file should be named `SurnameInitial_N.ipynb`, where `N` is two digit-padded problem number. **Correct:** `SmithJ_05.ipynb`. **Incorrect:** `JohnSmith_V12345 Problem 1.ipynb`, `prob1.pdf` etc.
- Zip all `ipynb` files and submit them as `assignment1.zip` to the Brightspace. Do not submit RAR, TAR, 7zip, SHAR and whatnot; just use good ol' ZIP. Do not include other files.
- The first cell of each Jupyter notebook must start with your name and V number. See the attached notebook for the details.
- Your notebook should be organized sequentially according to the problem statement. Use sections (with the appropriate numbers and labels) within the notebook. Figures and relevant code should be placed in the proper location in the document.
- Notebook code ***must be runnable***! Ideally, all answers will be the output of a code cell.
- You must use **Python 3** to complete the assignments. Feel free to use NumPy and `pandas` as you find it fit. Use SciPy, `scikit-learn`, and other non-standard libraries only when explicitly allowed to do so.
- Your first executable cell should set the random seed to 1337 to ensure the reproducibility of your results. For Numpy/SciPy and `pandas`, use `numpy.random.seed(1337)`; otherwise, use `random.seed(1337)`.
- Document your code! Use either Markdown cells or Python comments to let us know what you have done!
- Finally, **be concise**! We do not appreciate long essays that amount to basically nothing.

This assignment consists of 11 problems. Some are intended only for graduate students (those taking CSC 503), and are labelled as such. Some contain bonus sections: you can use bonus points to improve your overall homework score. Bonus points cannot be transferred to other assignments or the final project. Any graduate-level problem counts as a bonus problem for undergraduate students. Some problems are interconnected: you cannot solve Problem 2 without solving Problem 1.

Some problems are purposefully open-ended. Whatever you think a correct answer is, make sure to support it with code and data.

If all this feels dull, read this for some motivation (credits to M. Schmidt): `https://www.quora.com/Why-should-one-learn-machine-learning-from-scratch-rather-than-just-learning-to-use-the-available-libraries`.

Problem 1. The American Job [50 points]

The never-ending reality show called "The US Presidential Elections" has a saving grace: it leaves a long trail of data that we can use to come up with all sorts of nebulous "scientific" and "data-driven" conclusions. These conclusions can be, in turn, used to annoy our Twitter followers or Facebook friends.

Here is one such dataset: `https://raw.githubusercontent.com/kkehoe1985/ga_data_science_final_project/master/combined_data.csv`. As you can see, this dataset is quite messy— 78 features in total! Let's clean it up!

1. [**Bad data; 5 points**] Some cells are nonsensical: instead of having a number, they contain `"<bound method Series.mean` and multi-line data! How many such cells are there? Replace such cells with a zero.

2. [**Split; 5 points**] Split the `fips` feature into two discrete features: `State` and `County`.

    - Check out `https://www.census.gov/programs-surveys/geography/guidance/geo-identifiers.html` and `https://en.wikipedia.org/wiki/Federal_Information_Processing_Standard_state_code`. As you can see, FIPS is basically a concatenation of 2-digit state code and 3-digit county code. However, some codes have only 4-digits. Why? How can we correct them?

3. [**Aggregate; 10 points**] Aggregate and categorize the following features into a single feature:

    - Aggregate all education-related features into a single categorical feature `Education`.
    - Aggregate all religion-based features into a single categorical feature `Religion`. For each example, the appropriate category (e.g., `HighSchool` or `Mormon`) is the most represented feature that is to be aggregated. For example, having {`Amish: 100, Jewish: 200, Mormon: 150`} would result in `Religion: Jewish`.
    - Aggregate all age-related features into three numerical features: `Old` (anybody over 65 years of age), `Young` (anybody under 19), and `Adult` (the rest, obviously).
    - Aggregate different ethnic and racial groups into a single `EthnicMale` and `EthnicFemale` categorical features in the same way you did with the education-related features.
    - Remove the following features: `PovertyUnder18Pct2014`, `Deep_Pov_Children`, `Housing units`, `Area in square miles - Water area`, `Area in square miles - Land area`, `Density per square mile of land area - Housing units`, and `age_total_pop`.

4. [**Rename; 5 points**] Use canonical feature and category naming: use `CamelCase` naming and simplify names (e.g., rename `Density per square mile of land area - Population` to `PopDensity`).

5. [**Normalize; 5 points**] Normalize all population-related. income-related, and area-related features with z-score normalization (6 features in total). Ensure that all rate features are on the 0-1 scale (i.e. divide percentage features with 100; total 4 features).

6. [**Summary; 5 points**] Report the following:

    - New name, mean and standard deviation of `POP_ESTIMATE_2015` and`Population`. Also report their correlation score. Are they correlated (i.e. is their correlation score above 0.9)? If so, drop the `POP_ESTIMATE_2015`.

- New name, median, quartiles, and IQR of `PerCapitaInc` and `PovertyAllAgesPct2014` features
- Mode of combined `Religion` and `EthnicMale` feature

7. [**Visualize; 5 points**] Produce the following plots:

   - histogram of the following features: `Religion`, `EthnicMale`, `EthnicFemale`, `Education`
   - 2D scatter plot of `Area` and `Population` features
   - Box plot of normalized `PerCapitaInc` and `PovertyAllAgesPct2014` features

8. [**Conflict; 5 points**] Are there any contradicting samples in the dataset? Are there any nonsensical samples? What do they look like? How many of them are there?

9. [**Labels; 5 points**] What do you think the name of the label vector is?

The result should be saved as `elections_clean.csv`. At the end, your data frame should have roughly 3145 rows and 22 columns.

Problem 2. Decisions, decisions. [50 points]

Implement ID3 decision-tree inference algorithm ***from scratch*** and infer a decision tree from the cleaned-up US elections dataset (`elections_clean.csv`) that you have generated in Problem 1.

Implement ID3 from scratch. Use entropy-based split criteria. Only split on categorical features—do not use continuous features for splitting.

Evaluation: Shuffle the dataset. Use 70% of the shuffled dataset for training and the rest for validation. Use the same division for all subproblems. Report training and validation errors for all subproblems where applicable. Also, report the maximum tree depth and the number of features that are repeated in decision stumps.

Bonus [10 points]: If you are not happy with the results, you can make more categorical features from age-related features (hint: summarize them) and income features (hint: bin them). Do you get better predictions?

Bonus [10 points]: Use the Gini coefficient to calculate the impurity of a split. Do you observe any differences? Describe them.

Bonus [10 points]: Plot the decision boundary made by the tree! You can use the `(State, County)` tuple as the x-coordinate, and the most prominent feature (the root of the decision tree) as the y-coordinate. Limit yourself to Pennsylvania (state ID is 42). Colour (obviously) corresponds to the sample label.

Hint: The tree is not guaranteed to be clean and perfect. If you get "dirty" splits (i.e., contradictory data, or splits that contain examples whose categories are the same but whose labels do not match), use the majority vote to predict the "correct" label.

Problem 3. Pruning the tree [20 points, bonus for both sections]

Decision trees are prone to overfitting. As such, they should be pruned. Implement a pre-pruning strategy that prevents a tree from exceeding a depth of δ. Each leaf node that covers conflicting labels will predict the most likely label. What happens with the training and test error for $\delta \in \{3, 5, 7\}$? Plot the differences. Use the same evaluation data as in Problem 2.

Extra bonus [20 points]: Implement reduced error pruning on your tree. This time, use 50% of the original set (`elections_clean.csv`) for training, 25% for pruning decisions, and the remaining 25% for the final validation.

Hint: You can use `scikit-learn` for max. 10 points.

Problem 4. Library, help me out! [20 points]

Repeat everything you did in Problem 2, but use `scikit-learn` to infer a decision tree instead of your own ID3 implementation. Compare this tree with the one your code produced. Are they similar or not? Why? Are error metrics similar? You are allowed to use continuous features as well (if you prefer).

> **Hint:** `scikit-learn` does not support categorical data. Use `pandas.get_dummies` to convert categorical data to one-hot (1-of-k) encoding. Isn't it funny that you have to undo the transformations done in Problem 1?

Problem 5. Decision speed [20 points; only for CSC 503]

By default, decision stump learning requires $O(ndk)$ to find the optimal stump, where n is the number of samples, d number of dimensions (a.k.a. features), and k the number of thresholds (or categories per feature, if a feature is categorical). Can this be improved? If so, what would be the new complexity bound? If not, why? You will need to provide formal proof regardless of your answer.

Problem 6. Cross-regression [40 points]

So far, we have used decision trees primarily for binary classification. But we can also use them for regression— recall the details from the lecture! Let's give it a try.

1. [**Regress; 20 points**] We will use the same setup as in Problem 2. This time, we will introduce the following continuous features in addition to the existing categorical features: (normalized) `PerCapitaIncome` and `PovertyLevel`. Your goal this time is to predict `PovertyLevel` given other features (not the original label vector: that one should be ignored here). Again, provide relevant metrics and plots as in Problem 2.

   Hint: You can use mean to predict the poverty level for a leaf node.
   Hint: You can use `scikit-learn` to get max. 10 points here.

2. [**Cross-validate; 20 points**] Roll out your own (*implemented from scratch*) 5-fold cross-validation to select the best regression tree. You can use any decision tree implementation (your own, `scikit-learn`, etc.).

Problem 7. Old-growth forests [20 points]

Use `scikit-learn` to implement a random forest on 50 pre-pruned decision trees of depth 3. The setup is the same as in Problem 2. You are allowed to use continuous features as well. Provide the standard training and test error plots and statistics.

Are random forests better? How much? Why?

Bonus [30 points]: Implement random forests from scratch. Make sure that each tree in a forest is trained on a bootstrapped sample from your training data.

Problem 8. Linear matters [50 points]

Let's move on to the following dataset: `https://github.com/fivethirtyeight/data/blob/master/candy-power-ranking/candy-data.csv`. Yes, candies! Let's learn something about them!

- [**Linear; 10 points**] Use closed matrix form of linear regression to predict the popularity (`winpercent`) of candy from its sugar content (`sugarpercent`). You must only use NumPy and `pandas` for this. Plot the points (scatterplot), and the linear fit (straight line).
- [**EasyLinear; 10 points**] Use `scikit-learn` to do the same.
- [**NotSoLinear; 10 points**] What happens if you use polynomial regression with quadratic and cubic polynomials (e.g., $\alpha x^3 + \beta x^2 + \gamma x + \delta$) instead of a simple linear regression? Plot the fitted polynomials and the error bars. You can use `scikit-learn`.
- [**Multiple; 10 points**] What happens if you add an extra feature, say `pricepercent`, to predict `winpercent`? Use `scikit-learn` to predict the regression parameters (this time, use only simple linear features). Plot the points in 3D space and the regression hyperplane.
- [**Regularize; 10 points**] Let's apply regression on all features to predict `winpercent`. This time, use the ridge regression that penalizes the sum of squared coefficients. Use `scikit-learn` for this. Experiment with different values for regularization parameter $\lambda \in \{0.1, 1, 2\}$. Which one is the best? Are there any features that are useless for predicting how popular a particular brand of candy is?
  Again, use 70/30 split for testing and validation (see Problem 2 for more details).

Problem 9. Guessing the source [20 points]

Given a set of points from Github (`https://gist.github.com/inumanag/ebb1566746aba800899e406f03c799c1`), use linear regression to fit a line. What is the result? If the fit is not perfect, what is the best feature transformation to get a perfect fit? Prove it with a nice plot!

Problem 10. Algebraic regularization [20 points, CSC 503 only]

Can you come up with the closed-form formula for **w** when using ridge regression that penalizes the sum of squared coefficients? What would that be? Provide the formal proof.

Problem 11. Logistic matters [30 points, bonus for both sections]

Can you use linear regression to perform binary classification? If so, try to predict whether candy is a bar of chocolate or not through logistic regression on the dataset from Problem 8. Which features are the best to predict a bar of chocolate?