Data C100/200- Midterm

Summer 2025

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Name and SID of the person on your left: _____

Name and SID of the person on your right: _____

Exam Room:_____ Seat Number: _____

Instructions:

This exam consists of **38 points** spread out over **4 questions** and the **Honor Code certification**. The exam must be completed in **110 minutes** unless you have accommodations supported by a DSP letter. Note that you should:

- Each true/false question and multiple choice question has **exactly one** correct answer. Please **fully** shade in the circle to mark your answer.
- Blank answers and incorrect answers are graded identically, so it's in your best interest to answer every question.
- For all math questions, **please simplify your answer**. Please also **show your work** if a large box is provided.
- For all coding questions, you may use commas and/or one or more function calls in each blank.
- **You MUST write your Student ID number at the top of each page.**
- You should not use a calculator, scratch paper, or notes you own other than the reference sheets distributed at the beginning of the exam.

For all Python questions, you may assume `Pandas` has been imported as `pd`, `NumPy` as `np`, the Python RegEx library as `re`, `matplotlib.pyplot` as `plt`, and `seaborn` as `sns`.

Honor Code [1 Pt]:

As a member of the UC Berkeley community, I act with honesty, integrity, and respect for others. I am the person whose name is on the exam, and I completed this exam in accordance with the Honor Code.

Signature: _____

This exam was administered on a computer via PrairieLearn. The PrairieLearn software allows exam questions to differ across students. This PDF shows one possible instantiation of the exam. If you took this exam in Summer 2025, the questions you saw at the testing center may have been slightly different than the questions in this PDF.

1 Can pandas eat fruit? [15 Pts]

Oski visits several grocery store chains across Berkeley. At each store location, he records the price and quantity of every available fruit. He stores this information in a `DataFrame` called `fruit`. The columns of `fruit` are described below:

- `item`: Name of each fruit (type = `str`).
  Note: Each `item` only appears once for every store location.
- `chain`: Name of the grocery chain (type = `str`).
- `address`: Address of a specific store location, formatted as: `"StreetNumber StreetName"` (type = `str`).
  Note: Grocery chain names can appear more than once since grocery chains may have multiple locations in Berkeley.
- `price`: Price of one unit of fruit in dollars (type = `np.float64`).
- `inventory`: Current number of units of fruit in the inventory of the visited store (type = `np.int64`).
- `ranking`: Oski's personal rating of the fruit using the mapping of `1` (Great), `2` (Good), and `3` (Mediocre) (type = `np.int64`).

The first six rows of `fruit` are shown below:

	item	chain	address	price	inventory	ranking
0	apple	Trader Joe's	1885 University	1.00	75	1
1	apple	Safeway	1444 Shattuck	1.50	50	1
2	orange	Whole Foods	3000 Telegraph	0.50	230	2
3	strawberry	Berkeley Bowl	2020 Oregon	4.00	45	2
4	banana	Trader Joe's	1885 University	1.00	300	3
5	banana	Trader Joe's	5727 College	0.75	100	3

(a) [1 Pt] What is the granularity of `fruit`? Answer in no more than one sentence.

Solution: Each unique combination of `address` and `item`.

(b) [1 Pt] For each column of `fruit`, choose the most representative variable type.

(i) [0.5 Pts] The most representative variable type for the column `item` is:

- ◯ Qualitative ordinal
- ◯ **Qualitative nominal**
- ◯ Quantitative

(ii) [0 .5 Pts] The most representative variable type for the column `ranking` is:

- ◯ **Qualitative ordinal**
- ◯ Qualitative nominal
- ◯ Quantitative

(c) [2 Pts] Select all options below that return the name of the `item` that appears in the most rows in `fruit`. The return value must be a single `String`. You can assume that no two `items` appear the same number of times in `fruit`.

- ☐ `fruit["item"].value_counts().index[0]`
- ☐ `fruit.value_counts().iloc[0]`
- ☐ `fruit["item"].value_counts().loc[0, 0]`
- ☐ `fruit.groupby("item").count().iloc[0, :]`

(d) [2 Pts] To prepare for a Data100 potluck, Oski wants to buy 150 bananas.

1. He first filters the rows to those that have at least 150 bananas in their inventory.
2. He then sorts the rows so that the visited store with the highest-priced bananas are at the top of the returned `DataFrame`. You can assume that all banana prices are unique across the visited stores.

Fill in the blanks to carry out the two steps above.

```
fruit[_____(i)_____]._____(ii)_____(_____(iii)_____,ascending=_____(iv)_____)
```

(i) [1 Pt] Fill in blank `(i)`:

Solution:
```
(fruit["item"] == "banana") & (fruit["inventory"] >= 150)
```

(ii) [0.5 Pts] Fill in blank `(ii)`:

Solution: `sort_values`

(iii) [0.25 Pts] Fill in blank `(iii)`:

Solution: `"price"`

(iv) [0.25 Pts] Fill in blank `(iii)`:

Solution: `False`

(e) [3.5 Pts] Oski asks a research question: Which grocery chain has the most valuable fruit inventory across all of its locations and all of its available fruits?

The value of the inventory of a specific fruit can be calculated using the following formula:

```
inventory_value = fruit_price * number_of_fruit_in_inventory
```

	item_inventory_value	
	sum	max
Berkeley Bowl	255.0	180.0
Safeway	670.0	400.0
Trader Joe's	1195.0	390.0
Whole Foods	587.5	472.5

Fill in the blanks below to create `metrics` exactly as it appears above.

1. There should be one row of `metrics` for each unique grocery store chain.
2. The first column of `metrics` indicates the total value of the fruit inventory in all visited stores of each grocery chain.
3. The second column of `metrics` contains the inventory value of the **single** store with the highest inventory value. For example, consider the row corresponding to Trader Joe's. If the Trader Joe's closest to campus has the lowest inventory value of all Trader Joe's locations in Berkeley, then the second column would contain that single store's inventory value. You may assume that no two store locations have the same total inventory value.

```
# Step 1:  First, we make a copy of the `fruit` DataFrame
   called `q1e`.

q1e = fruit.copy()

# Step 2:  Add a column called "item_inventory_value" to `q1e`
   containing the total value of each fruit in each store
   location.

q1e["item_inventory_value"] = _____(i)_____

# Step 3:  Create a DataFrame called `metrics` that looks
   exactly like the image above.  Note:  The .agg() method
   automatically sorts the index of the resulting DataFrame

metrics = q1e.groupby(_____(ii)_____).agg(_____(iii)_____)
```

(i) [1 Pt] Fill in blank `(i)`:

Solution: `q1e["inventory"] * q1e["price"]`

(ii) [1 Pt] Fill in blank `(ii)`:

Solution: `"chain"`

(iii) [1.5 Pts] Fill in blank `(iii)`:

Solution: `{'item_inventory_value' : ('sum', 'min')}`

(f) [1.5 Pts] Oski is asked to host a fruit-eating event for Cal Day. He obtains a new `DataFrame` called `reachable` that reports whether each store location is easily accessible by public transit. The columns of `reachable` are described below:

- `chain`: Name of the grocery chain (type = `str`).
- `address`: Address of the visited store, formatted as: `"StreetNumber StreetName"` (type = `str`).
  Note: Store names can appear more than once since grocery chains may have multiple locations in Berkeley.
- `is_reachable`: Boolean indicating whether the store location is accessible by public transit (type = `bool`).

The first 5 rows of `reachable` are provided below:

	chain	address	is_reachable
0	Trader Joe's	1885 University	True
1	Safeway	1444 Shattuck	False
2	Whole Foods	3000 Telegraph	True
3	Berkeley Bowl	2020 Oregon	False
4	Trader Joe's	5727 College	False

Write **one line** of code to create a new `DataFrame` called `q1f` that is **identical** to `fruit`, **except** for one additional column called `is_reachable` that indicates whether the specific store the fruit is located at is reachable by public transit.

The first five rows `q1f` are below:

	item	chain	address	price	inventory	ranking	is_reachable
0	apple	Trader Joe's	1885 University	1.0	75	1	True
1	apple	Safeway	1444 Shattuck	1.5	50	1	False
2	orange	Whole Foods	3000 Telegraph	0.5	230	2	True
3	strawberry	Berkeley Bowl	2020 Oregon	4.0	45	2	False
4	banana	Trader Joe's	1885 University	1.0	300	3	True

Note: Carefully inspect the expected columns in `q1f` to ensure your outputted `DataFrame` will **match exactly**. You may find it helpful to recall that no two stores in `fruit` or `reachable` and share the same address.

Solution:
```
q1f = fruit.merge(reachable[["address", "is_reachable"]],
left_on = "address", right_on = "address")
```
Note: There are many correct ways to solve this!

(g) [2 Pts] Help Oski determine, for each unique combination of `item` and `is_reachable`, the average amount of each fruit across all store locations in the `q1f DataFrame`. This information will be stored as a `DataFrame` called `q1g`.

To guide you, the first 5 rows of `q1f` and the entire `q1g DataFrame` are provided below.

`q1f`:

	item	chain	address	price	inventory	ranking	is_reachable
0	apple	Trader Joe's	1885 University	1.0	75	1	True
1	apple	Safeway	1444 Shattuck	1.5	50	1	False
2	orange	Whole Foods	3000 Telegraph	0.5	230	2	True
3	strawberry	Berkeley Bowl	2020 Oregon	4.0	45	2	False
4	banana	Trader Joe's	1885 University	1.0	300	3	True

`q1g`:

item	apple	banana	grape	orange	strawberry
is_reachable					
False	95.0	110.0	65.0	0.0	62.5
True	75.0	300.0	0.0	230.0	105.0

```
q1g = q1f.pivot_table(_____(i)_____)
```

Fill in blank `i` to create `q1g`. If a particular combination of fruit and reachability does not exist, the average amount in `q1g` should have a value of 0.

Solution:
```
q1h = q1g.pivot_table(values="inventory",
index="is_reachable", columns="item",
aggfunc="mean", fill_value=0)
```

(h) [2 Pts] Several weeks after Cal Day, Oski conducts a survey to gauge how Cal Day attendees felt about the fruit-eating event. The event was attended by both current UC Berkeley students and admitted applicants who returned home after Cal Day ended. Oski walks to Memorial Glade and asks each person sitting on the glade if they attended the fruit-eating event. If they say yes, Oski asks them to fill out the survey.

(i) [1 Pts] Is Oski's sampling frame the exact same as Oski's population of interest? Explain in at most one sentence.

Solution: No, it is not since not all attendees are necessarily at Memorial Glade.

(ii) [1 Pts] Oski has the email addresses of all 1,000 attendees of the fruit-eating event. Of the attendees, 500 went to high school in Southern California, 400 in Northern California, and 100 outside of California. Oski collects a stratified random sample of 100 email addresses of the attendees. Oski stratifies on high school location. How many of the sampled email addresses are from attendees who went to high school anywhere in California?

Solution: $(500 + 400)/1000 * 100 = 90$

2 Data 10{2} Potluck today! Data 102 Potluck tomorrow. [5 Pts]

Cristina's dorm hosts a giant potluck every semester to celebrate their hard work! Cristina compiles a spreadsheet that lists each attendee, their dish, how many servings in their dish, what type of course it is (M - Main, S - Side, A - Appetizer, D - Dessert), and their phone number.

She transforms the spreadsheet into a multi-line string:

```
data = """
- Attendee: Cristina, Dish: 'Carrot Cake Muffins',
Servings: [24], Course: D, PN: 510-555-5555
- Attendee: Ben, Dish: 'Pesto Pasta',
Servings: [15], Course: M, PN: 510-777-7777
- Attendee: Ella, Dish: 'greek salad',
Servings: [15], Course: S, PN: 415-555-5555
- Attendee: Sara, Dish: 'chips and guac',
Servings: [23], Course: A, PN: 949-555-5555
- Attendee: Rohan, Dish: 'Chicken',
Servings: [18], Course: M, PN: 415-111-1111
"""
```

Note: For all parts, you only need to consider the examples in the string above. You may assume that these examples are representative of all possible edge cases. Attendees' names and dishes can only consist of uppercase letters, lowercase letters, and spaces. Servings will always be numbers. Phone numbers are always in the format ###-###-####, where is a digit 0 through 9.

(a) [3 Pts] Cristina wants to compile a list of each type of course along with the total number of servings for that course. She writes the code below:

```
pattern = r"\[(____(i)____)\],\sCourse:\s(____(ii)____)"
re.findall(pattern, data)
```

elect all options for blanks (i) and (ii) that will result the in the list below.

```
[('24', 'D'), ('15', 'M'),  ('15', 'S'), ('23', 'A'), ('18', 'M')].
```

(i) [1.5 Pts] Select successful patterns for `(i)`:

- ☐ `[0-9]?`
- ☐ `\W+`
- ☐ `[^\s]*`

(ii) [1.5 Pts] Select successful patterns for `(ii)`:

- ☐ `D|M|A|S`
- ☐ `\w{,5}`
- ☐ `[SAMD]`

(b) [2 Pts] Cristina reformats the information into a `DataFrame` called `dishes`. All entries in `dishes` are of the `str` datatype. The first 5 rows of `dishes` are below:

	name	dish	servings	course	phone number
0	Cristina	Carrot Cake Muffins	24	D	510-555-5555
1	Ben	Pesto Pasta	15	M	510-777-7777
2	Ella	greek salad	15	S	415-555-5555
3	Sara	chips and guac	23	A	949-555-5555
4	Rohan	Chicken	18	M	415-111-1111

Cristina wants to extract only rows with phone numbers that begin with `510` (Berkeley's area code). Which of the following lines of code successfully returns the `DataFrame` of interest? Select all that apply.

- ☐ `dishes[dishes["phone number"].str.startswith("510")]`
- ☐ `dishes[dishes["phone number"].str[:3] == "510"]`
- ☐ `dishes[dishes["phone number"].str.match("^510$")]`
- ☐ `dishes[dishes["phone number"].str.split("-").str[0] == "510"]`

3 Bacteria Bonanza! [9.5 Pts]

Justin and Hannah are researchers studying an experimental antibiotic. They want to investigate how the bacteria responds to various quantities of this antibiotic, so they create a `DataFrame` called `experiments`. `experiments` contains one row for each of 10,000 experiments that they conduct. The first column contains the amount of antibiotic added to each petri dish in units of µg/mL, and the second column contains the number of bacterial colonies present in the petri dish after 12 hours.

The first five rows of `experiments` are shown below.

	Antibiotic	Colonies
0	7.58	292
1	1.01	891
2	17.17	117
3	27.27	68
4	2.02	726

(a) [2 Pts] Select all of the following visualizations that would be appropriate for investigating the correlation of the amount of antibiotics added (`Antibiotic`) and the number of colonies present after 12 hours (`Colonies`).

- ☐ One histogram
- ☐ Two overlaid histograms
- ☐ Two side-by-side violin plots
- ☐ **One contour plot**

(b) [1.5 Pts] Fill in the `matplotlib` code below to help Justin produce a scatterplot showing the relationship of the number of colonies and amount of antibiotic.

```
plt._____(i)_____(_____(ii)____, ____(iii)____)
```

(i) [0.5 Pts] Fill in blank `(i)`:

Solution: `scatter`

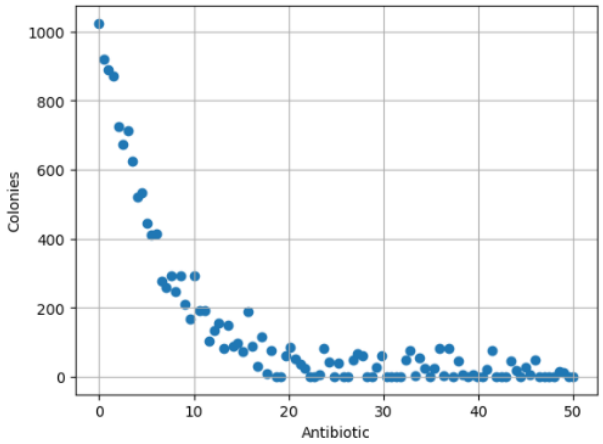
(ii) [0.25 Pts] Fill in blank `(ii)`:

Solution: `x = experiments['Antibiotic']`

(iii) [0.25 Pts] Fill in blank `(iii)`:

Solution: `y = experiments['Colonies']:`

(c) [2 Pts] Justin observes a non-linear relationship between `Antibiotic` and `Colonies` in the plot below. Select all of the following transformations that could potentially help to linearize the relationship between the `Antibiotic` (X) and `Colonies` (Y).



☐ X^2

☐ $\log(Y)$

☐ $\sqrt{X}$

☐ 10^X

☐ $Y/10$

(d) [2 Pts] Justin discovers that a data thief has stolen 10% of the collected Colonies values and replaced them with a filler value of -100. Justin suggests that dropping all of the the rows that contain -100 for their Colonies values **will not** introduce selection bias. In which of the following scenarios is Justin correct? Select all that apply.

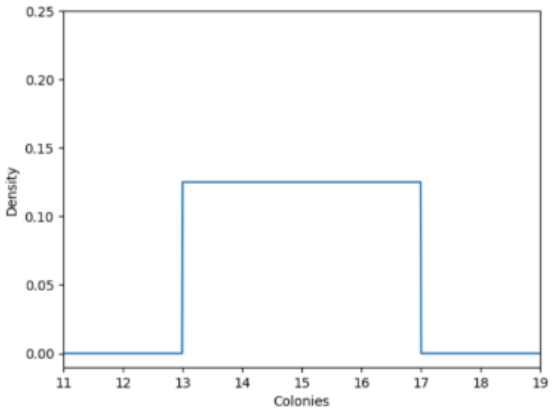
- ☐ **The data thief randomly chose 10% of experiments. Each experiment had the same chance of being selected, and each experiment could only be selected once. Chosen experiments' values were then were changed to -100.**
- ☐ The data thief chose the first 10% of experiments conducted by Justin and Hannah, and then changed their values to -100.
- ☐ **The data thief conducted a stratified random sample of all the experiments, stratifying based on the time of day the experiment was conducted. The data thief randomly selected 10% of experiments in each stratum and changed them to -100.**
- ☐ **The data thief put the ID of each experiment on a ping pong ball, put the ping pong balls in a giant box, thoroughly mixed the ping pong balls in the box, and then randomly selected 10% of the ping pong balls without replacement. The results of the experiments corresponding to the selected ping pong balls were changed to -100.**

(e) [1 Pt] Hannah wants to plot a KDE of the bacterial colony counts (`Colonies`) across all of the experiments. Hannah proposes her own custom kernel, defined below:

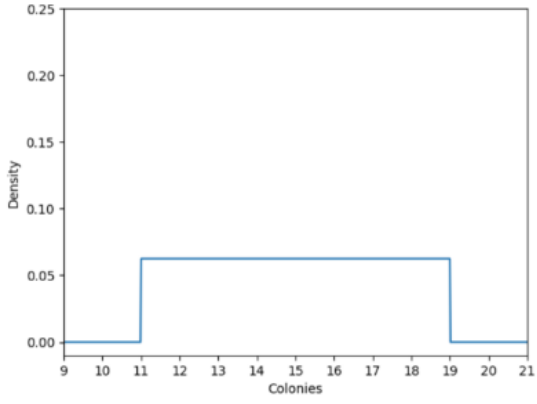
$$K_a(x, z) = \begin{cases} \frac{1}{2a}, & \text{if } -\frac{a}{2} \leq (x - z) \leq \frac{a}{2} \\ 0, & \text{otherwise} \end{cases}$$

Select the graph that visualizes Hannah's proposed kernel for a single data point with 15 colonies and $\alpha = 4$.

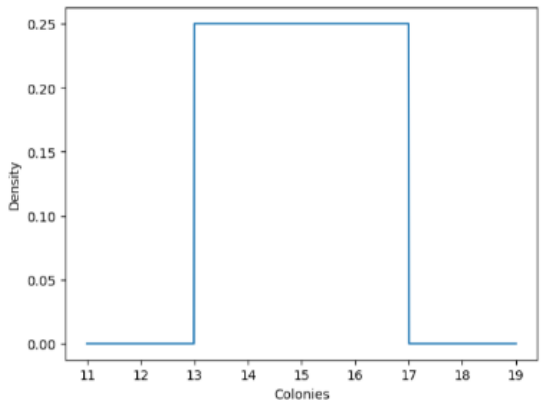
○



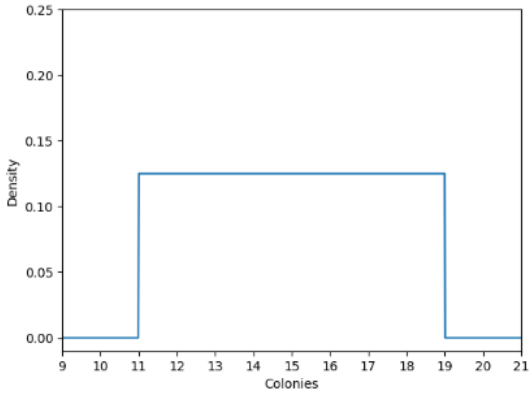
○



◯



◯



(f) [1 Pt] Hannah's proposed kernel function would be a valid density function if multiplied by:

◯ 8.

◯ 4.

◯ **2.**

◯ 1. In other words, Hannah's proposed kernel function is already a valid density function.

4 Michael's Modeling [8.5 Pts]

When Michael eats at his favorite restaurant, he notices that people tip differently based on the weather. He wonders to himself: "How well can I predict tip size based on just the outside temperature?". To help him investigate, the restaurant provides Michael with the following data from **N** customers:

- `temperature`: The temperature in Fahrenheit when the customer visited the restaurant. (type = `np.float64`)
- `tip` The tip the customer paid in dollars. You can assume all the tips are greater than \$1. (type = `np.float64`)

The dataset has no missing values.

(a) [1.5 Pts] Michael computes the correlation between `tip` and `temperature`. To Michael's surprise, the correlation is 0. Select the statements below that are true.

- ☐ A scatterplot of the values in the `tip` and `temperature` columns would NOT show a pattern. In other words, the points would look randomly scattered.
- ☐ If Michael fits the simple linear regression model $\widehat{\text{tip}} = \theta_0 + \theta_1 \times \text{temperature}$ using squared loss, the optimal $\hat{\theta}_0$ will always be unique and equal to 0.
- ☐ **If Michael fits the simple linear regression model $\widehat{\text{tip}} = \theta_0 + \theta_1 \times \text{temperature}$ using squared loss, the optimal $\hat{\theta}_1$ will always be unique and equal to 0.**

(b) Michael decides to fit the following constant model: $\widehat{\text{tip}} = \theta_0$. He tries a variety of loss functions.

(i) [1.5 Pts] Michael first considers using the mean absolute error (MAE) as the objective function. Select the statements below that are true.

- ☐ If Michael were to randomly shuffle the order of the rows in his data without replacement, it is possible for the minimum value of the MAE to change.
- ☐ There will always be one value of $\hat{\theta}_0$ that mimimizes the MAE.
- ☐ **Compared to the "mean 4th power error" loss function ($\frac{1}{N}\sum_{i=1}^{N}(\text{tip}_i - \theta_0)^4$), the MAE is more robust to outliers.**

(ii) [2 Pts] Michael then decides to use the following objective function: $\sum_{i=1}^{N}[3\theta_0 - \text{tip}_i * \log(\theta_0)]$ where log refers to the natural logarithm (i.e., ln). We will refer to this objective function as "the new objective (TNO)".

In the code cell below, complete the `tno_derivative()` function. `tno_derivative()` takes 2 inputs:

- `theta0`: A number between 0 and ∞. (type: `np.float64`)
- `tip`: A `Series` containing the values in the `tip` column of the original data. (type: `Series`)

The function should compute the value of $\frac{d}{d\theta}\sum_{i=1}^{N}[3\theta_0 - \text{tip}_i * \log(\theta_0)]$ in terms of `tip` and `theta0`.

Note: You must not use a `for` loop in your solution. You may find it helpful to recall that $\frac{d}{dx}\log(x) = \frac{1}{x}$.

```
def tno_derivative(theta0, tip):
    return ... # Your response here
```

Solution:

```
def tno_derivative(theta0, tip):
    return np.sum(3 - (tip/theta0))
```

(iii) [1.5 Pts] After some experimentation, Michael decides to go with yet another objective function (different from TNO). He calls this objective function "the ultimate objective" (TUO). Here is the expression for the **derivative** of TUO with respect to θ_0:

$$\frac{d}{d\theta_0}\text{TUO} = \sum_{i=1}^{N}\left(\frac{3}{\text{tip}_i} - \frac{1}{\theta_0}\right).$$

Select the option with the $\hat{\theta}_0$ that minimizes the TUO. You can assume that the TUO is convex and is defined for all positive θ_0. You may find it helpful to recall that $x^{-1} = \frac{1}{x}$

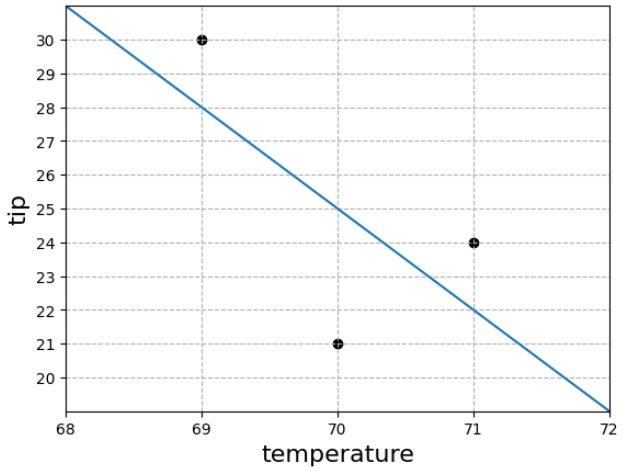
- ○ $\frac{1}{N}\sum_{i=1}^{N}\frac{\text{tip}_i}{3}$
- ○ $\frac{1}{N}\sum_{i=1}^{N}\log(\frac{\text{tip}_i}{3})$
- ○ $\frac{1}{N}\sum_{i=1}^{N}\frac{3}{\text{tip}_i}$
- ○ $N\left(\sum_{i=1}^{N}\frac{\text{tip}_i}{3}\right)^{-1}$
- ○ $N\left(\sum_{i=1}^{N}\log(\frac{\text{tip}_i}{3})\right)^{-1}$
- ○ $N\left(\sum_{i=1}^{N}\frac{3}{\textbf{tip}_i}\right)^{-1}$

(c) [2 Pts] Michael randomly samples 3 observations and fits a least squares linear regression model to the sampled observation:

$$\widehat{\text{tip}} = \theta_0 + \theta_1 \times \text{temperature}$$

The three data points and the fitted least squares line are plotted below. What is the MSE of the fitted model?

Note: You do not need to simplify algebra. For example, the expression $(5 + 10)^3 - \frac{2}{4}$ could be left as-is (i.e. `(5+10)^3 - 2/4` or `(5+10)**3 - 2/4`). PrairieLearn does not recognize decimals so all non-integers must be written as fractions.



Solution: `((30-28)^2 + (21-25)^2 + (24-22)^2)/3` = 8. Follow the MSE formula.

You are done with the midterm- Congratulations!

Draw your favorite DATA 100/200 memory so far!