Data C100, Midterm 1

Summer 2022

Name: ______________________________

Email: ______________________________@berkeley.edu

Student ID: ______________________________

Instructions:

This midterm exam consists of **80 points** spread out over **6 questions** and the Honor Code and must be completed in the **120 minute** time period ending at **8:00 PM**, unless you have accommodations supported by a DSP letter.

Note that some questions have circular bubbles to select a choice. This means that you should only **select one choice**. Other questions have boxes. This means you should **select all that apply**. Please shade in the box/circle to mark your answer.

Honor Code [4 Pts]:

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: ______________________________

1 Potpourri [10 Pts]

Choose whether each of the following statements is true or false, and then justify why that is the case. An example response for the statement "Pandas can read CSV files" is: "True. Pandas contains a `read_csv` function."

Any answer that is correct without any correct justification will receive no credit. Partial credit will be awarded to partially correct justification.

(a) [2 Pts] For non-convex empirical risk functions $R(\theta)$, it is possible that applying gradient descent to minimize $R(\theta)$ may converge to local minima or maxima.

> **Solution:** True. We may end up at a local minima or maxima, where the gradient is 0. A common mistake was mentioning that we cannot end up at a local maxima, but if the chosen learning rate is bad, we may "jump" to a local maxima where the gradient is 0 and we are stuck (technically, converged!).

(b) [2 Pts] Probability sampling is always more reflective of the population than quota sampling and convenience sampling.

> **Solution:** False. The probability sampling technique might be even less reflective of the population. For example, suppose the probability of choosing a particular selection is 1 and the probability of choosing anything else is 0. This may incur selection bias far worse than a reasonable quota sample, making it far less reflective of the population.

(c) [2 Pts] Scikit-Learn's `Ridge` class uses the normal equation to solve for $\hat{\theta}$.

> **Solution:** False. Ridge regression has an added regularization penalty, but the normal equation asserts that the residuals are orthogonal to the span of the design matrix. This condition need not hold now since Ridge regression does not output the optimal OLS $\hat{\theta}$ which minimizes the L-2 norm of the residuals.

(d) [2 Pts] The Tukey-Mosteller Bulge diagram may not work in linearizing a relationship, even if the shape of the curve matches.

> **Solution:** True. They are merely suggestions.

(e) [2 Pts] It is appropriate to use a bar chart to visualize a dataset of RateMyProfessor review scores on a scale from 1-5 that are in discrete 0.5 increments (i.e., you can only rate 1.0, 1.5, and so on up to 5.0 stars).

Solution: True. It is qualitative (ordinal) data, not quantitative data. Hence, a bar chart is appropriate! Note that a histogram may be used for discrete data, but this is not quantitative discrete data (such as age in years or number of days spent on holiday, for example).

2 Odd Odds [10 pts]

We will investigate a few events that we assume are completely random, for which the probabilities are given below. As a side note, these probabilities are not made up!

1. The odds of a person being born ambidextrous are around 1 in 100.
2. The odds of catching a shiny Pokemon in a Pokemon catch is 1 in 4,096.
3. The odds of shuffling a suit of cards into perfect numerical order randomly is 1 in 6 billion.

For all the following questions, round to the nearest 3 decimal places and provide a numerical answer (for example, 0.6666 rounds to 0.667 and 0.01 rounds to 0.010). Answers which are left as expressions will receive minimal or no credit.

Please feel free to use the Desmos scientific calculator. Note that the combination function ${}_nC_r \equiv \binom{n}{r}$ is in the `func` section of the calculator.

(a) [1 Pt] What is the probability of **not** being born ambidextrous (that is, monodextrous)?

Solution: Using the complement rule:

$$1 - \frac{1}{100} = 0.990$$

(b) [2 Pts] What is the probability that less than 3 people in Data 100 are ambidextrous? Assume there are 200 people enrolled in Data 100.

Solution: We can write the probability as a sum of the probabilities of k people being ambidextrous, where k is a number between 0 and 2. Specifically:

$$\binom{200}{0}(0.01)^0(0.99)^{200} + \binom{200}{1}(0.01)^1(0.99)^{199} + \binom{200}{2}(0.01)^2(0.99)^{198}$$

Simplifying, that is $0.99^{200} + 200(0.01)(0.99)^{199} + \frac{200(199)}{2}(0.01)^2(0.99)^{198} = 0.677$

Note that you could've used a calculator for any and all simplification!

(c) [2 Pts] Suppose we held a census of the world population and had them each randomly shuffle a suit of cards. What is the probability that at least 1 randomly shuffle the suit into perfect numerical order, assuming there are 8 billion people in the world?

Solution: We will use the complement rule by finding the probability nobody shuffles the suit into perfect numerical order:

$$1-(1-\frac{1}{6\cdot 10^9})^{8\cdot 10^9}=0.736$$

(d) [2 Pts] What is the probability of catching exactly 1 shiny Pokemon across 4,096 Pokemon catches if you are ambidextrous?

Solution: The events are independent. We can simply find the probability of catching 1 shiny Pokemon across 4,096 catches.

$$\binom{4096}{1}(\frac{1}{4096})^1(\frac{4095}{4096})^{4095}=0.368$$

(e) [3 Pts] Suppose we performed a simple random sample of 3 students from Data 100 to investigate for ourselves how rare it is to capture shiny Pokemon. What is the probability of selecting any **specific and unique** subset of 3 students (for example, Student A, B, and C) from the class? Assume that there are 200 students in Data 100.

- ◯ The probability of selecting the subset using SRS is the same for all subsets of students of size 3 is, and it is equal to $(\frac{1}{200})^3$.
- ◯ **The probability of selecting the subset using SRS is the same for all subsets of students of size 3 is, and it is equal to $\frac{1}{200}\cdot\frac{1}{199}\cdot\frac{1}{198}$.**
- ◯ The probability of selecting the subset using SRS varies based on what the subset of students of size 3 is, but it can be calculated individually for each subset.
- ◯ The probability of selecting the subset using SRS varies based on what the subset of students of size 3 is, and it cannot be calculated.

3 Misspellings Galoure [12 Pts]

Morgan, a Data 100 student, is trying to implement a simple fast spellchecker using Pandas and REGEX. You may assume that a *word* is any set of contiguous lowercase alphabet characters separated by non-word characters, either whitespace or punctuation character(s).

(a) [3 Pts] Which of the following patterns can **only** match strings where a "u" is after the letter "q" (such as queue or qubit but **not** qintar or qantas)?

Note that it need not match the entire string for it to count as a match (**partial matches count as matches!**). Select all that apply.

- [✓ ☐ `pat = r'qu'`]
- ☐ `pat = r'(qu)*'`
- ☐ `pat = r'qu*'`
- [✓ ☐ `pat = r'(qu)+'`]
- [✓ ☐ `pat = r'qu+'`]
- [✓ ☐ `pat = r'(qu){1,}'`]

(b) [4 Pts] To find common spelling errors, we will make use of two rules in the English language:

1. if there is a letter following "q", it must **always** be "u" (e.g., queue, qubit)
2. if there is a letter following "x", it must **never** be "s" (e.g., oxen, axes)

Given a list of lowercase words, any time that either of these two rules are **violated**, we wish to replace those words with "[redacted]". That is, if there is a character besides "u" immediately after a "q" or if "s" immediately follows "x", we wish to output "[redacted]". Fill in the pattern below to produce the following output.

```
>>> pattern = r'____________________'
>>> re.sub(pattern, '[redacted]', 'queue')
'queue'
>>> re.sub(pattern, '[redacted]', 'xq')
'xq'
>>> re.sub(pattern, '[redacted]', 'qebit')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'basqx')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'fuqxs')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'fuqs')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'puxs')
'[redacted]'
```

Solution:

```
>>> pattern = r'\w*(q[^u]|xs)\w*'
>>> re.sub(pattern, '[redacted]', 'queue')
'queue'
>>> re.sub(pattern, '[redacted]', 'qebit')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'basqx')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'xq')
'xq'
>>> re.sub(pattern, '[redacted]', 'fuqxs')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'fuqs')
'[redacted]'
>>> re.sub(pattern, '[redacted]', 'puxs')
'[redacted]'
```

(c) [3 Pts] Suppose Morgan decides to tackle all the spelling mistakes originating because of the wrong arrangement of "ie": beleive, feirce, dei, freind. We want to replace all lowercase instances of "ei" with "ie", except for after the lowercase letter "c".

Fill in the blanks below such that the desired outputs are shown.

Hint: You can access matched groups in `replace_with` using `\1`, `\2`, etc. For example, if we wanted to replace the matched text with the second captured group followed by the word "NOT", `replace_with = r'\2NOT'`.

```
>>> pattern = r'__________A___________'
>>> replace_with = r'_______B_____________'
>>> re.sub(pattern, replace_with, "this is a ceiling")
'this is a ceiling'
>>> re.sub(pattern, replace_with, "i love reimann sums!")
'i love riemann sums'
>>> re.sub(pattern, replace_with, "weird one, sufficeintly wrong")
'wierd one, sufficeintly wrong'
>>> re.sub(pattern, replace_with, "but i beleive this is right!")
'but i believe this is right!'
>>> re.sub(pattern, replace_with, "neice! ei!ei!")
'niece! ie!ie!'
```

Solution:

```
>>> pattern = r'([^c])ei'
>>> replace_with = r'\1ie'
>>> re.sub(pattern, replace_with, "this is a ceiling")
'this is a ceiling'
>>> re.sub(pattern, replace_with, "i love reimann sums!")
'i love riemann sums'
>>> re.sub(pattern, replace_with,
           "weird one, sufficeintly wrong")
'wierd one, sufficeintly wrong'
>>> re.sub(pattern, replace_with,
           "but i beleive this is right!")
'but i believe this is right!'
>>> re.sub(pattern, replace_with, "neice! ei!ei!")
'niece! ie!ie!'
```

(d) [2 Pts] Morgan wants to use the pattern and replace string from the previous subpart with Pandas string functions to efficiently remove **all** misspellings from our Pandas Series `spelling`. You may assume that you implemented the previous part correctly, where `pattern` and `replace_with` are defined for you.

Fill in the blanks below to yield the output shown as a Pandas Series, assuming `spelling` is a Pandas Series containing string values corresponding to the input values in the previous subpart.

```
>>> pattern = ... # assume this is correct
>>> replace_with = ... # assume this is correct
>>> spelling.______A__________(__________________B____________________)
0                 this is a ceiling
1              i love riemann sums!
2    wierd one, sufficeintly wrong
3     but i believe this is right!
4                     niece! ie!ie!
                        ...
dtype: object
```

Solution:

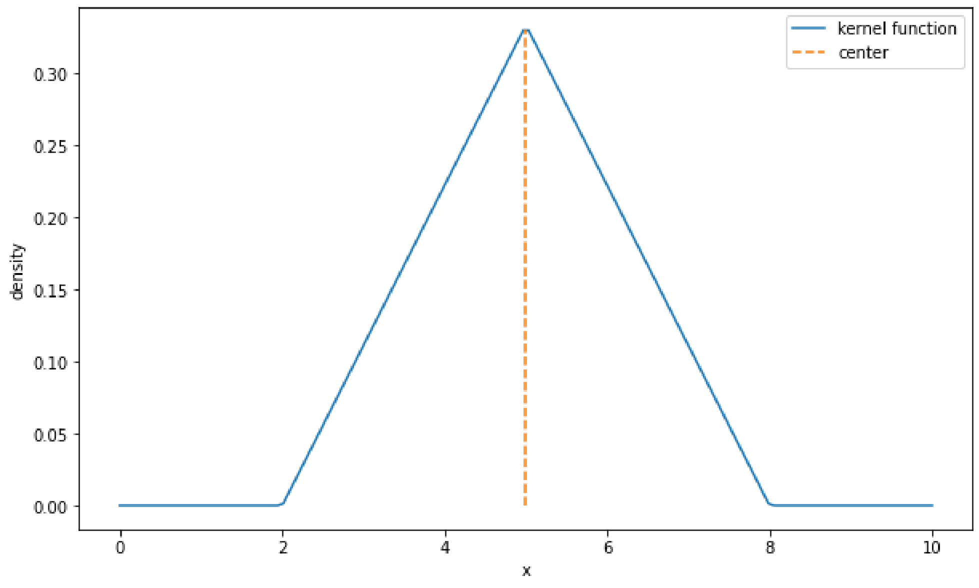
```
>>> pattern = ... # assume this is correct
>>> replace_with = ... # assume this is correct
>>> spelling.str.replace(pattern, replace_with)
0                 this is a ceiling
1              i love riemann sums!
2    wierd one, sufficeintly wrong
```

```
3     but i believe this is right!
4                    niece! ie!ie!
                   ...
dtype: object
```

4 Triangular Kernel? [10 Pts]

Brian finds that the boxcar and Gaussian kernels do not produce good visualizations for his dataset, so he decides to develop a new kind of (non-smooth) kernel called the triangular kernel! However, he is not sure how to finish the mathematical formulation for it! Luckily, he was able to generate a visualization that might help you help him.

A plot of the triangular kernel centered at $z = 5$ with bandwidth $\alpha = 3$ is shown below.



Below is the incomplete mathematical formulation of the triangular kernel, where β is a variable for which he has not yet assigned a value in terms of the function inputs x and z and bandwidth parameter α.

$$T_\alpha(x,z) = \begin{cases} \beta - \frac{1}{\alpha^2}|x-z| & |x-z| \le \alpha \\ 0 & |x-z| > \alpha \end{cases}$$

(a) [2 Pts] Which of the following is the value of β in terms of α and/or z?

Hint: the area of an isoceles triangle is the half the product of the base and height.

- [✓ $\frac{1}{\alpha}$]
- $\frac{1}{z}$
- α
- z
- $\frac{z}{\alpha}$

(b) [2 Pts] Which of the following is true regarding the relationship between the boxcar, Gaussian, and triangular kernel when implemented correctly?

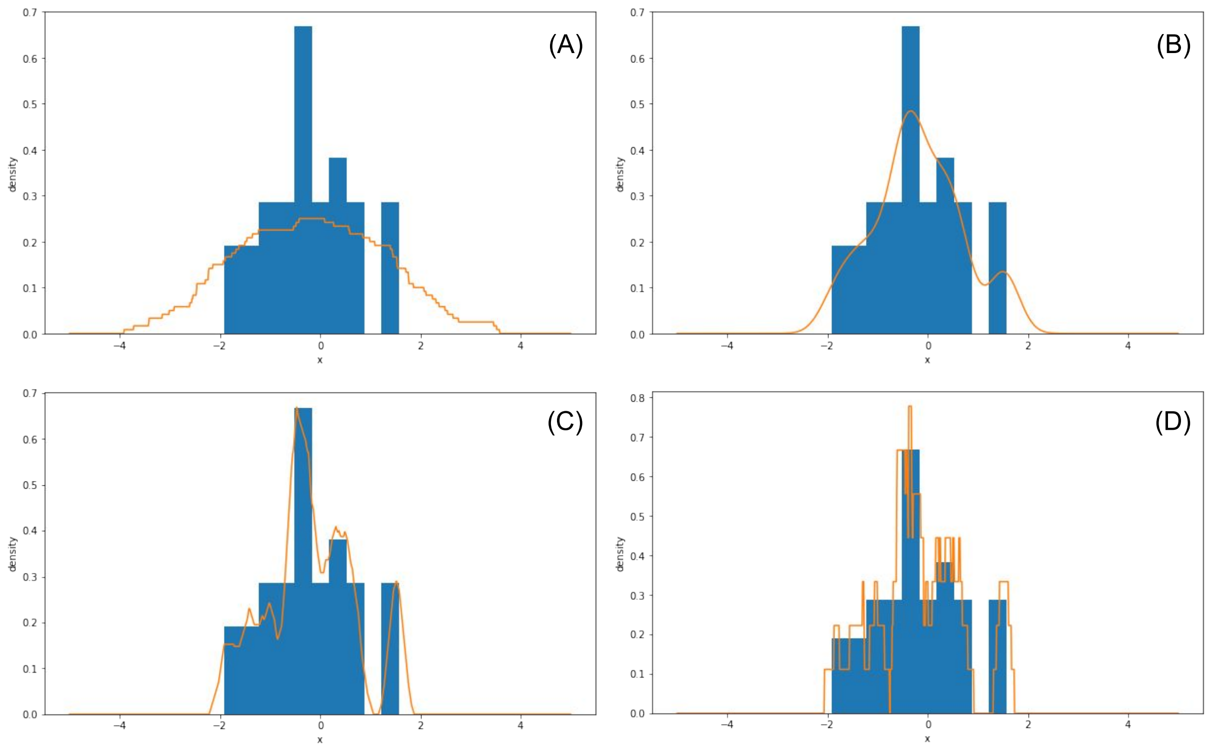
☐ **The density values generated by the triangular kernel using kernel density estimation are more likely to resemble density values generated by the Gaussian kernel than the boxcar kernel with appropriately matching bandwidth sizes.**

☐ The triangular kernel is likely to produce smooth visualizations as α decreases towards 0, but the boxcar kernel likely would not.

☐ The Gaussian kernel always produces smooth visualizations, but the triangular kernel will not produce smooth visualizations.

☐ **The area under all the kernel functions is 1.**

(c) [4 Pts] Match the following plots with the most likely kernel functions out of the following options: Gaussian, boxcar, triangular. Write the corresponding kernel function in the blank provided. All kernels have at least one corresponding plot shown below.

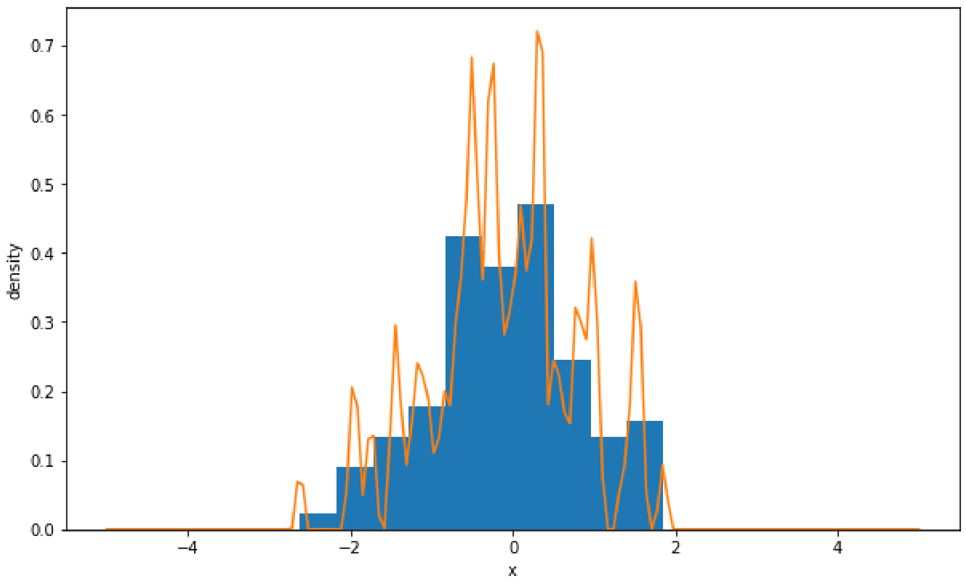


Solution: (A) and (D) are the boxcar kernel since there are visible ”square” edges. Gaussian kernel would yield smooth edges and the triangular kernel would yield triangular edges.

(C) is the triangular kernel since there are visible ”triangular” peaks.

(B) is the Gaussian kernel since the ”edges” are smooth, even when there are distinct modes. Note that there might be an argument for this being any of the other kernel functions with the appropriate bandwidth to have the plot being smooth, but since none of the other plots can be Gaussian (since they are all triangular or square) and the question states all kernels have a corresponding plot, this plot must be the Gaussian kernel.

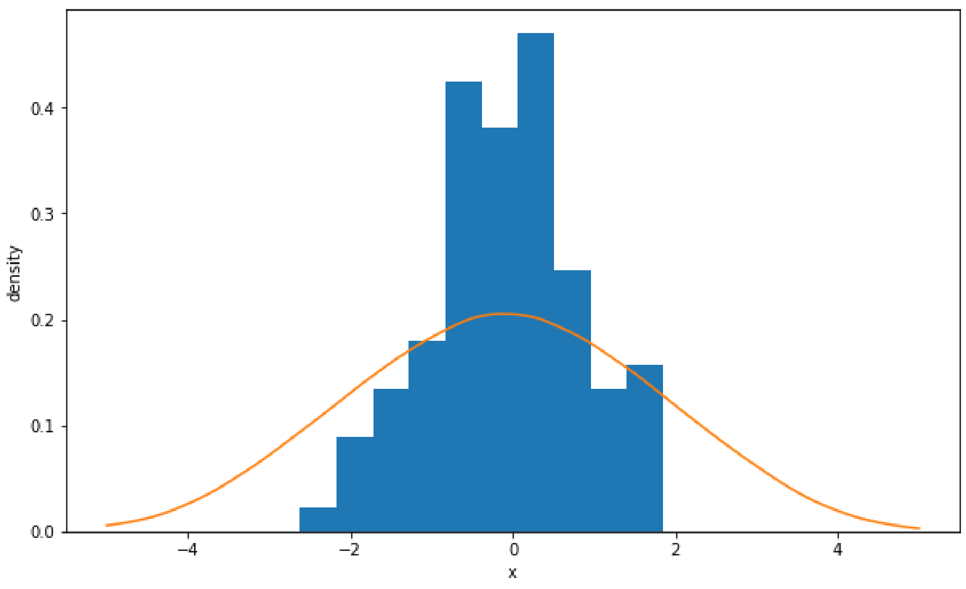
(d) [1 Pt] Brian generates a KDE plot using the triangular kernel. Which of the following is true about his choice of bandwidth parameter?



◯ The bandwidth parameter α is too large.

◯ **The bandwidth parameter α is too small.**

(e) [1 Pt] Trying to fix the bandwidth issue from the previous subpart, he ends up with the following figure. Which of the following is true now about his choice of bandwidth parameter?



◯ **The bandwidth parameter α is too large.**

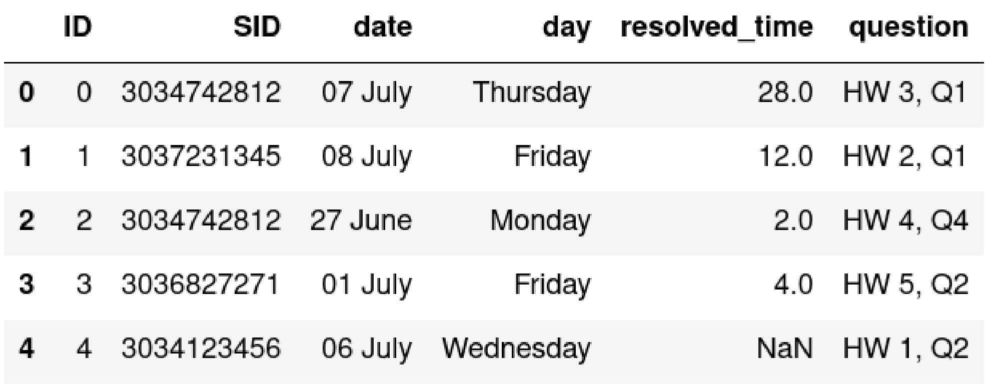
◯ The bandwidth parameter α is too small.

5 OH Analysis [17 Pts]

Throughout this question, we are dealing with Pandas DataFrame and Series objects. All code for this question, where applicable, must be written in Python. You may assume that Pandas has been imported as `pd`.

Anirudhan and Dominic wish to model the wait time (listed in the DataFrame as `resolved_time`) in the office hours queue at various points in time. To do this, Anirudhan has collected data from the office hours system. Before he jumps into modeling, he wishes to examine and analyze the data!

We will examine a DataFrame `oh` containing office hours tickets, the first 5 rows of which is shown below.



	ID	SID	date	day	resolved_time	question
0	0	3034742812	07 July	Thursday	28.0	HW 3, Q1
1	1	3037231345	08 July	Friday	12.0	HW 2, Q1
2	2	3034742812	27 June	Monday	2.0	HW 4, Q4
3	3	3036827271	01 July	Friday	4.0	HW 5, Q2
4	4	3034123456	06 July	Wednesday	NaN	HW 1, Q2

Note that some fields in this DataFrame are null. When the `resolved_time` is null, this means that the ticket was never resolved. The only fields **guaranteed** to not have null values are the `date`, the `day`, the `ID` (which is unique to every row, i.e., a primary key) and `SID` (student ID). Assume that the `question` is in the format "assignment name, question number".

(a) [2 Pts] Which of the following groupby expressions is **always** an equivalent expression to `oh['day'].value_counts()`?

○ `oh.groupby('day').count().sort_values(ascending = False)`

○ **`oh.groupby('day').size().sort_values(ascending = False)`**

○ `oh.groupby('day').agg(len).sort_values(ascending = False)`

○ `oh.groupby('day').agg(pd.value_counts)`

(b) [3 Pts] Write a Pandas expression that returns a Pandas Series showing the number of times each **unique** student has had an office hours ticket **resolved** (that is, when the `resolved_time` is not null).

Solution:

```
oh.groupby('SID')['resolved_time'].count()
```

(c) [3 Pts] We wish to find out the 5 students who made office hours tickets on the most number of unique dates. The output should be a Pandas Series. Fill in the blanks below to accomplish this.

```
def agg_func(s):
    return _________________________A___________________________

oh.groupby(__B__)['date'].agg(agg_func) \
                         .sort_values(ascending = False) \
                         .head(5)
```

Solution:

```
def agg_func(s):
    return len(s.unique())

oh.groupby('SID')['date'].agg(agg_func) \
                         .sort_values(ascending = False) \
                         .head(5)
```

(d) [6 Pts] Write a Pandas expression that returns a Pandas DataFrame that shows the the most common **assignment** (e.g., HW 1, HW 2, Lab 1) referenced in office hours tickets for **each unique day of the week** (Monday, Tuesday, ..., Friday) and assign it to the variable `result`.

The resulting column name and index order do not matter. You may define as many variables and functions as you want **within 8 lines of code** or add columns to the DataFrame. However, there must be no explicit for loops, list comprehensions, or generic function mapping (`map`/`apply`) in your answer. The output should resemble the following:

	most_common_assignment
day	
Friday	HW 2
Monday	HW 1
Thursday	HW 4
Tuesday	HW 2
Wednesday	HW 1

Solution:

```
def agg_fn(s):
    return (s.str.split(', ').str[0]
            .value_counts().index[0])
result = oh.groupby('day')[['question']].agg(agg_fn)
```

(e) [3 Pts] In preparation for modeling, suppose that we want to write our own one-hot encoding algorithm using Pandas to encode the day of the week... without using `get_dummies`. Fill in the blanks below to write a generic one-hot encoder function.

A sample output of `ohe(oh, 'day')` when implemented correctly is shown below.

day	Friday	Monday	Thursday	Tuesday	Wednesday
ID					
0	0	0	1	0	0
1	1	0	0	0	0
2	0	1	0	0	0
3	1	0	0	0	0
4	0	0	0	0	1
...	...	...	...	...	...

```
def ohe(input_df, ohe_column):
    """
    One-hot encodes a column of data called ohe_column
    in input_df and returns a Pandas DataFrame where
    the names of the columns correspond to the sorted,
    unique values in the data. The output must have the
    same number of rows as input_df.
    """
    return input_df.pivot_table(_______________________
                                _______________________
                                _______________________
                                _______________________
                                _______________________
                                fill_value = 0)
```

Solution:

```
def ohe(input_df, ohe_column):
    """
    One-hot encodes a column of data called ohe_column
    in input_df and returns a Pandas DataFrame where
    the names of the columns correspond to the sorted,
    unique values in the data.
    """
    return input_df.pivot_table(index = 'ID',
                                columns = ohe_column,
                                aggfunc = 'size',
                                fill_value = 0)
```

(f) [0 Pts] For 2 points of **extra credit**, unscramble the following two anagrams. Both correspond to cities or towns in the world.

```
ACBOEMIROT
NTLLKIAYOC
```

Please don't spend too much time on this - it's just for fun!

Hint: Both **c**ities/towns start with the same letter!

Solution:

```
>>> city1 = list('coimbatore')
>>> city2 = list('clonakilty')
>>> random.shuffle(city1)
>>> random.shuffle(city2)
>>> ''.join(city1).upper()
'ACBOEMIROT'
>>> ''.join(city2).upper()
'NTLLKIAYOC'
```

6 Office Hours Lifecycle [17 Pts]

Anirudhan and Dominic's modeling goal is to model the time it takes for a student to receive help ("wait time") on the office hours queue to display it on the oh.ds100.org website! Using data from historical tickets made by Data 100 students the office hours queue, they will construct linear models to predict the wait (or resolution) time!

(We already do this actually - check it out!)

(a) [2 Pts] Suppose that Dominic obtains the entire office hours dataset from Question 5 and decides to randomly sample with replacement 100 office hour tickets from each day of the week across the semester for modeling. Which of the following sampling techniques best describe this?

- ☐ Convenience sampling
- ☐ **Quota sampling**
- ☐ Linear sampling
- ☐ Correlated sampling
- ☐ Simple random sampling

(b) [2 Pts] After collecting the data, Dominic notices that there are a lot of missing values (around 10% of the data) for the wait time in minutes, concentrated mostly on due dates of the assignment.

Upon some investigation, he figures out that these corresponded to people whose tickets were never marked resolved. If we wish to use linear regression to predict the minutes of wait time, which option to resolve the issue of missing data is most reasonable?

- ○ Removing all data where the time taken to help is missing.
- ○ Imputing the missing values with the average time taken to help across the semester.
- ○ **Imputing the missing values with the average time taken to help for that day across the semester.**
- ○ Imputing the missing values with 0 minutes.
- ○ Imputing the missing values with 9999 minutes.

(c) [6 Pts] Dominic decides that there is a non-linear relationship between **time of day** (x) and **wait time in the office hours queue in standard units** (y). He applies a non-linear sinusoidal transformation, g, to the x values, and he fits the new variable $g(x)$ to y using a simple linear regression model.

Given the following information about the dataset containing 5 values, which of the following is true about optimizing this model's parameters on the given data to obtain $\hat{\theta} = [\hat{a}, \hat{b}]$?

SLR Dataset:

$g(x)$	y
3	-0.25
2	1.54
5	0.72
1	-0.94
5	-1.08

- ◯ There is no optimal analytical solution for SLR with the given transformations.
- ◯ There is an optimal analytical solution for SLR with the given transformations, but it is different from the ones in lecture.
- ◯ **There is an optimal analytical solution for SLR with the given transformations, and it is the same as the ones in lecture.**
- ◯ There are infinitely many optimal analytical solutions for SLR with the given transformations.

Justify your answer either by:

- explaining why it is impossible to derive an optimal solution, mathematically and/or intuitively
- deriving an optimal solution for $\hat{a}$ and $\hat{b}$ numerically

Note: Please do not try to use LaTeX! You may describe your process in words, numbers/calculations provided below, and standard mathematical symbols on the keyboard (e.g., /, *, +, -, sum, mean, std). Please don't spend too long perfecting the way the math looks - if needed, use words to describe your steps. We will understand your workflow! Answers without justification will receive minimal or no credit.

You may find the following calculations useful! Round to the 3 nearest decimal places if needed.

Possibly Helpful Calculations:

$$\overline{g(x)} = 3.2, \sigma_{g(x)} = 1.6$$

$$\sum_{i=1}^{5}(g(x_i) \cdot y_i) = -0.39$$

$$\sum_{i=1}^{5}((g(x_i) - 3.2) \cdot y_i) = -0.39$$

$$\sum_{i=1}^{5}(g(x_i) \cdot (y_i - 1)) = -16.39$$

Solution: We can apply the standard optimal solution for SLR. First, we calculate the correlation coefficient, which is the mean of the product of $g(x)$ and y in standard units:

$$\begin{aligned} r &= \frac{1}{5}\sum_{i=1}^{5}(\frac{g(x_i)-3.2}{1.6})y_i \\ &= \frac{1}{5(1.6)}\sum_{i=1}^{5}(g(x_i)-3.2)y_i \\ &= \frac{1}{8}\cdot -0.39 = -0.048 \end{aligned}$$

Then:

$$\hat{b} = \frac{r\sigma_y}{\sigma_{g(x)}} = \frac{r}{\sigma_{g(x)}} = \frac{-0.048}{1.6} = -0.030$$

and:

$$\hat{a} = \bar{y} - \hat{b}\bar{x} = 0.030(3.2) = 0.097$$

(d) [2 Pts] Regardless of your answer to the previous subpart, suppose that Anirudhan decides to use gradient descent for training this model and changes the function g, such that $g(x) = \sin(x)$. Derive the gradient update for the parameter b for the model $\hat{y} = a + bg(x) = a + b\sin(x)$ given the empirical risk function below in terms of the learning rate α, x_i, and y_i.

$$R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 = \frac{1}{n}\sum_{i=1}^{n}(y_i - (a + b\sin(x_i)))^2$$

- ○ $b^{(t+1)} = b^{(t)} + \frac{2\alpha}{n}\sum_{i=1}^{n}\sin(x_i)(y_i - \hat{y}_i)$
- ○ $b^{(t+1)} = b^{(t)} + \frac{2\alpha}{n}\sum_{i=1}^{n}\cos(x_i)(y_i - \hat{y}_i)$
- ○ $b^{(t+1)} = b^{(t)} + \frac{2\alpha}{n}\sum_{i=1}^{n}\sin(x_i)\cos(x_i)(y_i - \hat{y}_i)$
- ○ $b^{(t+1)} = b^{(t)} + \frac{2\alpha}{n}\sum_{i=1}^{n}x_i(y_i - \hat{y}_i)$

Solution: Note that $g(x)$ is simply a constant. There is no chain rule involving $g'(x)$, e.g. $\cos x$. The gradient update is:

$$b^{(t+1)} = b^{(t)} + \frac{2\alpha}{n}\sum_{i=1}^{n}\sin(x_i)(y_i - \hat{y}_i)$$

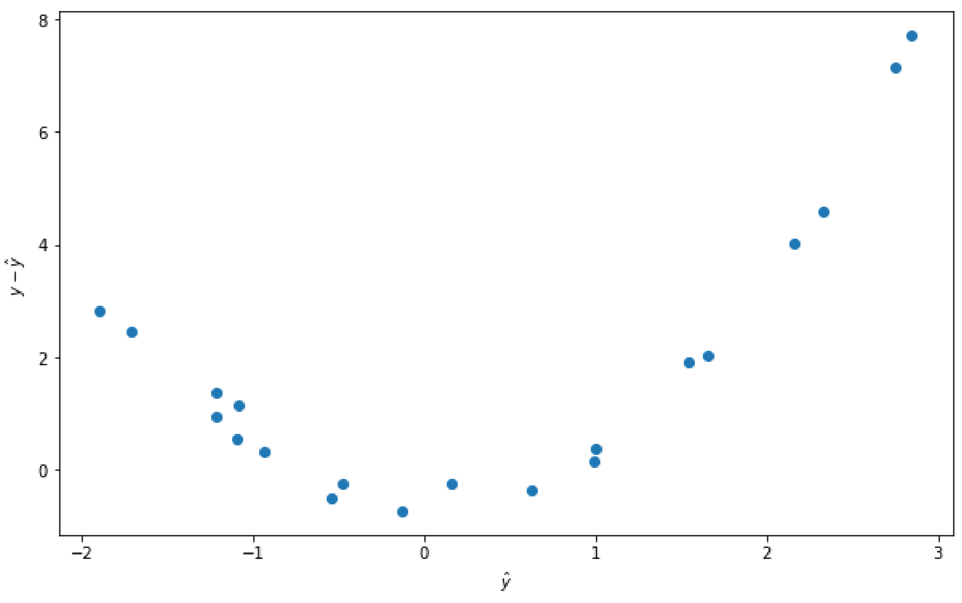
Effectively, this is the exact update used for standard SLR, except where x_i is replaced by $\sin(x_i)$ per the transformation.

(e) [2 Pts] Unfortunately, the previous part yielded a model that performed very poorly! Anirudhan decides to switch to a simple constant model (i.e. $\hat{y} = \theta$) and minimizing the corresponding mean absolute error, using the x and y shown below. With the shown sample of the data, what is the optimal constant model parameter $\hat{\theta}$?

x	y
1	2
3	4
6	5
10	6
9	8

Solution: The median of y is 5. Note the mean was also 5, so minimizing the MSE would have yielded the same answer!

(f) [3 Pts] Regardless of the previous part, suppose Anirudhan obtains a trained multilinear regression model from his favourite machine learning library. Using this, he constructs a residual plot as follows. Which of the following **could be true** based on this residual plot?



☐ This model is overfit to the data.

☐ **This model's parameters θ that generated the predictions $\hat{y}$ are not optimal.**

☐ **This model does not contain a bias term θ_0.**

☐ **This model is inappropriate for this data.**

☐ This model would be improved if we used regularization.

Solution: Option A is incorrect. The model is not overfit since the training loss is quite high in relation to the values shown in previous tables. It has not "memorized" the training data.

Option B is correct. If we use non-optimal parameters, we could end up with the residuals not summing to zero since the residuals need not be orthogonal to the span of the design matrix (if they were, we'd have the optimal solution). This means that the $\vec{1}$ bias vector is not orthogonal to the residual vector - meaning that the sum of residuals is nonzero (note we skipped a few steps here that are covered in the proof in HW 5). Intuitively, we only have optimality with orthogonality, and if we don't have optimality, we lose orthogonality and we lose properties such as the residuals summing to 0 (recalling from HW 5).

Option C is correct. Without a bias term, there is no guarantee that the residuals will sum to 0.

Option D is correct. The clear pattern in the data suggests a linear model with these transformations is inappropriate.

Option E is incorrect. We are not overfitting, so regularization will not help. In fact, it will increase the training loss!