Final Exam

The following exam is the final exam for Applications of Data Mining. This exam is open only to the material provided for this course on the `DS 210` Canvas page, our one true website course web page, Dr. P, Dr. G, Dave P., your course notes, Zoom recordings, or internet documentation for Python, SQL, python packages. You are not allowed to consult other people or other resources UNDER ANY CIRCUMSTANCES. If you have any questions, you must bring them to one or more of us. You may be more quickly served by emailing all of us in a joint email, and the first one to get to it will respond. As is the typical case for this course, you will submit your exam materials on Canvas. All of the exam auxiliary exam materials that you need for the exam will be available for you to use on the exam website at `blue.butler.edu/ãgupta/DS210/Projects/FinalExamSupplementaryCode/`. Please let us know if you have any issues accessing that folder.

Finally, your submission on Canvas will implicitly reflect your agreement with the following statement:

Name [0 points]

Catherine Bain

Type of Community that Gupta Trolls Online [0 points]

Flat earthers and also anti-vaxers

I certify that all work on this exam is solely the product of my own efforts.

Catherine Bain

(signature)

1. [9 points, 1 point each] You Definitely Want to Over-fit This Question. Circle **T** or **F** for each of the following statements to indicate whether the statement is true or false, respectively. No justification is necessary.

T (**F**) SQLite provides a client-server model for database management. ← no, MySQL does this, SQLite uses an embedded DB

T (**F**) In SQL, `NULL` has the same meaning as `''`. also not the same as 0

(**T**) **F** A dictionary in Python is implemented with a hash table.

(**T**) **F** When developing a database schema, it is important to organize it to minimize data storage costs. but there are also many other things to consider

T (**F**) A boosted decision tree is an example of an ensemble classifier algorithm. — but you can use boosting with ensemble methods

T (**F**) A perceptron can powerfully predict nearly any kind of mathematical function.

(**T**) **F** SQL supports modifications of databases using the `UPDATE` command.

(**T**) **F** Random forests are often used instead of decision trees to lower prediction error.

(**T**) **F** A CSV file is often referred to as unstructured data in the data science community.

2. [6 points] I Need (Cross)-Validation In My Life. Each of the questions below relates to machine learning.

(a) [2 points] Explain the basic idea behind cross-validation. You may use a picture if you like, but you must explain the concept in words as well.

cross-validation is a technique used to work against overfitting that allows you to leverage multiple models for the same data. It is used in basically every machine learning technique. You produce model 1 from 60%, verify on the other 20%. You then choose a different 60% to train on for model 2 and so on and so on. Typically this is done in 10% increments, i.e. (8) train sets/models.

you can then aggregate the models to create a much stronger model.

(b) [2 points] Explain the dangers of over-fitting the input data.

for one, if you over-fit, it may lead you to only be able to predict the data you already have, i.e. memorize the actual data. This is one reason that you don't allow access to the entire dataset, because this innately prevents you from memorizing or predicting all the data, i.e. overfitting. In addition, it is likely that the sample of the data you have is not representative of the population, so overfitting that sample leads to a terrible model for the population as a whole

(c) [2 points] Explain why a random forest and a boosted decision tree are not the same.

although random forests and boosted decision trees can both be used as ensemble methods, they are different most notably in the way they differentiate classifiers. Random forests make use of a technique called bagging in which all classifiers are created equal, i.e. each classification model is equally valuable. Boosted decision trees make use of boosting where classifiers are not created equal. Boosting iterates (whereas bagging is a parallel process) and adaptively changes a given distribution based on previously misclassified information making that singular prediction model stronger all on its own, even without ensembling it with other trees or models.

important note: a single boosted decision tree is not an ensemble method, you can use boosting to strengthen individual models in the ensemble

3. [15 points] Tony the Frosted Tiger. You are hungry. And, since you are fundamentally too lazy to get food (and recently were banned from Uber Eats because of an incident involving a spork and a feral cat, long story), you decided to build a database that keeps track of all of the food that appears in every episode of Tiger King. Because, you know, people watch Tiger King to find out the latest trends in food. You came up with the following database schema. You will also note that your database was written with the hopes that there will be a *season 2* of Tiger King. I mean there's basically no way they're not bringing these characters back for a second appearance, amirite? You'll be answering questions about this database.

except one of them is in jail now

```
Episodes(id, season, name)
Foods(id, food_type_id, name)
FoodTypes(id, name)
FoodEpisodes(food_id, episode_id)
```

★ (a) [5] Write a SQL query that outputs the number of foods from each category, along with their food type id and their name.

```
SELECT food_type_id, FoodTypes.name, COUNT(Foods.id) FROM Foods
INNER JOIN FoodTypes ON FoodTypes.id = Foods.food_type_id
GROUP BY food_type_id;
```

(b) [5] Write a SQL view that outputs all foods that appeared on shows in season 6 or later.

```
CREATE VIEW season6Foods AS
SELECT Foods.name FROM Foods
INNER JOIN FoodEpisodes ON FoodEpisodes.food_id = Foods.id
INNER JOIN Episodes ON Episodes.id = episode_id
WHERE Episodes.season >= 6;
```

(c) [5] Write a SQL query that outputs the top 10 most common foods from season 6 or later. *Hint: You may find it useful to create a view first.*

```
SELECT name FROM season6Foods
GROUP BY name
ORDER BY COUNT(name) DESC
LIMIT 10;
```

4. [15 points] Regular Depression. This question requires you to write a short Python script that processes a web server's access log. In case you don't know, all web servers keep a log of all requests for pages called from that server. These `access_log` files can become very large very quickly so they are usually summarized in a report and then deleted regularly. We have provided a small snippet of an `access_log` file on the web page mentioned at the beginning of the exam for this question. This snippet file follows the *Combined Log* format. Here is a web link that explains the information contained in each part of that file:

`http://fileformats.archiveteam.org/wiki/Combined_Log_Format`

Your task is to write a Python script to extract information from the `access_log` file to answer the following questions:

count the number of rows in the file

1. The number of total access calls in the log file
2. The number of unique IP addresses (essentially, computers) that made server requests
3. The number of requests made by each of the following types of "computers":
   - Windows
   - Mac
   - iPhone
   - Android Phone
   - Spider/Bot (an automated program that reads pages)
   - Other/Unknown

dictionary with IP as a key and count as a value

↳ could sum this count to get total

You should print this information out to the screen in some sort of easy-to-read format.

For this portion of the exam, you will submit `question4.py` and a corresponding output text file called `question4.txt` on Canvas. An easy way to produce this output file without actually writing Python code to write to a file is to execute the following command directly on Thomas:

```
python3 question4.py > question4.txt
```

And *voila* you have an output file! (This technique, by the way, is called *redirecting* the output of an executable file. It's a useful concept in a Unix/Linux environment.)

5. [40 points] That's What SHE Said About Deep Learning. The following questions are based on the following table of data and will be used to predict whether or not someone will be released after a recent hospital visit.

Patient	Major OP?	Family Support?	Old?	Discharged?
1	yes	no	~~no~~	~~no~~
2	yes	no	~~yes~~	~~no~~
3	no	no	no	yes
4	no	no	yes	no
5	~~no~~	~~yes~~	~~yes~~	~~yes~~

(a) [15 points] Build any valid decision tree that correctly classifies all of the training examples provided above. Provide the overall calculation (and all intermediate calculations) to calculate the GINI for the **root node** of your decision tree. (*Before you get started on part (a), look at part (b). If you approach these problems the right way, you may be able to save some effort.*)

C_1: Yes Discharged
C_2: Not Discharged

Major OP
- Yes: C_1: 0, C_2: 2 → $1-(\frac{0}{2})^2-(\frac{2}{2})^2=0$
- No: C_1: 2, C_2: 1 → $1-(\frac{2}{3})^2-(\frac{1}{3})^2=.4444$

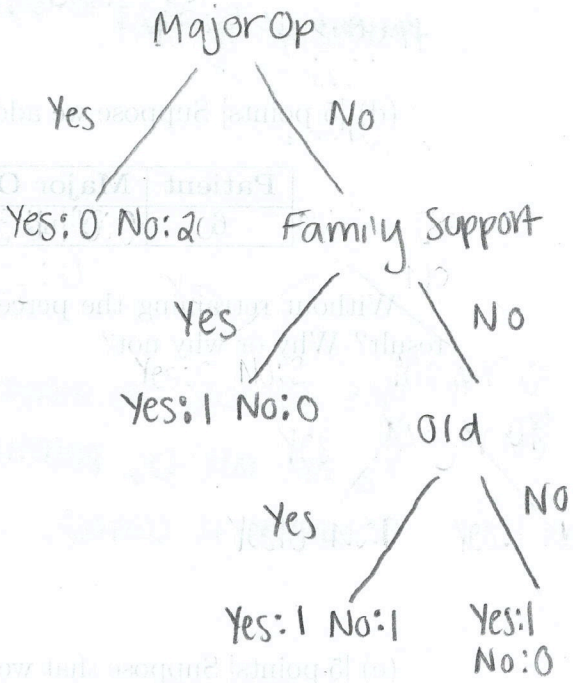
GINI(Major) = $\frac{2}{5}(0)+\frac{3}{5}(.4444)=.266664$

Family
- Yes: C_1: 1, C_2: 0 → $1-(\frac{1}{1})^2-(\frac{0}{1})^2=0$
- No: C_1: 1, C_2: 3 → $1-(\frac{1}{4})^2-(\frac{3}{4})^2=.375$

GINI(Family) = $\frac{1}{5}(0)+\frac{4}{5}(.375)=.3$

Old
- Yes: C_1: 1, C_2: 2 → $1-(\frac{1}{3})^2-(\frac{2}{3})^2=.4444$
- No: C_1: 1, C_2: 1 → $1-(\frac{1}{2})^2-(\frac{1}{2})^2=.5$

GINI(Old) = $\frac{3}{5}(.4444)+\frac{2}{5}(.5)=.466664$



(b) [5 points] Improve your prior decision tree by either showing a new tree with a better split (and an overall better GINI). If you had instead built an optimal tree to begin with, you will automatically get these points. (But you must justify that your tree was, in fact, optimal.)

GINI(Old | No MOP)
- Yes: C_1: 1, C_2: 1 → $1-(\frac{1}{2})^2-(\frac{1}{2})^2=.5$
- No: C_1: 1, C_2: 0 → $1-(\frac{1}{1})^2-(\frac{0}{1})^2=0$

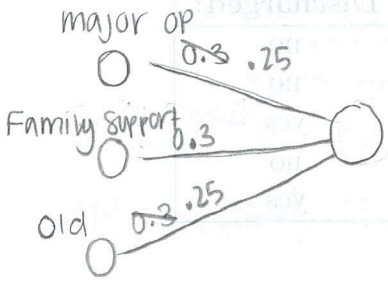
↳ $\frac{2}{3}(.5)+\frac{1}{3}(0)=.3333$

GINI(Fam | No MOP)
- Yes: C_1: 1, C_2: 0 → $1-(\frac{1}{1})^2-(\frac{0}{1})^2=0$
- No: C_1: 1, C_2: 1 → $1-(\frac{1}{2})^2-(\frac{1}{2})^2=.5$

↳ $\frac{1}{3}(0)+\frac{2}{3}(.5)=.3333$

When we run GINI a second time to divide the No side of Major OP, we find that Old and Family Support have the same GINI index of .3333. Therefore, it does not matter which criteria we split on next. I chose to split on Family Support because it had a lower GINI score during the first level analysis. However, the only difference between splitting on FS v. Old is which side of the tree you end up splitting. FS splits no while Old splits Yes.

(c) [10 points] Build a single perceptron "neural network", with initial weights on each edge initialized to 0.3. We will use an incremental gradient of 0.05. Run the perceptron through one round of training on the full dataset. Did it converge, and if not, would you expect the algorithm to converge? Why or why not?



Patient 1: $1(.3)+0(.3)+0(.3)=.3 \Rightarrow 0$
Patient 2: $1(.3)+0(.3)+1(.3)=.6 \Rightarrow 1$
Patient 3: $0(.25)+0(.3)+0(.25)=0 \Rightarrow 0$
Patient 4: $0(.25)+0(.3)+1(.25)=.25 \Rightarrow 0$
Patient 5: $0(.25)+1(.3)+1(.25)=.55 \Rightarrow 1$

No, the algorithm will never converge to the correct answer because x_1, x_2, and x_3 are all 0 for patient 3 which will give an output of 0 regardless of the weights when in reality, that patient had an outcome of 1.

(d) [5 points] Suppose we added another patient, as indicated in the table below:

Patient	Major OP?	Family Support?	Old?	Discharged?
6	no	no	no	no

Without retraining the perceptron, would you expect the the network to give the correct result? Why or why not?

Yes because no matter what the weights are, since x_1, x_2, and x_3 are all 0, patient 6 will always have the predicted value of 0 which is correct.

(e) [5 points] Suppose that we modified the input data slightly, and allowed the **Old** variable to instead have a range of age categories, for example `baby`, `child`, `adult`, or `senior`. How might you attempt to modify the neural network training to account for this more detailed information?

Well, one way to do it would be to separate the "old" variable into four other variables "baby", "child", "adult", or "senior" where the variable matching the value in "old" for each participant would recieve a value of yes (or 1) and the other 3 would recieve a value of No (or 0). This would solve that problem for sure. To do this, you use binary vectors. It is also possible to argue that assigning each value an ordered integer, i.e. baby=1, child=2, adult=3, senior=4 since, this particular example having a hierarchy based on age or "how" old they are may well be related to them being released. Before doing that, I might run a correlation to see if there is a relationship like that first.

6. [15 points] eMotion Drug Dealers. You are a senior data manager that is responsible for developing a service called XSTREAM that will stream hard-hitting dramas online based on their emotional content. The stronger a reaction it gets from the audience, the more you want it in your service. The company already has a detailed database of 100000 hit dramas that have been tagged with exactly one emotion: romance, comedy, action, thriller, or horror. Your job is to develop an experimental setup that can predict the emotion of new dramas that are added to the database so that you can avoid doing work in the future. Unfortunately, you must do work now to safeguard your ability to do no work in the future.

You have at your disposal a piece of software called `iDramaMine` that can capture around 1000 secondary features from each drama. Those features include things like common phrases, actor information, word sentiment, and so on. Some of the data is real-valued, and some is categorical. `iDramaMine` collects data that is also dirty, in that it sometimes extracts features from dramas that are not relevant to its emotional characterization. Furthermore, it may simply make mistakes like extracting words from silent dramas or incorrectly learning an actor's name as "Ryan Goosling." Engineers are working on improving `iDramaMine`, but you don't have control over the development lifecycle for that software, so you must use it as is.

Propose and state a full data flow (from start to finish) for how to utilize, clean, and engineer the information collected by `iDramaMine` to train, validate, and manage a (collection of) supervised learning techniques for predicting the emotional content of new dramas. This question is *not* asking you to write code. Your only job is to build an experimental framework that describes what happens to the data and what interactions different must happen with that data. You may assume that the individuals that are executing your data plan will flesh out instructions appropriately, but the more guidance you provide, the more likely they are to capture the spirit of the data plan.

The first thing that must be done is cleaning the data. This can include making sure all necessary information is present. For example, if an entry in the data does not have a coorresponding movie title, it cannot be used. In addition, data should be type checked, i.e. years should be integers, movie titles should be strings, etc. In addition, things that can be checked for validity should. For example, the year of release cannot be later than the current year. We should also check for duplicate entries, i.e. same title, actresses, and year of release it's likely to be a duplicate. Check for uniformity in formatting, i.e. times/dates in USA formatting not combination of USA and European. In addition we should remove irrelevant data like age of actors, ratings on Rotten tomatoes. things that will not help us predict. Once the data has been cleaned, it needs to be organized to make data interpretation/data analysis easier and more efficient. Once the data has been organized, it needs to be divided into data that is accessible to the analyst and hidden data. this step is incredibly important because it is the first line of defense against overfitting. It is typically done in an 80/20 split with 20% being hidden. Then the analyst should begin the process of learning/predicting the data. This can be done in a variety of ways such as linear classifiers like linear regression or Naive Bayes classifier, support vector machines, K-nearest neighbor, decision trees, neural networks, etc. Analysts should use many of these models to determine which has the strongest ability to predict correctly. In addition, ensemble methods like random forests could be implemented to leverage many weak models if a strong model cannot be found. In addition, preliminary models could be run to select the predictor variables if they aren't selected randomly. Whatever model is chosen in the end, the team should 7 be sure it is the strongest model w/o overfitting. In addition, they should employ cross validation or other methods to control for overfitting.

BONUS. [**10 points**] This question uses the original five patient data table from question 5. Iterate through two generations of a genetic algorithm that tries to predict whether or not someone will be discharged. The two initial parent strings will be `Y#Y` and `NY#`, and we are allowed to keep at most 4 genomes at any one time. You will score a genome as a probability of success on all examples given.

↳ we will use accuracy as our measure of success

$S = \{$ Y#Y, NY# $\}$
(4/5) (5/5)

Iteration 1:

Y#Y — Y## (3/5)
NY# — NYY (5/5)

$S = \{$ Y#Y, NY#, Y##, NYY $\}$

Y#Y => Y#N (4/5)

$S = \{$ Y#Y, NY#, NYY, Y#N $\}$ ← purged Y## b/c it had the lowest prob of success

Iteration 2:

Y#N — YYY (5/5)
NYY — N#N (5/5)

NY# => NN# (4/5)

$S = \{$ NY#, NYY, YYY, N#N $\}$ ← purged Y#Y, NN# & Y#N because they had the lowest probability of success