
Midterm Exam - EECS 398, Spring 2025

Full Name:

Uniqname:

UMID:

Time: ☐ 2-4PM ☐ 2-5PM (SSD 150%) ☐ 2-5:30PM (SSD 175%)

Instructions:

- You have 120 minutes to complete this exam.
- This exam consists of 12 questions, worth a total of 80 points.
- Write your unickname in the top right corner of each page in the space provided.
- Please write **clearly** in the provided answer boxes; we will not grade work that appears elsewhere. Completely fill in bubbles and square boxes; if we cannot tell which option(s) you selected, you may lose points.
 - ☐ A bubble means that you should only **select one choice**.
 - ☐ A square box means you should **select all that apply**.
- You may refer to a single two-sided handwritten notes sheet. Other than that, you may not refer to any other resources or technology during the exam (no phones, watches, or calculators).

You are to abide by the University of Michigan/Engineering Honor Code. To receive a grade, please sign below to signify that you have kept the honor code pledge.

I have neither given nor received aid on this exam, nor have I concealed any violations of the Honor Code.

Signature:

Data Overview: Cities in the United States

In this exam, we'll work with the DataFrame `cities`, which contains information about cities and towns in the United States.

The first few rows of `cities` are shown below, but `cities` has many more rows than are shown.

	city	state	county	pop	importance	zips														
0	Los Angeles	California	Los Angeles	11885717	1	90291 90293 90292 91316 91311 90035 90034 9003...														
1	Detroit	Michigan	Wayne	3716929	1	48209 48208 48202 48201 48207 48206 48205 4820...														
2	Louisville	Kentucky	Jefferson	965005	2	40245 40242 40241 40218 40219 40214 40216 4021...														
3	Anaconda	Montana	Deer Lodge	9512	4	59722 59756 59711 59762														
4	Birmingham	Alabama	Jefferson	778756	2	35218 35214 35215 35217 35210 35211 35212 3521...														
5	Grand Canyon West	Arizona	Mohave	0	5	86434														
6	Mentor	Ohio	Lake	47215	3	44060 44061														

Each row in `cities` contains information about a single city. The columns in `cities` are as follows:

- `"city"` (`str`): The name of the city.
- `"state"` (`str`): The state the city is in.
- `"county"` (`str`): The county the city is in.
- `"pop"` (`int`): The population of the city.
- `"importance"` (`int`): The importance of the city, ranging from 1 (most important) to 5 (least important).
- `"zips"` (`str`): A string containing all of the zip codes found within the city. If a city has multiple zip codes, they are each separated by a single space.

Throughout the exam, assume we have already run all necessary import statements.

username: _____

Make sure you have read the Data Overview before beginning!

Question 1 (5 pts)

Fill in the blanks below so that `option_one` and `option_two` **both** evaluate to the number of cities with an importance score of 5.

```
option_one = __(i)__.sum()
option_two = __(ii)__.loc[5]
```

(i):

(ii):

Question 2 (8 pts)

We say an **antidominated** state is one in which the most populated city in the state accounts for **less than 20%** of the total population in the state.

Fill in the blanks so that the expression below returns the **name** of the **most populated** antidominated state.

```
(cities
 .groupby(__(i)__)
 .__(ii)__(lambda x: __(iii)__)
 .groupby(__(iv)__)
 .__(v)__
 .index[-1]
)
```

(i): ☐ "city" ☐ "state" ☐ "county" ☐ "pop"

(ii): ☐ agg ☐ filter ☐ transform ☐ value_counts

(iii):

(iv): ☐ "city" ☐ "state" ☐ "county" ☐ "pop"

(v):

Question 3 (8 pts)

City names are not unique within a state. For example, there are multiple cities named Georgetown in Pennsylvania.

- a) (2 pts) Fill in the blank: If the expression below evaluates to `----`, it means that there are no cases in the dataset where two cities in the same county have the same name. Otherwise, at least one such case exists.

```
list(cities.groupby(["city", "state", "county"]).size().unique())
```

What goes in the blank?

- b) (6 pts) Consider the following expression and its output.

```
>>> cities.groupby(["city", "state"]).size().value_counts()
1      31119
2         57
3          7
Name: count, dtype: int64
```

Answer each of the following questions using only the output above. If the answer is a mathematical expression, leave the answer in unsimplified form, e.g. something of the form “ $1^2 + 2^2 + 3^2$ ” is fine. If it is impossible to answer a question using just this information, write “impossible”.

- (i) What is the largest number of times a city name is repeated within a state?

- (ii) How many states have at least one duplicated city name?

- (iii) How many unique city names are there in the entire country?

- (iv) How many rows are in `cities`?

Question 4 (8 pts)

Angela and Abhi both want to visit new places this summer. To decide where to go, they each pick 10 random rows without replacement from `cities`. Angela's 10 cities are stored in the DataFrame `angela`, while Abhi's 10 cities are stored in the DataFrame `abhi`.

- a) (2 pts) Suppose Angela and Abhi want to travel together, and want to visit all cities that were selected by **at least one** of them. Fill in the blank so that `num_unique` evaluates to the **number of unique states** that they will visit.

```
merged = angela.merge(abhi, __ (i) __)
num_unique = merged["state"].nunique()
```

(i):

- b) (6 pts) This part is independent of the last part. Consider the following expressions and their outputs.

```
>>> angela["state"].value_counts()
Colorado      5
California    2
Kentucky      2
Michigan       1
Name: count, dtype: int64
```

```
>>> abhi["state"].value_counts().value_counts()
3          2
1          2
2          1
Name: count, dtype: int64
```

To maximize the number of rows in `angela.merge(abhi, on="state")`, what specific sequence of 10 state names should the Series `abhi["state"]` Series contain? List the 10 state names separated by commas. There may be multiple correct answers; you only need to provide one.

```
abhi["state"] = pd.Series([
```

])

Question 5 (8 pts)

Consider the DataFrame `small_cities`, shown in its entirety below. Some of the values in the "importance" column of `small_cities` are missing.

	city	state	importance
0	Pajaro Dunes	California	4.0
1	Lennox	California	2.0
2	Menifee	California	NaN
3	Middletown	Michigan	3.0
4	Sault Ste. Marie	Michigan	NaN
5	Sidman	Pennsylvania	2.0
6	Puzzletown	Pennsylvania	NaN
7	Industry	Pennsylvania	4.0

In each part, assume that `filled` is a Series of length 8, resulting from imputing (filling) the missing values in the "importance" column of `small_cities` using some imputation strategy.

In parts (a) and (b), the “Mean imputation” and “Probabilistic imputation” options refer to unconditional mean imputation and unconditional probabilistic imputation, respectively.

- a) (2 pts) Suppose that `filled.mean()` evaluates to 3.0. Which of the following imputation strategies could have been used to create `filled`? Select all that apply.

- ☐ Mean imputation
 ☐ Mean imputation conditioned on state
☐ Probabilistic imputation
 ☐ Probabilistic imputation conditioned on state

- b) (2 pts) Suppose that `filled.mean()` evaluates to 3.125. Which of the following imputation strategies could have been used to create `filled`? Select all that apply.

- ☐ Mean imputation
 ☐ Mean imputation conditioned on state
☐ Probabilistic imputation
 ☐ Probabilistic imputation conditioned on state

- c) (4 pts) If we use probabilistic imputation conditioned on state, there are only three possible values for `filled.mean()`. Below, provide the possible values for `filled.mean()`, along with their probabilities. You need to provide six numbers total; two of them have been provided for you.

Possible value of <code>filled.mean()</code>	Probability of value
2.75	1 / 4

username: _____

Question 6 (7 pts)

The following HTML document contains information about tourist destinations in Michigan. The preview below only shows information about two cities on the site, but there are many more, as indicated by the ellipses (...).

```
<div class="block" blockname="a2"><h2>Ann Arbor</h2>
<p>Things to Do:<ul>
  <li>Visit University of Michigan</li>
  <li>Explore downtown bookstores</li>
  <li>Walk in the Arboretum</li>
</ul></p><p style="coord" loc="42.2808° N, 83.7430° W">Navigate here</p>
</div>

<div class="block" blockname="dtw"><h2>Detroit</h2>
<p>Things to Do:<ul>
  <li>Detroit Institute of Arts</li>
  <li>Ford Field</li>
</ul></p><p style="coord" loc="42.3314° N, 83.0458° W">Navigate here</p>
</div>

<div class="block" blockname="tc"><h2>Traverse City</h2>
...
```

Assume that `soup` is a BeautifulSoup object instantiated on the HTML document above.

Complete the implementation of the function `find_location`, which takes in the name of a city as a string and returns its latitude and longitude. Example behavior is given below.

```
>>> find_location("Ann Arbor")
"42.2808° N, 83.7430° W"

def find_location(city_name):
    cities = soup.__(i)__
    for city_soup in cities: # Assume that city names in Michigan are unique.
        this_city_name = __(ii)__
        if city_name == this_city_name:
            return __(iii)__
```

(i):

(ii):

(iii):

Question 7 (7.5 pts)

If `s` is a Series of strings, then `s.str.fullmatch(exp)` returns a Series of Booleans with the same length as `s`. Each element is `True` if the **entire** corresponding string matches the regular expression `exp`, and `False` otherwise.

Example behavior is given below.

```
>>> s = pd.Series(["hey there", "how is he"])
>>> s.str.fullmatch(r"he.*")
0      True
1     False # Requires the entire string to match, not just a substring.
dtype: bool
```

Consider the regular expressions labeled A through G listed below. (The `r` at the start of each string denotes a raw string in Python.)

- A. `r"[\w]+r"`
- B. `r"A[A-Za-z]+ [A-Za-z]+r"`
- C. `r"A[A-Za-z]+ [A-Z][a-z]+r"`
- D. `r"(\w+ \w+)+\w*"`
- E. `r"[\w]+ [\w]+"`
- F. `r"(\w+ \w+)+"`
- G. `r"\w+ \w+"`

In each part, **pick the regular expression** `exp` from the options above, such that

```
cities[cities["city"].str.fullmatch(exp)].shape[0]
```

is the answer to the given question. The first part is done for you. **Assume that the only characters in city names are letters, numbers, underscores, and spaces.**

a) How many city names end with `"r"`?

- ☒ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

b) (2.5 pts) How many city names contain at least one space (that is not at the start or end of the name)?

- ☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

c) (2.5 pts) How many city names contain a **positive** even number of spaces (2, 4, 6...)?

- ☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

d) (2.5 pts) How many city names are made up of exactly two words, where both words are at least two characters long, start with uppercase letters, and have no numbers, and where the first word starts with `"A"` and the last word ends with `"r"` (like in `"Ann Harbor"`)?

- ☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

Question 8 (7.5 pts)

Consider the following three state mottos:

- State 1: **strength for freedom**
- State 2: **justice and freedom**
- State 3: **freedom and strength**

The following options, labeled A through G, are possible mottos for State 4.

- A. **justice and justice**
- B. **justice and justice and**
- C. **justice justice freedom justice and justice**
- D. **freedom and freedom and**
- E. **freedom freedom and freedom**
- F. **freedom for freedom for**
- G. **freedom justice freedom**

In each part, you are given information about various state mottos, and your job is to select a motto for State 4 from the list above that satisfies the assumptions provided in that part only. Assume we use base 2 logarithms.

- a) (2.5 pts) Given that the cosine similarity between the **bag of words** representation of State 1's motto and State 4's motto is $\frac{2}{\sqrt{24}} \left(= \frac{1}{\sqrt{6}} \right)$, which option could be State 4's motto?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

- b) (2.5 pts) Given that:

- the TF-IDF of **freedom** in State 1's motto is 0, and that
- the TF-IDF of **justice** in State 4's motto is $\frac{2}{3}$,

which option could be State 4's motto?

☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

- c) (2.5 pts) Given that:

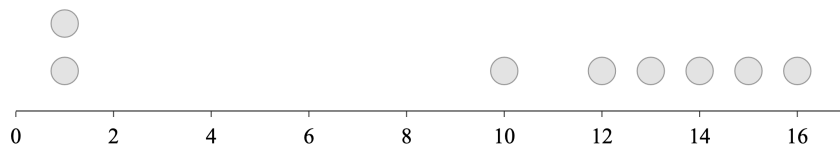
- the TF-IDF of **justice** in State 2's motto is $\frac{2}{3}$, and that
- the TF-IDF of **for** in State 4's motto is $\frac{1}{2}$,

which option could be State 4's motto?

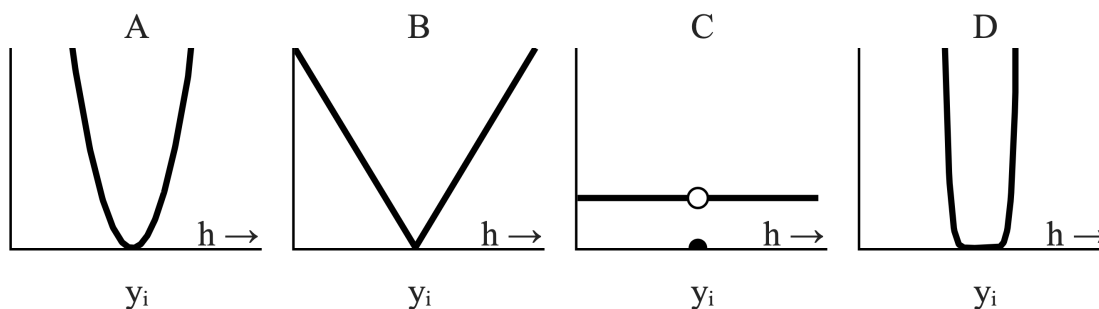
☐ A ☐ B ☐ C ☐ D ☐ E ☐ F ☐ G

Question 9 (8 pts; 2 pts each)

Suppose we'd like to fit a constant model, $H(x_i) = h$, to the dataset y_1, y_2, \dots, y_8 shown below.



To do so, we consider four loss functions $L(y_i, h)$, each of which is graphed below.



For reference, A is squared loss.

In each part below, the optimal constant prediction h^* that minimizes average loss, for one of the loss functions above, is drawn as a black vertical line. In each part, select the loss function that was used.

- a) ☐ A ☐ B ☐ C ☐ D
- b) ☐ A ☐ B ☐ C ☐ D
- c) ☐ A ☐ B ☐ C ☐ D
- d) ☐ A ☐ B ☐ C ☐ D

Question 10 (4 pts)

In this question, we will work with the populations of cities in thousands. For example, if Ann Arbor's population is 120,000, its population in thousands is 120. Working with populations in thousands allows us to use smaller numbers.

Suppose we'd like to predict the population of a city in thousands (y) given the number of zip codes it has (x). To do so, we fit a simple linear regression model of the form $H(x_i) = w_0 + w_1x_i$ to the `cities` dataset using squared loss.

After fitting our model, we find that, according to the model:

- For a city with 3 zip codes, the city's predicted population in thousands is 45.
- For a city with 5 zip codes, the city's predicted population in thousands is 105.

- a) (2 pts) What are w_0^* , the model's intercept, and w_1^* , the model's slope? Give your answers as numbers with no variables. (Don't accidentally reverse the intercept and slope!)

$w_0^* =$ $w_1^* =$

- b) (2 pts) Given that the average number of zip codes for cities in the dataset is 2, what is the average population of cities in the dataset, in thousands? Give your answer as a number with no variables. (Since populations are already measured in thousands, your answer should not end with 000.)

Average population (in thousands) =

Question 11 (4 pts)

Identify the most appropriate data visualization type, among those covered in class, for each of the scenarios below. Use 1-2 words for each answer.

- a) (2 pts) The mean population of each state.

- b) (2 pts) The distribution of city populations within Michigan.

Question 12 (5 pts)

For your convenience, the first few rows of `cities` are shown again below.

	city	state	county	pop	importance	zip
0	Los Angeles	California	Los Angeles	11885717	1	90291 90293 90292 91316 91311 90035 90034 9003...
1	Detroit	Michigan	Wayne	3716929	1	48209 48208 48202 48201 48207 48206 48205 4820...
2	Louisville	Kentucky	Jefferson	965005	2	40245 40242 40241 40218 40219 40214 40216 4021...
3	Anaconda	Montana	Deer Lodge	9512	4	59722 59756 59711 59762
4	Birmingham	Alabama	Jefferson	778756	2	35218 35214 35215 35217 35210 35211 35212 3521...
5	Grand Canyon West	Arizona	Mohave	0	5	86434
6	Mentor	Ohio	Lake	47215	3	44060 44061

The federal government would like to award two randomly selected neighborhoods with funding to improve their roads and schools. To do so, they:

1. Pick one city at random, from the set of cities **with at least two zip codes**.
2. Pick two **different** zip codes at random from that city.

Complete the implementation of the function `simulate_selection`, which takes in no arguments and returns two zip codes selected using the process above. Example behavior is given below.

```
>>> simulate_selection()
array(["92247", "92253"])
```

```
>>> simulate_selection()
array(["62948", "62933"])
```

```
def simulate_selection():
    valid_cities = cities[__(i)__]
    city_zips = np.random.choice(valid_cities["zip"])
    return ____(ii)__
```

(i):

(ii):

username: _____

Make sure you've written your username in the space provided in the top right corner of every page of this exam.

Congrats on finishing the exam! Feel free to draw us a picture about Practical Data Science below :)

A large, empty rectangular box with a thin black border, intended for a drawing about Practical Data Science.