Data C100, Midterm Exam

Summer 2023

Name: _____

Email: _____@berkeley.edu

Student ID: _____

Examination room: _____

Name of the student to your left: _____

Name of the student to your right: _____

Instructions:

Do not open the examination until instructed to do so.

This exam consists of **60 points** spread out over **6 questions** and must be completed in the **110 minute** time period on July 20, 2023, from 5:10 PM to 7:00 PM unless you have pre-approved accommodations otherwise.

For multiple-choice questions with circular bubble options, **select one choice**. For multiple-choice questions with box options, **select all choices that apply**. In both cases, please **shade in** full the box/circle to mark your answer. Do not use checkmarks or "×"s.

Make sure to write your student ID on each sheet (in the provided blanks) to ensure that your exam is graded.

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 page has been intentionally left blank.

1 The Fate of Evans [17 Pts]

You may have heard of the university's upcoming plans to demolish campus building Evans Hall. An **initial survey** was conducted by university administration a few years ago with the goal of **measuring the sentiment of UC Berkeley students towards the demolition of the building**.

The results of this initial survey are stored in the DataFrame `responses`, which contains information about each student who responded to the survey. The `"Response"` column indicates whether or not a student approved ("Yes") or disapproved ("No") of the demolition plans. The first few rows of `responses` are displayed below.

	Student Name	Department	Year	Response
0	Alan Jian	MIDS	5	No
1	Anthony Zhang	EECS	3	No
2	Zaid Maayah	CDSS	3	Yes
3	Tina Chen	CDSS	3	No
4	Yuerou Tang	ESPM	3	Yes
5	Hans Mao	Statistics	4	No

Figure 1: The first 6 rows of `responses`.

(a) [2 Pts] What is the **target population** of the initial survey? **Shade in** the bubble corresponding to your answer; do not use checkmarks or $\times$s.

◯ A. Students who take classes in Evans.
◯ C. UC Berkeley students.
◯ B. University administration.
◯ D. Students who responded to the initial survey.

(b) [2 Pts] We want to conduct a **new study** with the **same target population** as the initial survey. In this new study, we will randomly sample individuals from `responses` and contact them to ask follow-up questions. What is the **sampling frame** of this new study?

◯ A. Students who take classes in Evans.
◯ C. UC Berkeley students.
◯ B. University administration.
◯ D. Students who responded to the initial survey.

(c) [2 Pts] Rather than sample from all individuals in `responses`, we decide to sample from a subset of the DataFrame containing *only* Statistics and Data Science majors. We email the sampled individuals and ask them to fill out a form containing our survey questions. What errors or biases could possibly arise from this sampling scheme? Select all that apply.

☐ A. Selection bias ☐ B. Chance error ☐ C. Response bias ☐ D. Non-response bias

Before conducting our new survey, we want to explore the results stored in `responses`.

The following subparts all involve writing code. For the remainder of this question, assume that `pandas` and `numpy` have been imported as `pd` and `np`, respectively.

(d) [2 Pts] Select the option(s) below that will produce a `Series` **indexed by** `"Response"` that stores the number of students who gave each response in **descending order**.

```
No     4
Yes    2
Name: Response, dtype: int64
```

The desired output based on the previewed DataFrame rows from the previous page is given above. Select all that apply.

☐ A. `responses.groupby("Response").size().sort_values()`

☐ B. `responses["Response"].value_counts()`

☐ C. `responses.groupby("Response").agg("sum")`

☐ D.
```
responses.groupby("Response")["Department"].count() \
                        .sort_values(ascending=False)
```

(e) [2 Pts] Students who are in their 4th or 5th year of college will likely graduate before the building demolition. We suspect that these students may respond differently to other students.

To investigate this theory, select the option(s) below that will extract *only* rows from `responses` corresponding to students who have a `"Year"` **value of 4 or 5**. Assume that the `"Year"` column contains integer values. Select all that apply.

☐ A. `responses[(responses["Year"]==4)|(responses["Year"]==5)]`

☐ B. `responses.loc[:, (responses["Year"]>3)|(responses["Year"]<6)]`

☐ C. `responses.loc[(responses["Year"]>3)|(responses["Year"]<6), :]`

☐ D. `responses.loc[(responses["Year"]>3)&(responses["Year"]<6), :]`

In Parts (f) and (g), you may write **as much code** as you believe is necessary in each blank, provided it follows the given skeleton code.

(f) [3 Pts] Some departments had more respondents than others. We want to see results from departments that only had a few students responding to the survey.

Fill in the blanks to construct the DataFrame `results_dep`. The `results_dep` DataFrame contains the same columns as `results`, but **only** includes rows corresponding to departments that had **fewer than two students** in that department respond to the initial survey.

	Student Name	Department	Year	Response
0	Alan Jian	MIDS	5	No
1	Anthony Zhang	EECS	3	No
4	Yuerou Tang	ESPM	3	Yes
5	Hans Mao	Statistics	4	No

Figure 2: `results_dep` based on the previewed rows of `responses`.

```
results_dep = responses.groupby(___A___).___B___(lambda sf:___C___)
```

(i) Fill in blank `A`:

(ii) Fill in blank `B`:

(iii) Fill in blank `C`:

(g) [4 Pts] We think of a (zany) new research question: could a student's chocolate consumption impact their opinion on the building demolition?

We obtain a new DataFrame, `chocolate`, that contains the chocolate consumption for each student who responded to the initial survey. The first few rows of `chocolate` are displayed below on the left. You may assume that all names are unique.

We wish to construct a DataFrame named `result` that contains all of the original columns of `responses` **as well as** the chocolate consumption for each student. The desired form of `result` is shown below on the right.

	First name	Chocolate Consumption
0	Zaid	1.5
1	Hans	20.5
2	Alan	4.1
3	Tina	4.0
4	Yuerou	4.2
5	Anthony	1.3

Figure 3: `chocolate.head(6)`

	Student Name	Department	Year	Response	Chocolate Consumption
0	Alan Jian	MIDS	5	No	4.1
1	Anthony Zhang	EECS	3	No	1.3
2	Zaid Maayah	CDSS	3	Yes	1.5
3	Tina Chen	CDSS	3	No	4.0
4	Yuerou Tang	ESPM	3	Yes	4.2
5	Hans Mao	Statistics	4	No	20.5

Figure 4: `result.head(6)`

Fill in the blanks to produce `result`. You may include **as many** arguments to function calls and **as many** uses of `str` accessors as you believe are necessary.

```
responses["First name"] = responses["Student Name"].str.___A___

result = responses.___B___(___C___).drop(columns=["First name"])
```

(i) Fill in blank `A`:

(ii) Fill in blank `B`:

(iii) Fill in blank `C`:

2 Jokes Per Minute [12 Pts]

Veep is an American political satire comedy TV series that aired on HBO from 2012 to 2019. It ran for 7 seasons, with a total of 65 episodes. *Veep* has been praised as one of the funniest shows ever, with one of the highest number of jokes per minute (JPM). To see how funny *Veep* actually is, we obtain a dataset stored in the file `veep.tsv` that contains information about the series and its individual episodes, such as the running time and number of jokes in each episode.

For all parts of this question, you may assume that `pandas`, `numpy`, `seaborn`, and `matplotlib` are imported as `pd`, `np`, `sns`, and `plt`, respectively.

(a) [2 Pts] Before loading in the dataset using `pandas`, we inspect it by printing out the first few lines of the file:

```
'Season\tEpisode\tRunning time\tNumber of Jokes\n'
'All\t\t1940.2\t8227\n'
'1\t\t244.8\t1052\n'
'2\t\t294.3\t1264\n'
'3\t\t296.3\t1217\n'
```

Given this preview of the file, complete the following line of `pandas` code to **load this dataset** in as a DataFrame and save it to the **variable `veep`**. You may assume the file `veep.tsv` is in the same directory as the notebook you are working with.

```
veep = pd._____("veep.tsv", _____)
```

Assume we successfully loaded the dataset as a DataFrame named `veep`. The DataFrame contains many more rows than can be viewed on one page. For convenience, we display **only** the first and last 10 rows of the DataFrame below. Running times are measured in minutes.

	Season	Episode	Running time	Number of Jokes
0	All	NaN	1940.2	8227
1	1	NaN	244.8	1052
2	2	NaN	294.3	1264
3	3	NaN	296.3	1217
4	4	NaN	300.2	1277
5	5	NaN	297.1	1233
6	6	NaN	297.8	1289
7	7	NaN	209.7	895
8	1	1.0	30.5	111
9	1	2.0	29.9	126

	Season	Episode	Running time	Number of Jokes
63	6	8.0	29.3	115
64	6	9.0	29.5	143
65	6	10.0	31.1	141
66	7	1.0	30.4	118
67	7	2.0	28.2	120
68	7	3.0	30.4	120
69	7	4.0	29.6	133
70	7	5.0	29.3	134
71	7	6.0	30.7	124
72	7	7.0	31.1	146

Figure 5: The first 10 rows (on the left) and the last 10 rows (on the right) of `veep`.

(b) [2 Pts] Considering only the data displayed on the previous page, which of the following are true about the granularity and structure of the dataset? **Select all that apply**.

- ☐ A. The records are on the same level of granularity.
- ☐ B. The records are not on the same level of granularity.
- ☐ C. This dataset stores rectangular data.
- ☐ D. There are summary/rollup records in the dataset.

(c) [2 Pts] We notice there are some `NaN` values in the `"Episode"` column. Which of the following is most appropriate for addressing these missing values? **Select only one option.**

- ◯ A. Impute them using the average of the rest of the values in the same column.
- ◯ B. Impute them using the mode of the rest of the values in the same column.
- ◯ C. Keep them as `NaN`.
- ◯ D. Replace them with random values.

(d) [1 Pt] Which column (or set of columns) is the primary key of the DataFrame? Select the **minimum set of column(s)** from the options below.

- ☐ A. `"Season"`
- ☐ B. `"Episode"`
- ☐ C. `"Running time"`
- ☐ D. `"Number of Jokes"`

(e) [1 Pt] Jokes per Minute (JPM) is defined as the number of jokes per minute of running time. Complete the following line of `pandas` code to create a new column in `veep` named `"JPM"`.

_____ = _____

(f) [2 Pts] Assume that the `"JPM"` column was created correctly in the previous part.

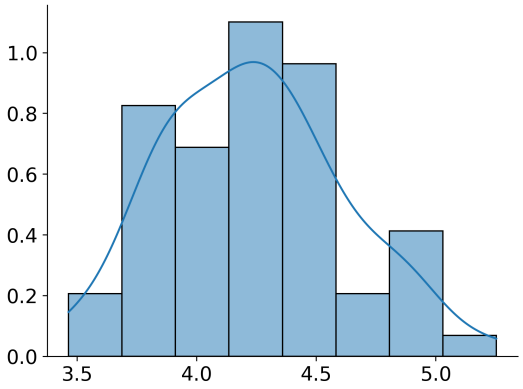
Which of the following lines of `seaborn` code is/are appropriate for visualizing the distribution of the `"JPM"` column using a **histogram** overlaid with a **kernel density estimate** (KDE)? **Select all that apply.**

- ☐ A. `sns.countplot(data=veep, x="JPM", kde=True, stat="density")`
- ☐ B. `sns.displot(data=veep, x="JPM", kde=True, stat="density", rug=False)`
- ☐ C. `sns.kdeplot(data=veep, x="JPM")`
- ☐ D. `sns.histplot(data=veep, x="JPM", kde=True, stat="density")`

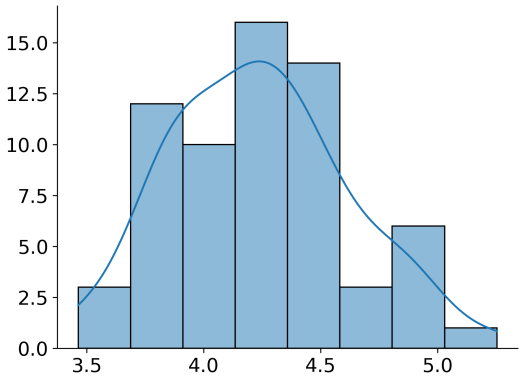
(g) [2 Pts] We want to plot the distribution of the `"JPM"` column.

The plots below show a few attempts at visualizing this distribution. Some of the plots visualize the incorrect variable, and some do not display **valid** kernel density estimates.

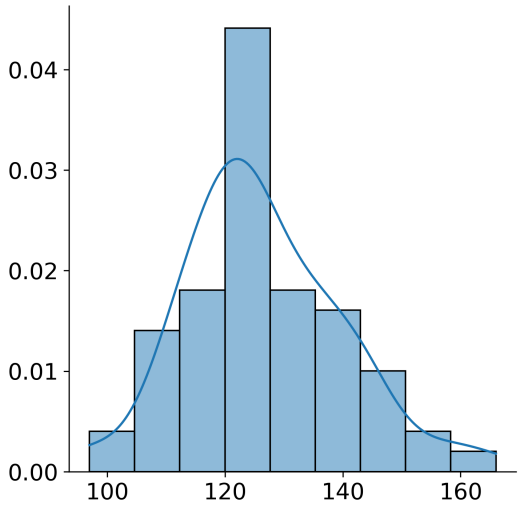
Based on the DataFrame shown in Part (a), which one of the following is most likely the correct plot? Select only one option.



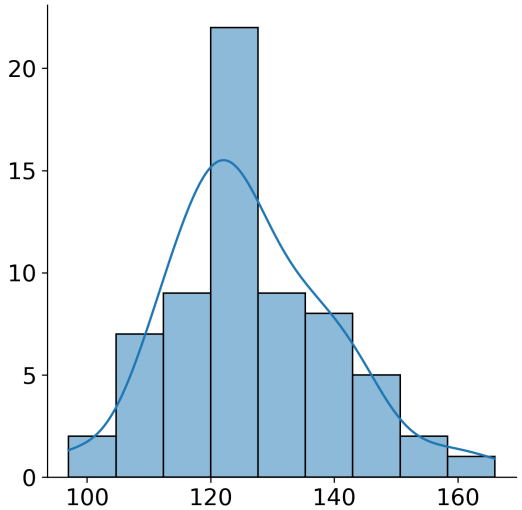
A



B



C



D

◯ A
◯ C

◯ B
◯ D

3 Double Feature [7 Pts]

In this question, the Python regular expression library has been imported as `re`.

Greta and Christopher are filmmakers who have recently released new movies. They decide to use regex to analyze movie reviews so they can understand what audiences like about their films.

(a) [3 Pts] Many critics rate films on a scale of 0 to 5 stars. Christopher wants to write a regex pattern that will **capture star-based ratings, between 0 and 5, mentioned in a review**. Assume that any star-based rating will take the form "# star", where # is a numeric digit between 0 and 5. If a review contains no star-based rating, no text should be returned.

```
pattern_a = # Your selected answer choice(s)

review_1 = "A bombshell. Gripping. A 5 star film."

re.findall(pattern_a, review_1)
```

```
['5 star']
```

```
review_2 = "Over 100 star celebrities came on opening night."

re.findall(pattern_a, review_2)
```

```
[]
```

The correct answer(s) should satisfy the test cases above when the regex pattern is assigned to the variable `pattern_a`. **Select all that apply**.

☐ A. `r"([0-5] star)"`
☐ B. `r"\D+[0-5] star"`
☐ C. `r".*\D([0-5] star)"`
☐ D. `r"[^0-9]+([0-5] star)"`

(b) [4 Pts] Greta is interested in the following movie review. In each of the following parts, select the output generated by running the code below when `pattern_b` is assigned to the corresponding regex pattern.

```
pattern_b = # Pattern specified in each subpart below

review_3 = "It's pink x1000, so fantastic \o/"

re.findall(pattern_b, review_3)
```

(i) Select the output generated when `pattern_b = r"^\D{4}"`

◯ A. `["1000"]`
◯ B. `["It's"]`
◯ C. `[]`
◯ D. `["It's pink x1000, so fantastic \o"]`

(ii) Select the output generated when `pattern_b = r".*\."`

◯ A. `["1000"]`
◯ B. `["It's"]`
◯ C. `[]`
◯ D. `["It's pink x1000, so fantastic \o"]`

4 Our Loss [9 Pts]

(a) [5 Pts] Suppose we have a dataset of one feature. We collect n observations of the form (x_i, y_i). In other words, we record one scalar feature x_i and one scalar target variable y_i for the ith datapoint. We model each observed y_i using a model with one scalar parameter, θ.

$$\hat{y}_i = f_\theta(x_i) = \theta \left(e^{\sqrt{|x_i|}} \right)$$

Determine the optimal value $\hat{\theta}$ that minimizes the mean squared error (MSE) of this model on the dataset in terms of x_i, y_i, and/or n. You may assume, without proof, that the critical point of this loss function is guaranteed to be a minimum.

For full credit, show all work in the space below and **write your final answer on the provided line**. You may receive partial credit for an incomplete solution.

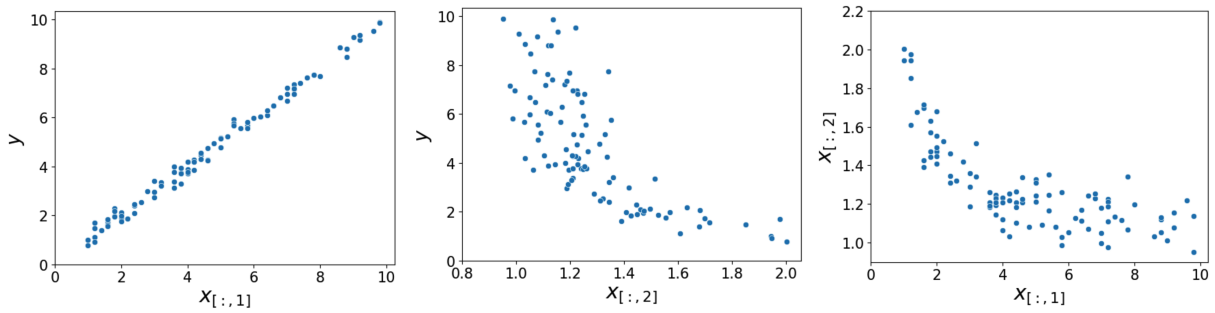
$\hat{\theta} =$ _____

(b) [2 Pts] Which of the following statement(s) are **guaranteed** to be true of the model and the loss function defined in Part (a) for **any dataset** with observations of the form (x_i, y_i)? Select all that apply.

- ☐ A. The sum of residuals of our model will always be 0.
- ☐ B. The predictions of our model can never be negative.
- ☐ C. We can use ordinary least squares (OLS) to estimate the optimal $\hat{\theta}$.
- ☐ D. There is always a single unique optimal value for $\hat{\theta}$ for a given dataset.

(c) [2 Pts] Consider a different dataset of the form $\{(x_{i1}, x_{i2}, y_i)\}_{i=1}^{n}$, where each datapoint i is associated with two features, x_{i1} and x_{i2}, and the target output is y_i.

The plots below display the relationships between the features ($x_{[:,1]}$ and $x_{[:,2]}$) and the target variable (y). You may assume in this subpart that all values of $x_{[:,1]}$, $x_{[:,2]}$, and y are positive.



Given the provided plots, which of the following relationships are likely to be **approximately linear** for appropriate choices of parameters θ_i? Select all that apply.

- ☐ A. $y = \theta_0 + \theta_1 x_{[:,1]}$
- ☐ B. $y = \theta_0 + \theta_1 \log x_{[:,1]}$
- ☐ C. $y = \theta_0 + \theta_1 \log x_{[:,2]}$
- ☐ D. $\sqrt{y} = \theta_0 + \theta_1 \sqrt{x_{[:,2]}}$
- ☐ E. $\log x_{[:,2]} = \theta_0 + \theta_1 \sqrt{x_{[:,1]}}$

5 Hat Trick [8 Pts]

Kippa loves exploring soccer statistics. He obtains a dataset with some of his favorite players' stats from the 2022-2023 season. The dataset contains the following stats for each soccer player:

- `"Dis"`: Total distance, in yards, the player moved the ball, stored as `float`.
- `"League"`: Soccer league of the player's squad, stored as `str`. There are **only two** possible values, La Liga (a Spanish soccer league) or Premier League (an English soccer league).
- `"Goals"`: Average number of goals scored by the player per match, stored as `float`.

Kippa wants to build a model to predict `"Goals"`. He starts with a small subset of five players.

	Player	Dis	League	Goals
0	Erling Haaland	55.7	Premier League	1.373626
1	Harry Kane	116.0	Premier League	0.745614
2	Robert Lewandowski	122.6	La Liga	0.945946
3	Marcus Rashford	148.4	Premier League	0.552764
4	Karim Benzema	165.2	La Liga	0.775862

(a) [1 Pt] Which of the following variable type(s) are **not present** in the DataFrame?

☐ A. Quantitative continuous ☐ B. Qualitative ordinal

☐ C. Quantitative discrete ☐ D. Qualitative nominal

(b) [2 Pts] Kippa uses `"Dis"` and `"League"` as features to build a linear model to predict `"Goals"`. After one-hot-encoding `"League"`, he constructs a design matrix $\mathbb{X}$ and solves the ordinary least squares normal equation to estimate $\hat{\theta}$. He finds that:

- The normal equation $(\mathbb{X}^\top \mathbb{X})\hat{\theta} = \mathbb{X}^\top \mathbb{Y}$ produces a **unique solution** for $\hat{\theta}$
- The residuals $e = \mathbb{Y} - \hat{\mathbb{Y}}$ **sum up to 0**.

Which of the following could **possibly** be his design matrix $\mathbb{X}$? Select all that apply.

☐ A. $\begin{bmatrix} 55.7 & 1 & 0 \\ 116.0 & 1 & 0 \\ 122.6 & 0 & 1 \\ 148.4 & 1 & 0 \\ 165.2 & 0 & 1 \end{bmatrix}$

☐ B. $\begin{bmatrix} 55.7 & 1 \\ 116.0 & 1 \\ 122.6 & 0 \\ 148.4 & 1 \\ 165.2 & 0 \end{bmatrix}$

☐ C. $\begin{bmatrix} 1 & 55.7 & 1 & 0 \\ 1 & 116.0 & 1 & 0 \\ 1 & 122.6 & 0 & 1 \\ 1 & 148.4 & 1 & 0 \\ 1 & 165.2 & 0 & 1 \end{bmatrix}$

☐ D. $\begin{bmatrix} 1 & 55.7 & 1 \\ 1 & 116.0 & 1 \\ 1 & 122.6 & 0 \\ 1 & 148.4 & 1 \\ 1 & 165.2 & 0 \end{bmatrix}$

(c) [2 Pts] Which of the following are **true** about the ordinary least squares estimate $\hat{\theta}$? Select all that apply.

- ☐ A. If $\mathbb{X}$ is not invertible, then $\hat{\theta}$ cannot be calculated using the normal equation.
- ☐ B. $\hat{\theta}$ minimizes the length of the prediction vector $\hat{\mathbb{Y}}$.
- ☐ C. $\hat{\theta}$ minimizes the distance between the true output $\mathbb{Y}$ and predictions $\hat{\mathbb{Y}}$.
- ☐ D. $\hat{\theta}$ minimizes the mean squared error (MSE).

(d) [2 Pts] When making predictions using the fitted model parameter $\hat{\theta}$, Kippa calculates

$$\hat{\mathbb{Y}} = \mathbb{X}\hat{\theta} = \mathbb{X}(\mathbb{X}^\top \mathbb{X})^{-1}\mathbb{X}^\top \mathbb{Y}.$$

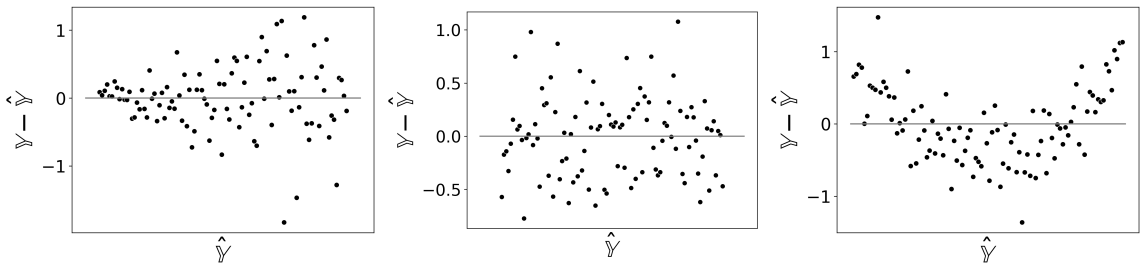
Let $H = \mathbb{X}(\mathbb{X}^\top \mathbb{X})^{-1}\mathbb{X}^\top$, so we can rewrite the above as $\hat{\mathbb{Y}} = H\mathbb{Y}$. Since H projects $\mathbb{Y}$ onto the column space of $\mathbb{X}$, it is often referred to as the projection matrix. It is also called the "hat matrix", because it puts a "hat" on $\mathbb{Y}$.

Which of the following are **true** about the hat matrix H? I is an identity matrix with the same shape as H. Select all that apply.

- ☐ A. H is unchanged when multiplied by itself, that is, $HH = H$.
- ☐ B. $(I - H)\mathbb{Y}$ is orthogonal to the column space of $\mathbb{X}$.
- ☐ C. If a non-zero vector v is in the column space of $\mathbb{X}$, then $Hv = v$.
- ☐ D. If a non-zero vector w is orthogonal to the column space of $\mathbb{X}$, then $Hw = w$.

(e) [1 Pt] Kippa collects more data and fits three different models. To evaluate the models' performance, he generates residual plots for each of the models. He plots **horizontal lines** at $\mathbb{Y} - \hat{\mathbb{Y}} = 0$.

Which of the following residual plots corresponds to the model with the **best** linear regression fit to the data?



A B C

- ◯ A
- ◯ B
- ◯ C

6 Making My Way Downtown [6 Pts]

Vanessa is a taxi driver shuttling passengers across the city of Berkeley. She decides to build a model to predict the time of each trip, y_i, as a function of the level of traffic, x_i. Her model contains two parameters, θ_0 and θ_1, which she stores in the parameter vector $\theta = [\theta_0, \theta_1]^T$.

Vanessa's choice of loss function is given below. She decides to use **batch gradient descent** to select the optimal value of θ.

$$L(\theta_0, \theta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_0 x_i^2 - \theta_1 e^{x_i} + \theta_0\theta_1)$$

(a) [4 Pts] What is the gradient vector, $\nabla_\theta L$, for Vanessa's choice of loss function? **Show your work** in the box below.

◯ A. $\nabla_\theta L = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}(-2\theta_0^{(t)} x_i - \theta_1^{(t)} e^{x_i}) \\ \frac{1}{n}\sum_{i=1}^{n}(1) \end{bmatrix}$

◯ B. $\nabla_\theta L = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}(-x_i^2 + \theta_1^{(t)}) \\ \frac{1}{n}\sum_{i=1}^{n}(-e^{x_i} + \theta_0^{(t)}) \end{bmatrix}$

◯ C. $\nabla_\theta L = \begin{bmatrix} \frac{1}{n}\sum_{i=1}^{n}(-e^{x_i}) \\ \frac{1}{n}\sum_{i=1}^{n}(-x_i^2) \end{bmatrix}$

◯ D. The gradient vector is undefined.

(b) [2 Pts] Which of the following are **not** appropriate choices for the learning rate, α, in gradient descent? Select all that apply.

☐ A. $\alpha = -0.5$ ☐ B. $\alpha = -0.2$ ☐ C. $\alpha = 0$ ☐ D. $\alpha = 0.2$ ☐ E. $\alpha = 0.5$

7 Congratulations [0 Pts]

Congratulations! You have completed the Midterm.

- **Make sure that you have written your student ID number on *every other page* of the exam.** You may lose points on pages where you have not done so.
- Also ensure that you have **signed the Honor Code** on the cover page of the exam for 1 point.
- If more than 10 minutes remain in the exam period, you may hand in your paper and leave. If $\leq$ 10 minutes remain, please **sit quietly** until the exam concludes.

[Optional, 0 pts] Which dog – Beck, Toto, or Shadow – do you think should have won the Data 100 Cutest Dog Contest from Homework 5? Draw a picture!

Summer 2023 Data C100 Midterm Reference Sheet

Pandas

Suppose `df` is a DataFrame; `s` is a Series. `import pandas as pd`

Function	Description
`pd.read_csv(filepath, delimiter)`	Loads the data file at `filepath` into a DataFrame with column fields separated by `delimiter` (`','`, `'\t'`)
`df[col]`	Returns the column labeled `col` from `df` as a Series.
`df[[col1, col2]]`	Returns a DataFrame containing the columns labeled `col1` and `col2`.
`s.loc[rows] / df.loc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their index values.
`s.iloc[rows] / df.iloc[rows, cols]`	Returns a Series/DataFrame with rows (and columns) selected by their positions.
`s.isnull() / df.isnull()`	Returns boolean Series/DataFrame identifying missing values
`s.fillna(value) / df.fillna(value)`	Returns a Series/DataFrame where missing values are replaced by `value`
`s.isin(values) / df.isin(values)`	Returns a Series/DataFrame of booleans indicating if each element is in `values`.
`df.drop(labels, axis)`	Returns a DataFrame without the rows or columns named `labels` along `axis` (either 0 or 1)
`df.rename(index=None, columns=None)`	Returns a DataFrame with renamed columns from a dictionary `index` and/or `columns`
`df.sort_values(by, ascending=True)`	Returns a DataFrame where rows are sorted by the values in columns `by`
`s.sort_values(ascending=True)`	Returns a sorted Series.
`s.unique()`	Returns a NumPy array of the unique values
`s.value_counts()`	Returns the number of times each unique value appears in a Series, sorted in descending order of count.
`pd.merge(left, right, how='inner', on='a')`	Returns a DataFrame joining `left` and `right` on the column labeled `a`; the join is of type `inner`
`left.merge(right, left_on=col1, right_on=col2)`	Returns a DataFrame joining `left` and `right` on columns labeled `col1` and `col2`.
`df.pivot_table(index, columns, values=None, aggfunc='mean')`	Returns a DataFrame pivot table where columns are unique values from `columns` (column name or list), and rows are unique values from `index` (column name or list); cells are collected `values` using `aggfunc`. If `values` is not provided, cells are collected for each remaining column with multi-level column indexing.
`df.set_index(col)`	Returns a DataFrame that uses the values in the column labeled `col` as the row index.
`df.reset_index()`	Returns a DataFrame that has row index 0, 1, etc., and adds the current index as a column.

Let `grouped = df.groupby(by)` where `by` can be a column label or a list of labels.

Function	Description
`grouped.count()`	Return a Series containing the size of each group, excluding missing values
`grouped.size()`	Return a Series containing size of each group, including missing values
`grouped.mean()`/`.min()`/`.max()`	Return a Series/DataFrame containing mean/min/max of each group for each column, excluding missing values
`grouped.filter(f)` `grouped.agg(f)`	Filters or aggregates using the given function `f`

Text Wrangling and Regular Expressions

Pandas `str` methods

Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str[a:b]`	Returns a Series where each element is a slice of the corresponding string indexed from `a` (inclusive, optional) to `b` (non-inclusive, optional)
`s.str.lower()`/`s.str.upper()`	Returns a Series of lowercase/uppercase versions of each string
`s.str.replace(pat, repl)`	Returns a Series that replaces occurences of substrings matching the regex `pat` with string `repl`
`s.str.contains(pat)`	Returns a boolean Series indicating if a substring matching the regex `pat` is contained in each string
`s.str.extract(pat)`	Returns a Series of the first subsequence of each string that matches the regex `pat`. If `pat` contains one group, then only the substring matching the group is extracted
`s.str.split(pat)`	Splits the strings in `s` at the delimiter `pat`. Returns a Series of lists, where each list contains strings of the characters before and after the split.

Regex patterns

Operator	Description	Operator	Description
`.`	Matches any character except `\n`	`*`	Matches preceding character/group zero or more times
`\`	Escapes metacharacters	`+`	Matches preceding character/group one or more times
`|`	Matches expression on either side of expression; has lowest priority of any operator	`^`	Matches the beginning of the string
`\d`, `\w`, `\s`	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively	`$`	Matches the end of the string
`\D`, `\W`, `\S`	Inverse sets of `\d`, `\w`, `\s`, respectively	`( )`	Capturing group or sub-expression
`{m}`	Matches preceding character/group exactly `m` times	`[ ]`	Character class used to match any of the specified characters or range (e.g. `[abcde]` is equivalent to `[a-e]`)
`{m, n}`	Matches preceding character/group at least `m` times and at most `n` times. If either `m` or `n` are omitted, set lower/upper bounds to 0 and ∞, respectively	`[^ ]`	Invert character class; e.g. `[^a-c]` matches all characters except `a`, `b`, `c`

Python `re` methods

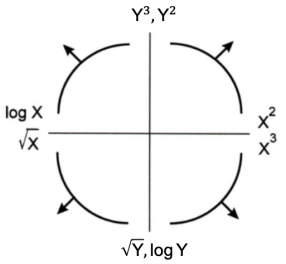
Function	Description
`re.match(pattern, string)`	Returns all matching characters if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns all matching characters if zero or more characters anywhere in `string` matches `pattern`, else None
`re.findall(pattern, string)`	Returns a list of all non-overlapping matches of `pattern` in `string` (if none, returns empty list). If `pattern` includes capturing groups, only return captured characters.
`re.sub(pattern, repl, string)`	Returns `string` after replacing all occurrences of `pattern` with `repl`

Visualization

Matplotlib: `x` and `y` are sequences of values. `import matplotlib.pyplot as plt`

Function	Description
`plt.plot(x, y)`	Creates a line plot of `x` against `y`
`plt.scatter(x, y)`	Creates a scatter plot of `x` against `y`
`plt.hist(x, bins=None)`	Creates a histogram of `x`; `bins` can be an integer or a sequence
`plt.bar(x, height)`	Creates a bar plot of categories `x` and corresponding heights `height`

Tukey-Mosteller Bulge Diagram.



Seaborn: `x` and `y` are keyword arguments assigned to string column names in a DataFrame `data`. `import seaborn as sns`

Function	Description
`sns.countplot(data=None, x=None)`	Create a barplot of value counts of variable `x` from `data`
`sns.histplot(data=None, x=None, stat='count', kde=False)` `sns.displot(data=None, x=None, stat='count', rug=True, kde=True)`	Creates a histogram of `x` from `data`, where bin statistics `stat` is one of `'count'`, `'frequency'`, `'probability'`, `'percent'`, and `'density'`; optionally overlay a kernel density estimator.
`sns.boxplot(data=None, x=None, y=None)` `sns.violinplot(data=None, x=None, y=None)`	Create a boxplot of a numeric feature (e.g., `y`), optionally factoring by a category (e.g., `x`), from `data`. `violinplot` is similar but also draws a kernel density estimator of the numeric feature
`sns.scatterplot(data=None, x=None, y=None)`	Create a scatterplot of `x` versus `y` from `data`
`sns.lmplot(data=None, x=None, y=None, fit_reg=True)`	Create a scatterplot of `x` versus `y` from `data`, and by default overlay a least-squares regression line
`sns.jointplot(data=None, x=None, y=None, kind)`	Combine a bivariate scatterplot of `x` versus `y` from `data`, with univariate density plots of each variable overlaid on the axes; `kind` determines the visualization type for the distribution plot, can be `scatter`, `kde` or `hist`
`sns.kdeplot(data=None, x=None)`	Create a kernel density estimate (KDE) of the distribution of `x` from `data`

Modeling

Concept	Formula	Concept	Formula
Variance, σ_x^2	$\frac{1}{n}\sum_{i=1}^{n}(x_i-\bar{x})^2$	Correlation r	$r=\frac{1}{n}\sum_{i=1}^{n}\frac{x_i-\bar{x}}{\sigma_x}\frac{y_i-\bar{y}}{\sigma_y}$
L_1 loss	$L_1(y,\hat{y})=\mid y-\hat{y}\mid$	Linear regression estimate of y	$\hat{y}=\theta_0+\theta_1 x$
L_2 loss	$L_2(y,\hat{y})=(y-\hat{y})^2$	Least squares linear regression	$\hat{\theta}_0=\bar{y}-\hat{\theta}_1\bar{x}$ $\quad$ $\hat{\theta}_1=r\frac{\sigma_y}{\sigma_x}$
Empirical risk with loss L	$R(\theta)=\frac{1}{n}\sum_{i=1}^{n}L(y_i,\hat{y_i})$		

Ordinary Least Squares

Multiple Linear Regression Model: $\hat{\mathbb{Y}} = \mathbb{X}\theta$ with design matrix $\mathbb{X}$, response vector $\mathbb{Y}$, and predicted vector $\hat{\mathbb{Y}}$. If there are p features plus a bias/intercept, then the vector of parameters $\theta = [\theta_0, \theta_1, \ldots, \theta_p]^T \in \mathbb{R}^{p+1}$. The vector of estimates $\hat{\theta}$ is obtained from fitting the model to the sample $(\mathbb{X}, \mathbb{Y})$.

Concept	Formula	Concept	Formula
Mean squared error	$R(\theta) = \frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2$	Normal equation	$\mathbb{X}^T\mathbb{X}\hat{\theta} = \mathbb{X}^T\mathbb{Y}$
Least squares estimate, if $\mathbb{X}$ is full rank	$\hat{\theta} = (\mathbb{X}^T\mathbb{X})^{-1}\mathbb{X}^T\mathbb{Y}$	Residual vector, e	$e = \mathbb{Y} - \hat{\mathbb{Y}}$
		Multiple R^2 (coefficient of determination)	$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y}$

Scikit-Learn

`import sklearn.linear_model`

Function(s)	Description
`LinearRegression(fit_intercept=True)`	Returns an ordinary least squares Linear Regression model.
`model.fit(X, y)`	Fits the scikit-learn `model` to the provided `X` and `y`.
`model.predict(X)`	Returns predictions for the `X` passed in according to the fitted `model`.
`model.coef_`	Estimated coefficients for the linear model, not including the intercept term.
`model.intercept_`	Bias/intercept term of the linear model. Set to 0.0 if `fit_intercept=False`.

Gradient Descent

Let L be an objective function to minimize with respect to θ; assume that some optimal parameter vector $\hat{\theta}$ exists. Suppose $\theta^{(0)}$ is some starting estimate at $t = 0$, and $\theta^{(t)}$ is the estimate at step t. Then for a learning rate α, the gradient update step to compute $\theta^{(t+1)}$ is:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha\nabla_\theta L$$

where $\nabla_\theta L$ is the partial derivative/gradient of L with respect to θ, evaluated at $\theta^{(t)}$.