Midterm Exam - EECS 398, Winter 2025

Full Name:

Uniqname:

UMID:

Room: ◯ 1010 DOW ◯ 1013 DOW ◯ 1017 DOW ◯ 1018 DOW ◯ 2311 EECS ◯ TAC

Instructions:

- This exam consists of 15 questions, worth a total of 90 points. You have 120 minutes to complete the exam.
- Write your uniqname 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:

Version A

Data Overview: Food Deliveries

In this exam, we'll work with the DataFrame `orders`, which contains information about deliveries made using various food delivery services in India.

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

	driver	type	rating	dist	traffic	minutes
0	BANGRES05DEL02	Buffet	4.5	15.56	High	39.98
1	VADRES01DEL03	Drinks	4.0	7.41	Low	26.15
2	MUMRES06DEL03	Snack	4.8	13.16	High	45.16
3	BANGRES17DEL01	Drinks	4.7	20.00	Very High	62.35
4	HYDRES19DEL01	Meal	4.2	14.67	High	39.76

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

- `"driver" (str)`: The delivery driver's ID. Note that there are duplicate values in this column, because some drivers have made multiple deliveries.
- `"type" (str)`: The type of food ordered; either `"Buffet"`, `"Drinks"`, `"Meal"`, or `"Snack"`.
- `"rating" (float)`: The average rating of the driver out of 5, **after** making the given delivery.
- `"dist" (float)`: The distance from the restaurant to the delivery address, in kilometers.
- `"traffic" (str)`: The level of traffic; either `"High"`, `"Low"`, `"Moderate"`, or `"Very High"`.
- `"minutes" (float)`: The number of minutes between the order being placed by the customer and the order being delivered by the driver.

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

uniqname: _____

Make sure you have read the Data Overview before beginning!

Question 1 (4 pts)

Driver `"WOLVAA01"` has made several deliveries. Write an expression that evaluates to the number of minutes `"WOLVAA01"` took to deliver their **second**-fastest order.

Question 2 (5 pts)

Drivers' ratings can change over time, as they perform more and more deliveries. For instance, after one delivery a driver's rating might be 4.8, and after their next delivery it might drop to 4.7. We say a driver is *consistent* if their rating was the exact same for all of their deliveries in `orders`.

Fill in the blanks so `prop_consistent` below evaluates to a float, corresponding to the **proportion of drivers** who are consistent.

```
def f(x):
    return __(iv)__

prop_consistent = (
 orders
 .groupby(__(i)__)
 [__(ii)__]
 .__(iii)__(f)
 .mean()
)
```

(i): _____ (ii): _____

(iii): ◯ agg ◯ filter ◯ transform

(iv): _____

Question 3 (3 pts)

Suppose the expression below evaluates to `1`.

```
(orders[orders["type"] == "Snack"]["driver"]
 .value_counts()
 .value_counts()
 .shape[0]
)
```

What can we conclude about `orders`?

- ◯ Every driver made at least 1 `"Snack"` delivery.
- ◯ Every driver made exactly 1 `"Snack"` delivery.
- ◯ Every driver made exactly k `"Snack"` deliveries, where k is some positive constant.
- ◯ Every driver made exactly 0 or exactly k `"Snack"` deliveries, where k is some positive constant.
- ◯ Every driver that made a `"Snack"` delivery did not make any other kind of delivery.

Question 4 (3 pts)

In one English sentence, describe what the following expression computes. Your sentence should start with "The driver", and should be understandable by someone who has never written code before, i.e. it should not use any technical terms.

```
(orders
 .groupby("driver")
 .filter(lambda df: df.shape[0] >= 10 and df["rating"].min() >= 4.5)
 .groupby("driver")
 ["minutes"]
 .sum()
 .idxmax()
)
```

uniqname: _____

Question 5 (6 pts)

Consider the DataFrame `C`, defined below.

```
C = orders.pivot_table(
  index="type", # Possible values: "Buffet", "Drinks", "Meal", "Snacks"
  columns="traffic", # Possible values: "High", "Low", "Moderate", "Very Low"
  values="minutes",
  aggfunc="count"
)
```

Throughout this question, assume that after defining `C` above, **we sort `C` such that both its index and columns are in ascending alphabetical order** (as shown above).

a) (3 pts) Fill in the blanks so that the expression below evaluates to the **proportion of `"Snack"` deliveries that were made in `"Moderate"` traffic**. Each blank should be filled with a single integer, float, string, or Boolean value.

```
C.iloc[__(i)__, __(ii)__] / C.loc[__(iii)__].sum()
```

(i): [ ] (ii): [ ] (iii): [ ]

b) (3 pts) Fill in the blanks so that the expression below evaluates to the **proportion of deliveries made in `"Low"` traffic that were for `"Buffet"`s**. Blanks (i) and (ii) should each be filled with a single integer, float, string, or Boolean value.

```
C.loc[__(i)__, __(ii)__] / __(iii)__.sum()
```

(i): [ ] (ii): [ ]

(iii):

- ◯ `C.iloc[:, -1]`
- ◯ `C.iloc[:, 0]`
- ◯ `C.iloc[:, 1]`
- ◯ `C.iloc[-1, :]`
- ◯ `C.iloc[0, :]`
- ◯ `C.iloc[1, :]`

Question 6 (9 pts)

Consider the DataFrames `A` and `B`, shown below in their entirety.

	driver	type	rating
0	CHENRES18DEL01	Drinks	4.6
1	JAPRES20DEL01	Buffet	4.8
2	CHENRES05DEL03	Buffet	4.7
3	CHENRES20DEL02	Buffet	4.7
4	MUMRES12DEL03	Buffet	3.3
5	COIMBRES15DEL02	Drinks	4.2
6	MYSRES12DEL03	Buffet	4.6

A

	driver	type	rating
0	MUMRES02DEL02	Buffet	4.6
1	PUNERES18DEL03	Meal	4.9
2	MYSRES17DEL02	Snack	4.7
3	CHENRES12DEL02	???	4.6
4	COIMBRES010DEL03	Snack	4.5
5	BANGRES13DEL02	Meal	4.7

B

Note that the `"type"` value in row **3** of DataFrame `B` is unknown. Remember from the Data Overview page that the only possible values in the `"type"` column are `"Buffet"`, `"Drinks"`, `"Meal"`, and `"Snack"`.

a) (2 pts) Suppose the DataFrame `A.merge(B, on="type")` has 7 rows.
What must the unknown value, ???, be?
◯ `"Drinks"` ◯ `"Buffet"` ◯ `"Meal"` ◯ `"Snack"`

b) (2 pts) Suppose the DataFrame `A.merge(B, on="type")` has 10 rows.
What must the unknown value, ???, be?
◯ `"Drinks"` ◯ `"Buffet"` ◯ `"Meal"` ◯ `"Snack"`

c) (3 pts) Suppose the DataFrame `A.merge(B, on="type", how="outer")` has k rows. The value of k depends on the unknown value, ???.
Three of the following four integers *could* be k; which option **cannot** be k?
◯ 11 ◯ 12 ◯ 14 ◯ 16

d) (2 pts) Suppose the unknown value, ???, is `"Buffet"`.
How many rows are in the DataFrame `A.merge(B, on=["type", "rating"])`?
◯ 0 ◯ 1 ◯ 2 ◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7

uniqname: _____

Question 7 (7 pts)

The delivery company wants to reward a subset of its drivers with a gift card for their service. To do so, they:

1. Choose an integer k between 1 and 10 inclusive, uniformly at random.
2. Choose k **unique** drivers, uniformly at random, such that all drivers have the same chance of being chosen, no matter how many deliveries they have made. Note that a driver cannot be selected more than once.

Driver `"WOLVAA01"` asks ChatGPT to write code that simulates the probability that he wins a gift card, and it gives him back the following:

```
drivers = orders["driver"].unique()

def choose_k():
    return np.random.choice(np.arange(1, 11))

def one_sim(k):
    selected = np.array([])
    for i in range(k):
        selectee = np.random.choice(drivers, 1, replace=False)
        selected = np.append(selected, selectee)
    return "WOLVAA01" in selected

def simulation():
    k = choose_k()
    N = 100_000
    total = 0
    for i in range(N):
        total = total + one_sim(k)
    return total / N
```

`simulation()` *should* return an estimate of the probability that `"WOLVAA01"` wins a gift card, but some of the code is potentially buggy. **Select all issues** with the code above.

- ☐ `drivers` only includes the drivers that made one delivery, rather than all drivers.
- ☐ `choose_k` returns a random integer between `1` and `11`, instead of one from `1` to `10`.
- ☐ `choose_k` always returns the same number.
- ☐ `one_sim` draws `k` drivers with replacement, instead of without replacement.
- ☐ `one_sim` draws `k` drivers without replacement, instead of with replacement.
- ☐ `one_sim` always returns `False`, because `selected` is always an empty array.
- ☐ `simulation` only picks one value of `k`, when it should select a new `k` on each iteration.
- ☐ None of the above.

Question 8 (9 pts)

Suppose the DataFrame `D` contains a subset of the rows in `orders`, such that:

- Some of the values in the `"minutes"` column are missing.
- None of the values in the `"minutes"` column are missing for orders made in `"Moderate"` traffic.

Each part of this question is independent of all other parts. In each part, **select all** expressions that are **guaranteed** to evaluate to the same value, **before and after** the imputation code is run. The first part is done for you.

a)

```
t = lambda x: x.fillna(x.mean())
D["minutes"] = t(D["minutes"])
```

☐ `D["minutes"].isna().sum()`

■ `D.shape[0] # Correct; D.shape[0] doesn't change after the code above runs.`

b) (3 pts)

```
t = lambda x: x.fillna(x.mean())
D["minutes"] = t(D["minutes"])
```

☐ `D["minutes"].mean()`

☐ `D.loc[D["traffic"] == "Moderate", "minutes"].mean()`

☐ `D.groupby("traffic")["minutes"].mean()`

☐ None of the above.

c) (3 pts)

```
t = lambda x: x.fillna(x.mean())
D["minutes"] = D.groupby("traffic")["minutes"].transform(t)
```

☐ `D["minutes"].mean()`

☐ `D.loc[D["traffic"] == "Moderate", "minutes"].mean()`

☐ `D.groupby("traffic")["minutes"].mean()`

☐ None of the above.

d) (3 pts)

```
present = D.loc[D["minutes"].notna(), "minutes"]
n = D["minutes"].isna().sum()
D.loc[D["minutes"].isna(), "minutes"] = np.random.choice(present, n)
```

☐ `D["minutes"].mean()`

☐ `D.loc[D["traffic"] == "Moderate", "minutes"].mean()`

☐ `D.groupby("traffic")["minutes"].mean()`

☐ None of the above.

Question 9 (4 pts)

In the DataFrame `orders`, assume that:

- The median delivery time in `"Low"` traffic was 22 minutes.
- The median delivery time in `"Moderate"` traffic was 31 minutes.
- The median delivery time in `"High"` traffic was 42 minutes.
- The median delivery time in `"Very High"` traffic was 60 minutes.

Draw the following visualization, given the information you have.

```
orders.plot(kind="box", x="traffic", y="minutes")
```

While it's not possible to draw the visualization exactly, since you don't have all of the exact delivery times, it is possible to roughly sketch it, such that the information provided above is clearly visible. Some additional instructions:

- For simplicity, assume that across `"traffic"` categories, the variation in delivery times is roughly the same.
- Make sure your axes are labeled correctly. *(Hint: The y-axis should have numerical labels.)*

Question 10 (10 pts)

Consider the following corpus of two documents:

1. butter chicken $\underbrace{\text{naan naan ... naan}}_{k \text{ "naan"s total}}$
2. curry with naan

The total number of occurrences of "naan" in document 1 is k, so the total number of terms in document 1 is $k+2$, where $k \geq 3$ is some positive integer. Note that the two documents above are the only two documents in the corpus, meaning that there are 5 unique terms total in the corpus.

a) (4 pts) Given that the cosine similarity between the bag-of-words vector representations of the two documents is $\frac{5}{9}$, what is the value of k?

◯ 3 ◯ 4 ◯ 5 ◯ 6 ◯ 7 ◯ 8 ◯ 9 ◯ 10

b) (3 pts) In this part, assume we are using a base-2 logarithm.

(i) Which term in document 1 has the largest TF-IDF?
If there are ties, select them all.

☐ butter ☐ chicken ☐ naan ☐ curry ☐ with

(ii) Which term in document 2 has the largest TF-IDF?
If there are ties, select them all.

☐ butter ☐ chicken ☐ naan ☐ curry ☐ with

c) (3 pts) This part is independent of the previous parts.
In practice, you'll encounter lots of new metrics and formulas that you need to make sense of on the job. For instance, the **Wolverine Score (WS)**, defined below, is an alternative to the TF-IDF that also tries to quantify the importance of a term t to a document d, given a corpus of documents $d_1, d_2, ..., d_n$.

$$\text{WS}(t,d) = \left(\frac{\text{total \# of terms in } d}{\text{\# of occurrences of } t \text{ in } d}\right) \cdot \left(\frac{\sum_{i=1}^{n} \text{\# of occurrences of } t \text{ in } d_i}{\sum_{i=1}^{n} \text{total \# of terms in } d_i}\right)$$

Fill in the blank to complete the statement below.
"If ____, then term t is more common in document d than it is across the entire corpus, and so t is likely an important term in d."
What goes in the blank?

◯ $\text{WS}(t,d) \leq 0$
◯ $\text{WS}(t,d) \geq 0$
◯ $\text{WS}(t,d) \leq 1$
◯ $\text{WS}(t,d) \geq 1$
◯ $\text{WS}(t,d) \geq \frac{1}{n}$, where n is the number of documents in the corpus
◯ $\text{WS}(t,d) \leq \frac{1}{n}$, where n is the number of documents in the corpus

uniqname: _____

Question 11 (9 pts)

The HTML document below contains the items on Wolverine Flavors Express' menu. We've only shown three menu items, but there are many more, as indicated by the ellipses ...

```
<html>
    <head>
        <title>Wolverine Flavors Express: A2's Favorite Indian Spot</title>
    </head>
    <body>
        <h1>Wolverine Flavors Express Menu</h1>
        <div class="meta">Last Updated February 25, 2025</div>
        <div class="menu-item" data-price="14.99" data-calories="650">
            <h2>Butter Chicken</h2>
            <p>Tender chicken in creamy tomato sauce - $14.99</p>
        </div>
        <div class="menu-item" data-price="12.99" data-calories="480">
            <h2>Chana Masala</h2>
            <p>Spiced chickpea curry - $12.99 ($11.99 on Tuesday)</p>
        </div>
        ...
        <div class="menu-item" data-price="21.99" data-calories="1050">
            <h2>Chicken 65 Biryani</h2>
            <p>Spicy chicken marinated with rice - $21.99 (special)</p>
        </div>
    </body>
</html>
```

Suppose we define `soup` to be a BeautifulSoup object that is instantiated using the HTML document above. Fill in the blanks below so that `low_cals` contains the names of the menu items with less than 500 calories.

```
low_cals = []
for x in __(i)__:
    if __(ii)__:
        low_cals.append(__(iii)__)
```

(i): _____

(ii): _____

(iii): _____

Question 12 (6 pts)

When we originally downloaded the `orders` dataset from the internet, some of the values in the `"minutes"` columns were formatted incorrectly as strings with two decimals in them. To clean the data, we implemented the function `clean_minutes`, which takes in an invalid minutes value as a string and returns a correctly formatted minutes value as a float. Example behavior of `clean_minutes` is given below.

```
>>> clean_minutes("8.334.108")
83.34
>>> clean_minutes("5.123.999")
51.23
>>> clean_minutes("12.091.552")
120.91
>>> clean_minutes("525.345.262")
5253.45
```

As a helper function, we implemented the function `split_pieces`; example behavior is given below.

```
>>> split_pieces("8.334.108")
("8", "334")
>>> split_pieces("12.091.552")
("12", "091")
```

Fill in the blanks to complete the implementations of the functions `split_pieces` and `clean_minutes` so that they behave as described above. Assume that the inputs to both functions are formatted like in the examples above, i.e. that there are exactly 3 digits between the middle two decimals.

```
def split_pieces(s):
    return re.findall(__(i)__, s)[0]

def clean_minutes(s):
    pieces = split_pieces(s)
    return __(ii)__
```

(i): `r"` [ ] `"`

(ii): [ ]

uniqname: _____

Question 13 (9 pts)

Suppose we'd like to fit a constant model, $H(x_i) = h$, to predict the number of minutes a delivery will take. To find the optimal constant prediction, h^*, we decide to use the $\alpha\beta$**-balanced** loss function, defined below:

$$L_{\alpha\beta}(y_i, h) = (\alpha y_i - \beta h)^2$$

where α and β are both constants, and $\beta \neq 0$.

Below, assume $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ is the mean number of minutes a delivery took.

a) (2 pts) Suppose $\alpha = 3$ and $\beta = 3$, so $L_{\alpha\beta}(y_i, h) = (3y_i - 3h)^2$.
What is the optimal constant prediction, h^*, that minimizes average $\alpha\beta$-balanced loss?

◯ $\bar{y}$ ◯ $\frac{1}{2}\bar{y}$ ◯ $2\bar{y}$ ◯ $\frac{1}{3}\bar{y}$ ◯ $3\bar{y}$ ◯ $\frac{1}{6}\bar{y}$ ◯ $6\bar{y}$

b) (2 pts) Suppose $\alpha = 6$ and $\beta = 3$, so $L_{\alpha\beta}(y_i, h) = (6y_i - 3h)^2$.
What is the optimal constant prediction, h^*, that minimizes average $\alpha\beta$-balanced loss?

◯ $\bar{y}$ ◯ $\frac{1}{2}\bar{y}$ ◯ $2\bar{y}$ ◯ $\frac{1}{3}\bar{y}$ ◯ $3\bar{y}$ ◯ $\frac{1}{6}\bar{y}$ ◯ $6\bar{y}$

c) (5 pts) Find the optimal constant prediction, h^*, that minimizes average $\alpha\beta$-balanced loss **in general**, for any valid choice of α and β. Show your work, and $\boxed{\text{box}}$ your final answer, which should be an expression involving $\bar{y}$, α, β, and/or other constants.

Question 14 (6 pts)

Suppose we'd like to predict the number of minutes a delivery will take (y) as a function of distance (x). To do so, we look to our dataset of n deliveries, $(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)$, and fit two simple linear models:

- $F(x_i) = a_0 + a_1 x_i$, where:

$$a_1 = r\frac{\sigma_y}{\sigma_x}, \qquad a_0 = \bar{y} - r\frac{\sigma_y}{\sigma_x}\bar{x}$$

  Here, r is the correlation coefficient between x and y, $\bar{x}$ and $\bar{y}$ are the means of x and y, respectively, and σ_x and σ_y are the standard deviations of x and y, respectively.
- $G(x_i) = b_0 + b_1 x_i$, where b_0 and b_1 are chosen such that the line $G(x_i) = b_0 + b_1 x_i$ minimizes **mean absolute error** on the dataset. Assume that $b_0 \neq a_0$ and $b_1 \neq a_1$, i.e. that F and G are two different lines. Also assume that no line minimizes mean absolute error on the dataset, i.e. the values of b_0 and b_1 are unique.

a) (2 pts) Fill in the ???:

$$\sum_{i=1}^{n}(y_i - G(x_i))^2 \quad \boxed{???} \quad \sum_{i=1}^{n}(y_i - F(x_i))^2$$

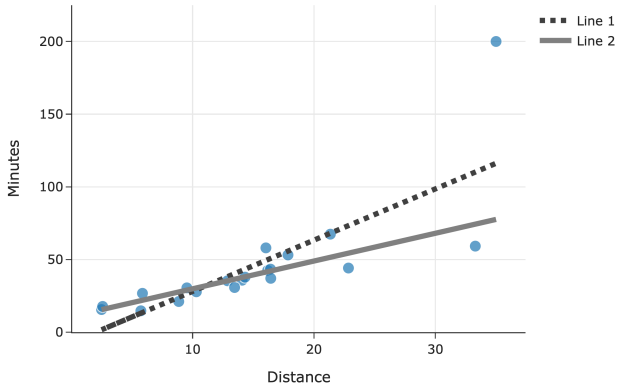
◯ $\geq$ ◯ $>$ ◯ $=$ ◯ $<$ ◯ $\leq$ ◯ Impossible to tell

b) (2 pts) Fill in the ???:

$$\left(\sum_{i=1}^{n}\left|y_i - G(x_i)\right|\right)^2 \quad \boxed{???} \quad \left(\sum_{i=1}^{n}\left|y_i - F(x_i)\right|\right)^2$$

◯ $\geq$ ◯ $>$ ◯ $=$ ◯ $<$ ◯ $\leq$ ◯ Impossible to tell

c) (2 pts) Below, we've drawn both lines, F, and G, along with a scatter plot of the original n deliveries.



Which line corresponds to line F?

◯ Line 1 ◯ Line 2

Question 15 (0.5 pts; extra credit)

In Question 7, assuming that there are 100 unique drivers, what is the true, theoretical probability that `"WOLVAA01"` wins a free gift card? Show your work, and box your final answer, which should be a **fraction**.

Don't spend time on this question if you haven't attempted all other questions, as it is purely for (very little) extra credit.

Make sure you've written your uniqname 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 EECS 398 below :)