Data C100/C200, Final Exam

Spring 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 **130 points** spread out over **12 questions** and must be completed in the **170 minutes** time period on May 11, 2023, from 8:10 AM to 11:00 AM 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.

Make sure to write your SID on each page 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: ______________________________

1 Nom Nom Nom [16 Pts]

Note: For each coding part, you may write as much code as is necessary in the provided blanks.

As a soon-to-be graduate of Data 100, you have been promoted to the principal researcher of your research team, which specializes in analyzing Snackpass meal orders in the city of Berkeley. Fortunately, Snackpass has given you access to their anonymized data, primarily contained in two DataFrames `orders` and `tickets`, shown below on the left and right, respectively.

Each individual is uniquely represented by a distinct **userID**, a sequence of five digits stored as a column of strings in the `orders` DataFrame. In addition, every order is given a unique **orderID** that is a sequence of alphanumeric characters (i.e., letters and/or digits). You may assume that individuals may place multiple orders.

	userID	orderID	restaurant	cuisine
0	12942	7k2l6h	Tony's Style	American
1	14898	8u0n0m	T-toust	Korean
2	48392	4p9a1j	La Burrita	Mexican
3	24849	4z3a2s	Boba Ninja	Korean
4	38041	5v0k3g	Poke Bar	Japanese

`orders`

	ticketID	order_time	completion_time
0	8u0n0m	2023-02-26 12:53:00	2023-02-26 13:59:00
1	5v0k3g	2023-02-26 18:01:00	2023-02-26 18:23:00
2	3d8j3d	2023-02-26 13:32:00	2023-02-26 13:56:00
3	4z3a2s	2023-02-27 09:01:00	2023-02-27 09:12:00
4	2k2s2d	2023-02-27 09:45:00	2023-02-27 10:05:00

`tickets`

(a) [2 Pts] Suppose we want to know whether the **orderID** column is a primary key of the `orders` DataFrame. Which of the following Boolean expressions will evaluate to `True` *only if* the **orderID** column is a primary key of `orders`? **Select all that apply.**

☐ `len(orders) == len(orders["orderID"])`

☐ **`len(orders) == len(orders["orderID"].unique())`**

☐ **`max(orders["orderID"].value_counts()) == 1`**

☐ **`len(orders) == len(orders.groupby("orderID").count().index)`**

(b) [2 Pts] To begin conducting data analysis on restaurant wait times, we must join the data from our two DataFrames. Specifically, we will join the **orderID** column of the `orders` DataFrame with the **ticketID** column of the `tickets` DataFrame. Fortunately, we are told that all the **orderID** values in the `orders` table are present in the **ticketID** column of the `tickets` table, and vice versa. Furthermore, you can assume that the values in each of the **ticketID** and **orderID** columns are unique.

For the following joins, `orders` is our *left* DataFrame, and `tickets` is our *right* DataFrame. Which of the following types of joins will produce our desired result? **Select all that apply.**

☐ **Full Outer Join** ☐ **Right Join** ☐ **Inner Join**

☐ **Left Join** ☐ Cross Join ☐ None of the above

Define the **wait_time** as the time elapsed between **order_time** and **completion_time**. We perform the join from part (b) and compute the wait times to get the Dataframe `merged`, shown below.

	userID	orderID	restaurant	cuisine	order_time	completion_time	wait_time
0	14898	8u0n0m	T-toust	Korean	2023-02-26 12:53:00	2023-02-26 13:59:00	01:06
1	24849	4z3a2s	Boba Ninja	Korean	2023-02-27 09:01:00	2023-02-27 09:12:00	00:11
2	38041	5v0k3g	Poke Bar	Japanese	2023-02-26 18:01:00	2023-02-26 18:23:00	00:22

`merged`

(c) [3 Pts] We want to compare wait times for various restaurants. Replace the **wait_time** column in `merged` with the wait time for each order in minutes. Please fill in the following blanks of code to complete this task. Every new value in this column should be of type `int`.

Notes/Hints: (1) The **wait_time** column currently stores `str` data in the format `HH:MM`, where `HH` represents "hour" and `MM` represents "minutes". (2) When converting a `str` numerical value to an `int` data type, the leading 0 is ignored.

```
hours = ________________________________________

minutes = ______________________________________

merged["wait_time"] = __________________________
```

Solution:

```
hours = merged["wait_time"].str.split(":").str[0].astype(int)
minutes = merged["wait_time"].str.split(":").str[1].astype(int)
merged["wait_time"] = 60 * hours + minutes
```

(d) [3 Pts] Suppose that you would like to analyze Snackpass orders at restaurants where **at least 50 different** individuals have ordered.

Fill in the code below to assign a new DataFrame to `popular`, which contains the same columns as `merged`, but only the subset of rows that satisfy the above restaurant criterion.

```
def f(subframe):

    ____________________________________________

    ____________________________________________

popular = merged.______________(______________)._________(f)
```

Solution:

```
def f(subframe):
    if len(subframe['userID'].unique()) >= 50:
        return True

popular = merged.groupby('restaurant').filter(f)
```

(e) [2 Pts] You further hypothesize that restaurant **wait_time** is related to its **cuisine**. You decide to make a bar plot with every unique **cuisine** type on the x-axis, and the average (mean) **wait_time** on the y-axis. Fill in the following code.

```
avg_wait = popular.groupby(______________)________________________


plt.________________(____________________________, avg_wait.values)
```

Solution:

```
avg_wait = popular.groupby('cuisine')['wait_time'].mean()

plt.bar(avg_wait.index, avg_wait.values)
```

(f) [2 Pts] Which of the following visualizations are most accurate for investigating if some particular cuisines have a longer **median** wait time than others?

☐ **Side-by-side Violin plots** ☐ **Bar plot** ☐ KDE plot

☐ **Side-by-side Box plots** ☐ Count plot ☐ None of the above

Solution: The goal of this question is to plot a quantitative continuous variable (median wait time) against a qualitative categorical variable (cuisine).

- Violin plots and Box plots display the distribution of a continuous variable, with a marker indicating the median of the distribution.
- Bar plots are a flexible visualization that can plot any numerical value against a categorical variable. Here, the median is selected as the numerical value.

Your analysis has revealed a relationship between cuisine and wait time. Moreover, you have strong reason to believe that wait times vary across restaurants, so you decide to use the `popular` DataFrame from part (d) to construct a design matrix $\mathbb{X}$ for further modeling.

(g) [1 Pt] You construct $\mathbb{X}$ by extracting the month, day, and hour from the **order_time** column in the `popular` DataFrame. Additionally, you one-hot-encode both the **restaurant** and **cuisine** columns. Assume that there are **k** unique restaurants and **m** unique cuisines in `popular`. All other columns not mentioned in this problem context are dropped from your design matrix $\mathbb{X}$. In other words, the columns of $\mathbb{X}$ can be represented as follows:

$$\mathbb{X} = [OHE(\textbf{restaurant}), OHE(\textbf{cuisine}), \textbf{month}, \textbf{day}, \textbf{hour}]$$

How many columns does $\mathbb{X}$ have?

Number of columns: ______________

Solution: The one-hot encoding process creates one column in the design matrix for each unique value in the original column of categorical data. Encoding the k unique restaurants in `popular` will introduce k columns to the design matrix; likewise, encoding the m unique restaurants will introduce m more columns. We are also told that the design matrix should include the month, day, and hour columns from `popular`. The total number of columns is:

$$k + m + 3$$

(h) [1 Pt] To increase the predictive power of your model, you decide to augment your feature matrix $\mathbb{X}$ with a bias vector. In other words, the columns of $\mathbb{X}$ can be represented as follows:

$$\mathbb{X} = [\mathbf{1}, OHE(\textbf{restaurant}), OHE(\textbf{cuisine}), \textbf{month}, \textbf{day}, \textbf{hour}]$$

What is the **minimum** number of columns you must remove from your design matrix to ensure $\mathbb{X}$ is full column rank?

Minimum number of columns to remove: ______________

Solution: The bias column and two sets of one-hot encoded columns are linearly dependent – the set of columns in either $OHE($**restaurant**$)$ or $OHE($**cuisine**$)$ can be summed to give $\mathbf{1}$. To prevent this linear combination, we can either (a) remove one column from each of $OHE($**restaurant**$)$ and $OHE($**cuisine**$)$ or (b) remove the bias column and one column from either $OHE($**restaurant**$)$ or $OHE($**cuisine**$)$. In either case, the minimum number of columns to remove is 2

2 Picture This [6 Pts]

In this question, the Python regular expression library has been imported as `re`. Note that \ is used to break strings across code lines without inserting newline characters:

```
In [1]: import re

        oneline_str = "This is a string \
        with no newline characters"
        oneline_str
Out[1]: 'This is a string with no newline characters'
```

Samantha, the internet's favorite data science influencer, is back again! To boost her follower count, she performs EDA on the image captions on the social media platform, Winstagram.

(a) [2 Pts] Emojis are represented by a colon (`:`), followed by the name of the emoji, followed by a second colon (`:`). For example, the sequence `":smile:"` corresponds to the emoji name `"smile"`. Samantha wants to write a regex pattern that will extract the names of all emojis that appear in a string, **without capturing any colons.**

```
In [2]: pattern = # Your selected answer choice

        caption_1 = "omg this dog is soooo cute :dog:"
        re.findall(pattern, caption_1)
Out[2]: ['dog']

In [3]: caption_2 = "can't wait :alarm_clock: for summer vacay :desert_island:"
        re.findall(pattern, caption_2)
Out[3]: ['alarm_clock', 'desert_island']
```

Select the regex pattern below that will achieve this goal. The correct answer should satisfy the test cases above when the regex pattern is assigned to the variable `pattern`.

○ `r":.+:"`　　○ `r"^:([\w\d]*):$"`

[✓ `r":([^:]*):"`]　　○ `r":(\w)+:"`

Solution: `r":.+:"` does not account for the fact that regex is greedy – it will begin matching at the first `:` in the string, then continue matching continuously until the final `:`, regardless of whether or not there are multiple emojis present.

`r"^:([\w\d]*):$"` will only capture captions that contain an emoji and no other text. The match is only satisfied if the first `:` appears the the very beginning of the string (`^`) and the last `:` appears at the very end of the string (`$`).

`r":(\w)+:"` contains a capturing group in the incorrect position. `re.findall` will return each matched capturing group as a separate list item. This pattern encloses just a single character (`\w`) in the capturing group, rather than the full emoji. Note that Regex101 defaults to using `re.match`, rather than `re.findall`.

(b) [4 Pts] Samantha takes a special interest in the following caption. In each of the following parts, select the output generated by running the below cell when `pattern` is assigned to the corresponding regex pattern.

```
In [4]: pattern = # pattern specified in each subpart below

        caption_3 = "YaYayyyyy I never (!!!) have to do another regex final q again :-D"
        re.findall(pattern, caption_3)[0]
```

(i) `pattern = r"[A-Z][a-z]+"`

[✓ `"Ya"`] ○ `"YaYayyyyy"`

○ `"YaYa"` ○ `IndexError`

(ii) `pattern = r"\w|\W+"`

○ `""` ○ `"YaYayyyyy"` ○ `"YaYayyyyy I \ never (!!!) have \ to do another regex \ final q again :-D"`

[✓ `"Y"`] ○ `IndexError`

Solution:
(i) This pattern searches for one uppercase letter and one or more consecutive lowercase letters
(ii) This pattern searches for *either* a single word character *or* one or more non-word characters

3 Estimating Sleep Duration [14 Pts]

(a) [4 Pts] Suppose that we would like to fit a Simple Linear Regression (SLR) model to estimate the sleep duration, $\hat{y}$, given the input feature room temperature, x, measured from the baseline room temperature of $70°$F. The equation for this SLR model is $\hat{y} = \theta_0 + \theta_1 x$. Assume we are given the data table below:

sleep duration, y	room temperature (from $70°$F), x
6	0
12	1
6	-1

Given the summary statistics table below, calculate the parameter estimates of $\theta_0 = \hat{\theta}_0$ and $\theta_1 = \hat{\theta}_1$ that minimizes the Mean Squared Error (MSE) of the SLR model above.

$\bar{x}$	$\bar{y}$	σ_x	σ_y	r	Cov
0	8	$\sqrt{\frac{2}{3}}$	$\sqrt{8}$	$\frac{\sqrt{3}}{2}$	2

$\hat{\theta}_0 =$ ______________ $\hat{\theta}_1 =$ ______________

Solution: $x = room\_temperature$ in this problem context.

$$\hat{\theta}_1 = r \times \frac{\sigma_y}{\sigma_x} = \frac{\frac{\sqrt{3}}{2}\sqrt{8}}{\sqrt{\frac{2}{3}}} = 3$$

$$\hat{\theta}_0 = \bar{y} - \hat{\theta}_1 \times \bar{x} = 8 - 3(0) = 8$$

(b) [8 Pts] Now suppose that we are able to obtain an additional feature: body temperature, measured from the baseline body temperature of 97°F. Given the data table below, we use Ordinary Least Squares (OLS) to fit the model $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2$, which estimates sleep duration, $\hat{y}$, given input features room temperature, x_1, and body temperature, x_2.

sleep duration, y	room temperature (from 70°F), x_1	body temperature from 97°F, x_2
6	0	-2
12	1	1
6	-1	1

Find the ordinary least squares parameter estimate, $\hat{\theta} = \left[\hat{\theta}_0, \hat{\theta}_1, \hat{\theta}_2\right]^T$.

Hint: $\begin{bmatrix} a & 0 & 0 \\ 0 & b & 0 \\ 0 & 0 & c \end{bmatrix}^{-1} = \begin{bmatrix} 1/a & 0 & 0 \\ 0 & 1/b & 0 \\ 0 & 0 & 1/c \end{bmatrix}$ $\hat{\theta} =$ ______________________________

Solution:

$$\mathbb{X} = \begin{bmatrix} 1 & 0 & -2 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix}$$

$$(\mathbb{X}^\top \mathbb{X})^{-1} = (\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -2 \\ 1 & 1 & 1 \\ 1 & -1 & 1 \end{bmatrix})^{-1} = (\begin{bmatrix} 3 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 6 \end{bmatrix})^{-1}$$

$$= \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/6 \end{bmatrix}$$

$$\hat{\theta} = (\mathbb{X}^\top \mathbb{X})^{-1}\mathbb{X}^\top \mathbb{Y} = \begin{bmatrix} 1/3 & 0 & 0 \\ 0 & 1/2 & 0 \\ 0 & 0 & 1/6 \end{bmatrix} \begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & -1 \\ -2 & 1 & 1 \end{bmatrix} \begin{bmatrix} 6 \\ 12 \\ 6 \end{bmatrix}$$

$$= \begin{bmatrix} 8 \\ 3 \\ 1 \end{bmatrix}$$

Therefore, the model will be $\hat{y} = 8 + (3)x_1 + (1)x_2$.

The box below is additional working space for Part (b). You may leave it blank if you completed your solution on the previous page.

(c) [2 Pts] Which of the following is **incorrect** about the OLS model in part (b), fitted to **this specific dataset**?

○ $\hat{\mathbb{Y}}$ is in span($\mathbb{X}$).

○ **$\mathbb{Y}$ is not in span($\mathbb{X}$).**

○ The vector of residuals $\mathbb{Y} - \hat{\mathbb{Y}} = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix}$ is orthogonal to span($\mathbb{X}$).

○ The sum of the residuals $(e_1 + e_2 + e_3)$ is 0.

Solution: The observed y data can be perfectly predicted by the regression line $\hat{y} = 8 + (3)x_1 + (1)x_2$. This can be validated by substituting the values of x_1 and x_2 given in the prompt into the regression equation. In each case, the predicted $\hat{y}$ is equal to the observed sleep duration y.

This means that the predictions $\hat{\mathbb{Y}}$ are equal to the observations $\mathbb{Y}$. Because the predictions $\hat{\mathbb{Y}}$ of an OLS model always lie in span$(\mathbb{X})$, $\mathbb{Y}$ is in span(X) for this specific dataset.

4 So Valid [6 Pts]

Petey, a hungry panda, wants to build a model to predict the growth rate of bamboo. He has collected a training dataset of **72 observations** containing information about bamboo growth patterns. He builds a multiple linear regression model with **four numerical features** and a **bias term**.

Petey wishes to regularize his model. To do so, he decides to use **six-fold cross-validation** to examine **three possible choices** of the regularization hyperparameter, λ. For the purposes of this problem, we will not consider a test dataset; all 72 training datapoints will be used in this cross-validation procedure.

(a) [2 Pts] How many observations will Petey use to train his model in each fold of his cross-validation procedure?

Number of observations = ______________________

Solution: All observations not in the validation set of each fold will be used for training: 72 - 12 = 60 observations.

(b) [2 Pts] How many model parameters must be fitted in each fold of Petey's cross-validation procedure for one choice of a hyperparameter?

Number of parameters = ______________________

Solution: The model uses four numerical features and a bias term, so we assume it takes the form $\hat{y} = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3 + \theta_4 x_4$.

There are a total of five parameters θ_i that must be fitted in each cross-validation fold.

(c) [2 Pts] How many times will Petey compute the error on a validation fold throughout his entire cross-validation procedure?

Number of times validation error is calculated = ______________________

Solution: For each of the three hyperparameter candidate values, Petey will perform six training-validation splits. The validation error will be computed $6 \times 3 = 18$ times

5 Irregular Regularization [15 Pts]

Congratulations! You just landed a new job as a data scientist working at Corn Hole Inc., which has a training program for corn hole, a competitive party game with bean bags. Novices enrolled in the program practice corn hole for a given number of hours then play in a scrimmage game totaling 100 points. You decide to model a linear relationship between a novice's practice time (in hours), x, and their scrimmage score, y. Your model takes on the form $\hat{y} = \theta x$, where θ is a scalar.

(a) [2 Pts] Rather than using L1 or L2 regularization, you decide to create a new regularization term to penalize models with large magnitudes of θ. Which of the following options is the most appropriate choice of regularization term?

○ $e^{-\theta^2}$ [✓ ○ $5^{|\theta|}$] ○ $\frac{1}{\theta}$ ○ θ^3

Solution: An appropriate regularization term should penalize complex models. This means that the regularization term should increase the loss function by a large, positive value when the parameter θ has large magnitude. Of the possible choices, only $5^{|\theta|}$ is large and positive for both very large positive and negative θ.

(b) [5 Pts] Your colleague asks you to use a different regularization term called "Irregular Regularization." The Irregular Regularization objective function takes the form on the right, where λ is the regularization penalty hyperparameter.

$$\left(\frac{1}{n}\sum_{i=1}^{n}(y_i - \theta x_i)^2\right) + \lambda(e^{2+\lambda})\theta^2$$

Derive the optimal solution for $\hat{\theta}$ using the Irregular Regularization objective function in terms of x_i, y_i, λ, **and** n. Please **write your final answer on the provided line**.

$\hat{\theta} =$ ______________________________

Solution: Take the derivative of the objective function with respect to θ, set it equal to 0, and solve for θ.

$$\frac{\partial L}{\partial \theta} = \frac{1}{n}\sum_{i=1}^{n} 2(y_i - \theta x_i)(-x_i) + \lambda(e^{2+\lambda})(2\theta)$$

$$0 = \frac{-2}{n}\sum_{i=1}^{n}(x_i y_i - \theta x_i^2) + \lambda(e^{2+\lambda})(2\theta)$$

$$0 = \frac{-1}{n}\sum_{i=1}^{n} x_i y_i + \frac{\theta}{n}\sum_{i=1}^{n} x_i^2 + \lambda(e^{2+\lambda})\theta$$

$$0 = \frac{-1}{n}\sum_{i=1}^{n} x_i y_i + \theta\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})\right)$$

$$\frac{1}{n}\sum_{i=1}^{n} x_i y_i = \theta\left(\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})\right)$$

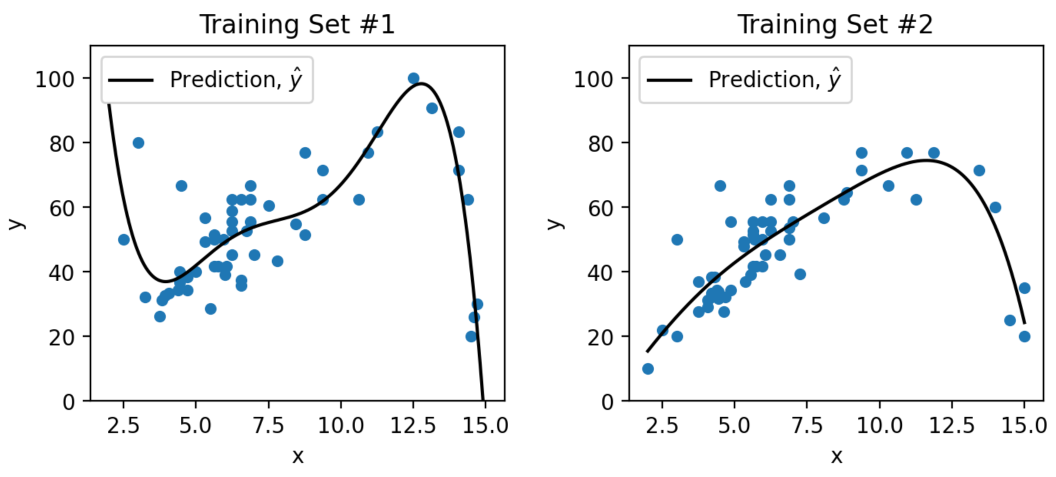
$$\hat{\theta} = \frac{\frac{1}{n}\sum_{i=1}^{n} x_i y_i}{\frac{1}{n}\sum_{i=1}^{n}(x_i^2) + \lambda(e^{2+\lambda})}$$

(c) [2 Pts] You now wish to evaluate your model's performance by computing its MSE on the training set. Assume that the optimal choice of the model parameter under Irregular Regularization is represented by $\hat{\theta}$. Which of the following options should be used to compute the MSE of the regularized model on the training data?

- ◯ $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\theta}x_i)^2$
- ◯ $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\theta}x_i)^2 + \lambda|\hat{\theta}|$
- ◯ $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\theta}x_i)^2 + \lambda\hat{\theta}^2$
- ◯ $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{\theta}x_i)^2 + \lambda(e^{2+\lambda})\hat{\theta}^2$

(d) [3 Pts] Next, you consider models other than the linear model you constructed above. You learn that your new company has a strange policy where all models are fitted to **two training sets**, Training Set #1 and Training Set #2. You next decide to evaluate a new model fit to each of these two training sets.

Consider a quintic (degree 5) model $\hat{y} = \hat{q}_0 + \hat{q}_1 x + \ldots + \hat{q}_5 x^5$, where $\hat{q}_j$ is the MSE parameter estimate for feature j of this quintic model. You first fit this model to Training Set #1; then, you fit this model again to Training Set #2.



(i) The primary contributor to this model's risk is: ◯ High bias ◯ **High variance**

(ii) This contributor to model risk can be addressed by the following (select all that apply):

- ☐ **Reducing model complexity**
- ☐ Adding additional features to the model
- ☐ Reducing the regularization hyperparameter λ
- ☐ **Increasing the regularization hyperparameter λ**

Solution: The prediction $\hat{y}$ for a given value of x varies dramatically when the model is trained to two different training sets. This suggests that the model has high variance, which is indicative of high model complexity. Model complexity can be reduced by increasing the regularization hyperparameter λ.

(e) [3 Pts] Your manager hypothesizes that for a given novice player with x practice hours, their scrimmage performance is $g(x)$, where g is some unknown function you are trying to model. However, the *observed* performance is a random variable $Y = g(x) + \epsilon$, where ϵ is the player's random, zero-mean performance error on scrimmage day. A model's prediction $\hat{Y}(x)$ is also a random variable because the parameter estimates depend on the training set (which your manager assumes is a random sample of past novices' scrimmage performances).

Match the following expressions to the appropriate terms:

		Model Risk	(Model Bias)^2	Model Variance
(i)	$\text{Var}(\hat{Y}(x))$	◯	◯	[✓ ◯]
(ii)	$\left(\mathbb{E}[\hat{Y}(x)] - g(x)\right)^2$	◯	[✓ ◯]	◯
(iii)	$\mathbb{E}\left[\left(\hat{Y}(x) - \mathbb{E}[\hat{Y}(x)]\right)^2\right]$	◯	◯	[✓ ◯]

Solution:

(i) Model variance is defined as the variance of the predictions $\hat{Y}(x)$

(ii) Model bias is the expected difference between the predictions $\hat{Y}(x)$ and the ground truth $g(x)$

(iii) The variance of an arbitrary random variable Z is $\mathbb{E}\left[(Z - \mathbb{E}[Z])^2\right]$. Substituting the predictions $\hat{Y}(x)$, we find that this statement is equivalent to the model variance $\text{Var}(\hat{Y}(x))$

6 Gradient Grab Bag [13 Pts]

(a) [2 Pts] Consider the constant model $\hat{y} = \theta$ used to model an observation y from a dataset $\{y_1, \ldots, y_n\}$. For which of the following objective functions is gradient descent **guaranteed** to find the optimal model parameter $\hat{\theta}$, assuming an appropriate choice of step size and sufficient time for convergence? Select all that apply.

☐ $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} \theta^3$

☐ $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \theta)^6$

☐ $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \theta)$

☐ $L(\theta) = \frac{1}{n}\sum_{i=1}^{n} (y_i - \theta)^2 + \lambda\theta^2$

Solution: Gradient descent can optimize a **convex** function. Convexity can be tested by computing the second derivative of each objective function with respect to θ. Options 2 and 4 have second derivatives that are positive for all values of θ, and are hence convex.

(b) [5 Pts] Now, consider a more complex model: $\hat{y} = \theta_1 + \theta_2^2 x^2 + e^x$. Suppose that MSE is used as the objective function, L, to select the optimal choice of θ_1 and θ_2 in the model above. Assume θ represents the 2×1 column vector $[\theta_1, \theta_2]^T$.

Which of the following expressions represents the **gradient vector** $\nabla_\theta L$? To receive full credit, **show your work** in the box below.

○ $\nabla_\theta L = \begin{bmatrix} \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-2\theta_2^2 x_i - e^{x_i}) \\ \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i}) \end{bmatrix}$

○ $\nabla_\theta L = \begin{bmatrix} \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-1) \\ \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(+1) \end{bmatrix}$

○ $\nabla_\theta L = \begin{bmatrix} \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-1) \\ \frac{2}{n}\sum_{i=1}^{n}(y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-2\theta_2 x_i^2) \end{bmatrix}$

○ $\nabla_\theta L$ is undefined for some θ

○ None of the above. If you select this choice, circle your final answer for the gradient vector in the box below.

Solution: For the given model, the MSE objective function has the form:

$$L(\theta_1, \theta_2) = \frac{1}{n} \sum_{i=1}^{n} (y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})^2$$

To find the gradient vector, we take the derivative of L with respect to θ_1 and θ_2, then stack these results in a column vector.

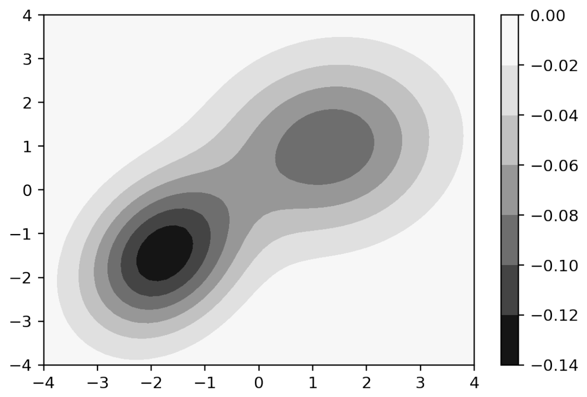
$$\frac{\partial L}{\partial \theta_1} = \frac{2}{n} \sum_{i=1}^{n} (y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-1)$$

$$\frac{\partial L}{\partial \theta_2} = \frac{2}{n} \sum_{i=1}^{n} (y_i - \theta_1 - \theta_2^2 x_i^2 - e^{x_i})(-2\theta_2 x_i^2)$$

Hence, Option 3 is correct.

(c) [2 Pts] Consider the plot shown to the right. This visualization is a contour plot of the loss surface for a model with two parameters, θ_1 and θ_2. The loss is computed on a dataset with observations of the form (x_1, x_2, y).

What values are plotted on the horizontal and vertical axes, respectively, of this contour plot?



◯ x_1 and x_2 ◯ The magnitude of loss and y

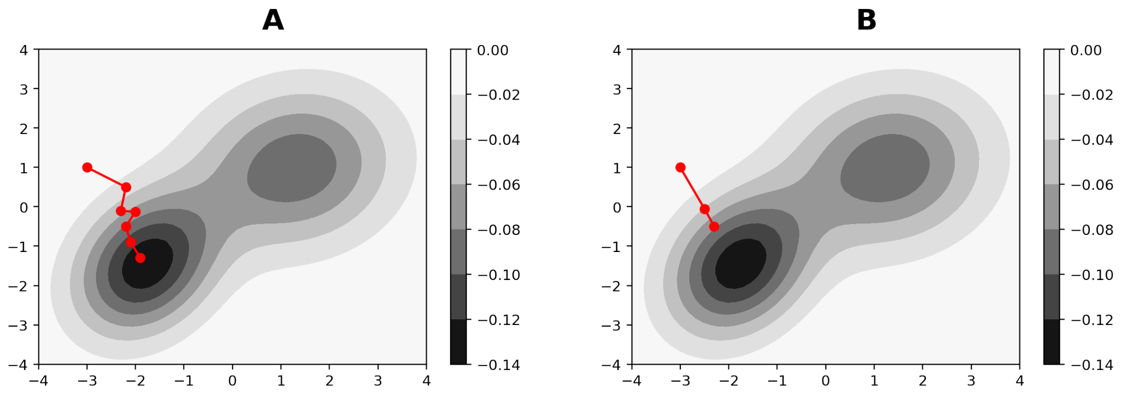
[✓ θ_1 **and** θ_2] ◯ The magnitude of loss and x_1

(d) [2 Pts] Which of the following Gradient Descent (GD) techniques are **guaranteed** to converge to the optimal choice of model parameters for the loss surface in part (c)?

☐ Batch GD ☐ Mini-batch GD ☐ Stochastic GD [✓ **None of the above**]

Solution: Gradient descent is guaranteed to converge on a **convex** function with an appropriate choice of step size and sufficient time for converge. The contour plot above contains two local minima, so the loss surface it represents cannot be convex. It is possible that the three gradient descent techniques listed above may identify the incorrect local minimum as the optimal solution for $\hat{\theta}$.

(e) [2 Pts] The plots below show the same loss surface as in part (c), now annotated with updates for **two full gradient descent epochs** after initialization at the same starting position. Each plot displays the update iterations for a different gradient descent technique: batch gradient descent or mini-batch gradient descent.



In the boxes below, indicate which plot corresponds to batch gradient descent and which plot corresponds to mini-batch gradient descent. Fill each box with "A" or "B".

Batch gradient descent

Mini-batch gradient descent

Solution:

Batch gradient descent: B

Mini-batch gradient descent: A

- Mini-batch gradient descent only uses a subset of the total training set to compute the gradient vector at each update, so it is not guaranteed to always find the direction of greatest descent towards the global minimum. This causes the mini-batch gradient descent updates to zig-zag away from the path to the true minimum
- Additionally, mini-batch gradient descent includes multiple updates in each epoch. This is the reason for the extra steps taken in the mini-batch plot

7 Live from Wheeler [11 Pts]

Professors Yan and Norouzi want to understand student attendance at Data 100 in-person lectures, where there are 1200 total students enrolled in Data 100 this semester.

Let the random variables X and Y represent the total number of students who attend the Data 100 lecture in-person at Wheeler 150 on a single Tuesday and a single Thursday, respectively. Suppose that X and Y are independent (restated, whether or not a student attends a lecture on Tuesday does not influence if they attend a lecture on Thursday) with the following properties:

X	$\mathbb{E}[X] = 900$	$\text{Var}(X) = 225$
Y	$\mathbb{E}[Y] = 600$	$\text{Var}(Y) = 300$

(a) [2 Pts] The professors want to simulate the *total* number of students who attend an in-person live lecture in a single week. To do so, they construct the random variable $Z = X + Y$ (which intentionally double-counts students who attend both days). Compute the expectation of Z, $\mathbb{E}[Z]$. Justify your answer.

$\mathbb{E}[Z] =$ ______________________

Solution: By the linearity of expectation:

$$\mathbb{E}[Z] = \mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$$

$$\mathbb{E}[Z] = 900 + 600 = 1500.$$

(b) [1 Pt] What is the covariance of X and Y, $\text{Cov}(X, Y)$?

○ -450

○ -56.25

[✓ **0**]

○ 56.25

○ 450

Solution: It is given in the prompt that X and Y are independent variables. Variables that are independent necessarily have covariance 0.

(c) [4 Pts] Assume that we are in Week 5 of the semester. To encourage attendance, the professors plan to bring boba to all students who attend in-person lectures next week! Assuming not all students will want boba, they define the random variable $B = 0.5Z + 10$ to compute how many total drinks to purchase next week.

Compute the **variance** of B, Var(B). You may leave your answer as a numeric expression without simplifying, but justify your answer.

Var(B) = ______________________

Solution: Because Var$(aX + b) = a^2$Var(X), we can simplify:

$$\text{Var}(B) = \text{Var}(0.5Z + 10) = (0.5^2)\text{Var}(Z) = 0.25\text{Var}(X + Y)$$

In general, Var$(X + Y) =$ Var$(X) +$ Var$(Y) + 2$Cov(X, Y). From the previous part, X and Y are independent and therefore have covariance 0. Then:

$$\text{Var}(B) = 0.25\text{Var}(X + Y) = 0.25\,(\text{Var}(X) + \text{Var}(Y))$$

$$\text{Var}(B) = 0.25\,(225 + 300) = 131.25$$

(d) [2 Pts] Prior to ordering boba, the professors want to determine which flavor of tea students like: Oolong, Jasmine, or Passionfruit. On Thursday (of Week 5), the professors stand outside the lecture hall and survey anyone who passes by them in the hour before Data 100 lecture starts ("survey time"). Based on survey results, the professors then order boba for the following week.

Which of the below best describes the sampling frame?

○ Every student in Data 100

○ Data 100 students who like Oolong, Jasmine, or Passionfruit tea

[✓ **Anyone passing the professors during survey time**]

○ Any Data 100 student passing the professors during survey time

○ Any Data 100 student attending the lecture that day

(e) [2 Pts] Based on the sampling procedure in part (d), which of the following statements are true? Assume the population of interest is Data 100 students enrolled this semester, and that all survey respondents respond exactly once. Select all that apply.

- ☐ **This procedure produces a convenience sample but not a probability sample.**
- ☐ If all Data 100 students pass by the professors during survey time, the sample is the same as the population of interest.
- ☐ **Selection bias is present in this procedure.**
- ☐ None of the above

8 Come Fly With Me [15 Pts]

This question involves SQL databases. All code for this question, where applicable, must be written as SQLite queries. ***In each blank, you may write as much code as is necessary, provided it fits the given skeleton code***. Throughout this question, you may assume that **any numeric data is stored as floats**.

A UC Berkeley professor is conducting research on peregrine falcons, a type of bird. The professor has collected information about all falcons she has observed in a table named `falcons` within a SQLite database.

The result of the following query is shown to the right.
`SELECT * FROM falcons LIMIT 5;`

name	age	nesting_site
Annie	5	Campanile
Lou	3	Campanile
Alden	4	Mount Tam
Lindsay	1	Golden Gate
Grinnell Jr	1	Mount Tam

(a) [4 Pts] The professor wants to summarize the demographics of the falcons she observes. Write a SQL query to return a copy of the original `falcons` table with an additional column named `age_group`, as shown below. The `age_group` column should have a value of `"young"` if a falcon is **2 years old or younger**. Otherwise, the `age_group` column should have a value of `"adult"`.

name	age	nesting_site	age_group
Annie	5	Campanile	adult
Lou	3	Campanile	adult
Alden	4	Mount Tam	adult
Lindsay	1	Golden Gate	young
Grinnell Jr	1	Mount Tam	young

```
SELECT name, age, nesting_site, _______A_______ AS age_group
FROM falcons;
```

Fill in Blank A:

Solution:

```
CASE WHEN age > 2 THEN "adult"
     ELSE "young"
     END
```

For the remainder of this question, assume that `falcons` has been reassigned to the result of the query written in Part (a). In other words, `falcons` is the table displayed in Part (a).

(b) [5 Pts] As part of her research, the professor wants to determine which falcons may be mates. She theorizes that falcons with the same `age_group` and `nesting_site` are likely mates.

Write a SQL query to identify pairs of falcons who share the same `age_group` and `nesting_site`. The first two rows of the resulting table are displayed to the right.

mate_1	mate_2
Annie	Lou
Lou	Annie

Hint: Make sure that no falcon is a mate with itself!

```
SELECT ______A_______ AS mate_1, _______B_______ AS mate_2
FROM ____________C_____________
WHERE _____________________D______________________;
```

(i) Fill in Blank A:

Solution:

```
a.name
```

(ii) Fill in Blank B:

Solution:

```
b.name
```

(iii) Fill in Blank C:

Solution:

```
falcons AS a JOIN falcons AS b
```

(iv) Fill in Blank D:

Solution:

```
a.name != b.name AND a.age_group = b.age_group
AND a.nesting_site = b.nesting_site
```

(c) [6 Pts] Each time the professor observes a falcon, she records information about the falcon and its flight behavior at the time of the observation. She stores all observation data recorded *before* the year 2023 in the table `past_obs`, and the data recorded in 2023 or later in the table `new_obs`. The first few rows of `past_obs` and `new_obs` are shown below.

date	name	speed	duration
1/13/2022	Alden	200	5
2/5/2022	Alden	240	2
4/10/2022	Annie	100	10
5/11/2022	Grinnell Jr	150	13
11/26/2022	Lindsay	210	4

past_obs

date	name	speed	duration
1/5/2023	Lou	160	8
2/1/2023	Lindsay	30	19
3/23/2023	Annie	120	2
3/30/2023	Lou	170	3
4/11/2023	Annie	230	6

new_obs

Write a SQL query to output a table with **two columns**, `name` and `speed_diff`. The column `speed_diff` should be the difference between each falcon's **average speed in 2023** and its **average speed in 2022 and earlier**.

If a falcon is **not present in** `past_obs`, it should have a `speed_diff` of NULL or NaN. If a falcon is **not present in** `new_obs`, it should be omitted.

name	speed_diff
Annie	75.0
Lindsay	-180.0
Lou	NaN

```
SELECT n.name, _______A_______ AS speed_diff
FROM new_obs AS n
_______B_______ past_obs AS p ON n.name = p.name
_________________C_________________;
```

(i) Fill in Blank A:

Solution:

`AVG(n.speed) - AVG(p.speed)`

(ii) Fill in Blank B:

Solution:

`LEFT JOIN`

(iii) Fill in Blank C:

Solution:

`GROUP BY n.name`

9 Decisions, Decisions [14 Pts]

Consider the following binary classification problem with 4 data points. Suppose that we use a logistic regression model to predict the probability that $y = 1$ given x:

$$\hat{y} = P_{\hat{\theta}}(y = 1|x) = \sigma(x^T\theta)$$

x_1	x_2	y
0	1	1
1	0	0
-1	0	1
0	-1	0

(a) [1 Pt] After fitting a logistic regression model (without regularization) on features x_1, x_2 without an intercept and class labels y, we find that the predicted probabilities returned by sklearn are given below.

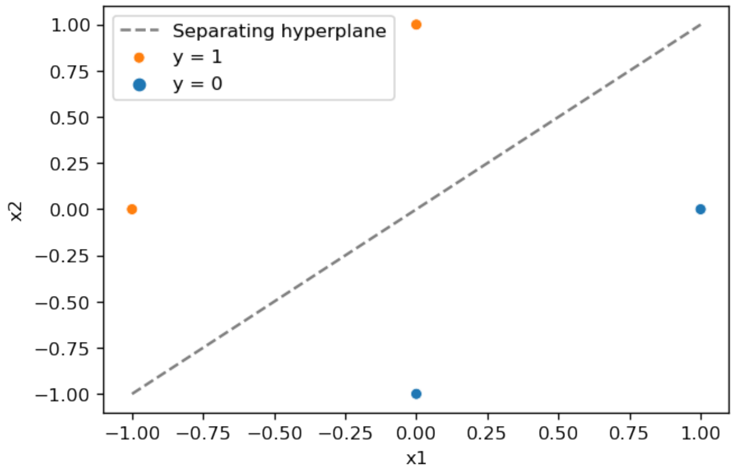
```
array([[2.69411751e-05, 9.99973059e-01],
       [9.99973059e-01, 2.69411751e-05],
       [2.69411751e-05, 9.99973059e-01],
       [9.99973059e-01, 2.69411751e-05]])
```

Is this data linearly separable?

○ Yes

○ No

Solution: A dataset is linearly separable if a line can be drawn in terms of the features x_i that perfectly separates the two classes y. A sketch of the data shows that this dataset is linearly separable:



(b) [2 Pts] Given your answer to part (a), what are weights $\hat{\theta} = [\hat{\theta}_1, \hat{\theta}_2]^T$ that minimize mean cross entropy loss? Assume no regularization.

$\hat{\theta}_1 = \boxed{}$

$\hat{\theta}_2 = \boxed{}$

Solution: When a dataset is linearly separable, the weights of an unregularized logistic regression model diverge to infinity. To determine the appropriate signs, consider the provided datapoints.

When $x_1 = 1, x_2 = 0$, $\hat{y} = \frac{1}{1+e^{-\theta_1(1)-\theta_2(0)}} = \frac{1}{1+e^{-\theta_1}}$ should approach $y = 0$.

This occurs as $\theta_1 \to -\infty$

When $x_1 = 0, x_2 = 1$, $\hat{y} = \frac{1}{1+e^{-\theta_1(0)-\theta_2(1)}} = \frac{1}{1+e^{-\theta_2}}$ should approach $y = 1$.

This occurs as $\theta_2 \to \infty$.

(c) [2 Pts] We train a new logistic regression model (without an intercept) with regularization and find the optimal model parameters to be $\hat{\theta} = [-2/3, 2/3]^T$.

Suppose that we observe a new data point $x_{new} = [x_{new,1}, x_{new,2}]^T = [2, 1]^T$. Calculate the probability that our model believes x_{new} belongs to class 0.

Note: You may leave your final answer as an expression in terms of e.

Answer: ______________________________

Solution:

$$\hat{y} = P_{\hat{\theta}}(y = 1|x_{new}) = \sigma([2, 1] * [-2/3, 2/3]^T) = \sigma(-2/3) = \frac{1}{1 + e^{2/3}}$$

$$P_{\hat{\theta}}(y = 0|x_{new}) = 1 - \frac{1}{1 + e^{2/3}}$$

(d) [2 Pts] We decide to use a new dataset of 6 training points. For the remainder of this question, we will **only consider these 6 new training points**. We train a new logistic regression model to find the model's predicted probabilities, displayed in the table below.

Data Point #	x_1	x_2	y	$\hat{y}$
1	1	1	1	0.65
2	0.5	2.5	1	0.90
3	0.75	0.5	1	0.75
4	0	0	1	0.85
5	0.5	1.5	0	0.70
6	1	0	0	0.45

Which data point incurs the highest cross-entropy loss?

Hint: You may find that computing cross-entropy loss is not required to answer this question correctly.

○ Point 1 ○ Point 3 [✓ **Point 5**]

○ Point 2 ○ Point 4 ○ Point 6

Solution: The prediction $\hat{y} = 0.70$ for Point 5 is furthest from the corresponding observed value $y = 0$.

(e) [2 Pts] What are the range(s) of classification thresholds T that maximize(s) accuracy? **Select all that apply**.

Note: Interval notation (*a*, *b*] refers to the range of integers from (but not including) a up to and including b.

☐ [0, .45] ☐ (.65, .70] ☐ (.75, .85]

☐ **(.45, .65]** ☐ **(.70, .75]** ☐ (.85, .90]

Solution: Any threshold between .45 and .65 results in 4 true positives and 1 true negative. Any threshold between .70 and .75 results in 3 true positives and 2 true negatives. In either case, the accuracy of the model is $\frac{TP+TN}{TP+TN+FP+FN} = \frac{5}{6}$

(f) [2 Pts] Suppose we are interested in evaluating our model's performance with an ROC curve.

To construct an ROC curve, we first need to find our model's TPR and FPR at various classification thresholds T. The TPR and FPR for the majority of thresholds have been computed for you in the table below; please select the correct values for the remaining entries.

Note: Multiple letters may correspond to the same value.

T	TPR	FPR
0.95	0	0
0.9	1/4	0
0.85	1/2	0
0.75	A	B
0.7	C	D
0.65	1	0.5
0.45	1	1

For your convenience, the data table from **part (d)** is repeated below.

Data Point #	x_1	x_2	y	$\hat{y}$
1	1	1	1	0.65
2	0.5	2.5	1	0.90
3	0.75	0.5	1	0.75
4	0	0	1	0.85
5	0.5	1.5	0	0.70
6	1	0	0	0.45

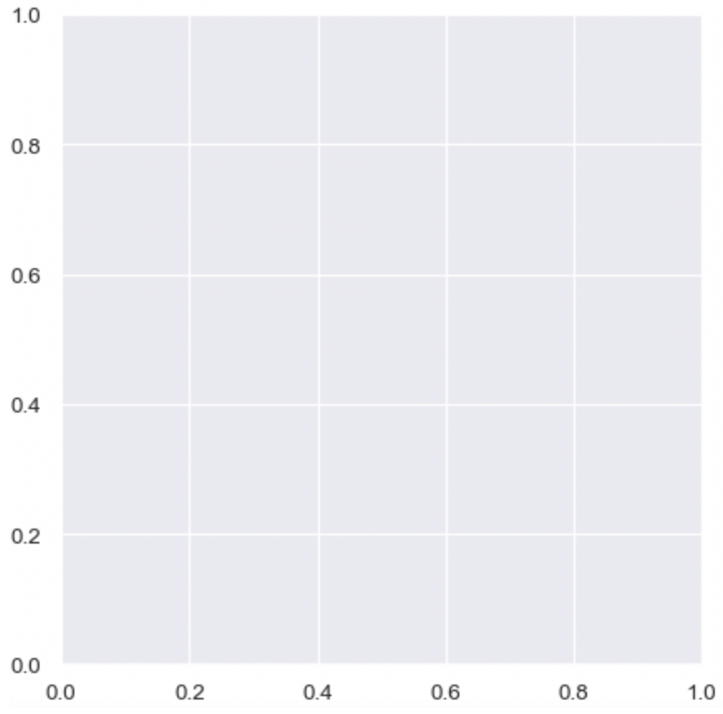
	0	1/4	2/4	3/4
(i) A	○	○	○	[✓ ○]
(ii) B	[✓ ○]	○	○	○
(iii) C	○	○	○	[✓ ○]
(iii) D	○	○	[✓ ○]	○

Solution: When the threshold is 0.75, there are 3 true positives, 2 true negatives, and 1 false negative. The resulting metrics are $TPR = \frac{3}{3+1} = \frac{3}{4}$ and $FPR = \frac{0}{0+2}$

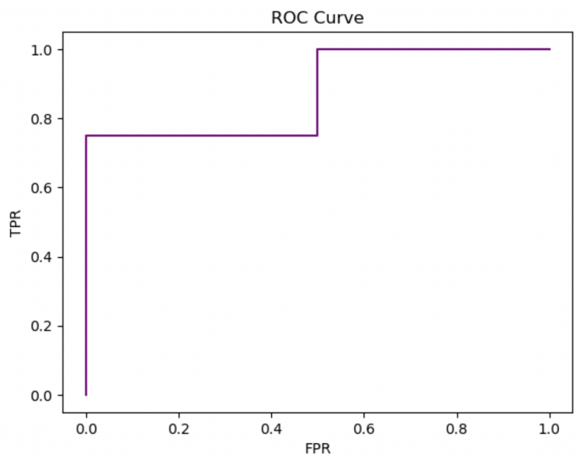
When the threshold is 0.7, there are 3 true positives, 1 true negative, 1 false positive, and 1 false negative. The resulting metrics are $TPR = \frac{3}{3+1} = \frac{3}{4}$ and $FPR = \frac{1}{1+1} = \frac{1}{2} = \frac{2}{4}$

(g) [3 Pts] Let's construct our ROC curve! For various threshold values, the x-axis of an ROC curve displays the FPR, while the y-axis shows the TPR.

Given your answers from the last subpart, construct the ROC curve on the given plot **and** compute the AUC (Area under the ROC curve). You can write the AUC as a simplified fraction or decimal.



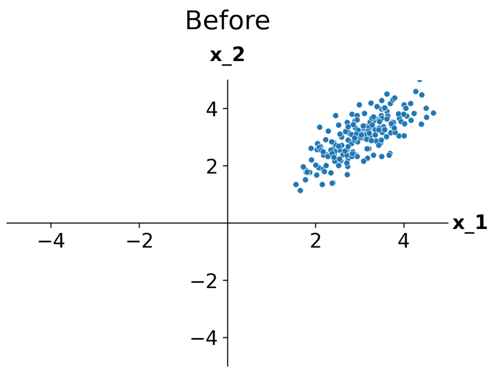
AUC: ______________________



Solution:

The AUC under this curve can be found by taking areas of rectangles. We have $(3/4 * 1/2) + (1 * 1/2) = 7/8$

10 Give it a Twirl [8 Pts]



Consider a dataset containing two features, x_1 and x_2. The dataset is stored in a DataFrame `X` with two columns, `"x_1"` and `"x_2"`, corresponding to the features x_1 and x_2. On the right, the plot labeled "Before" visualizes the variables `"x_1"` and `"x_2"`.

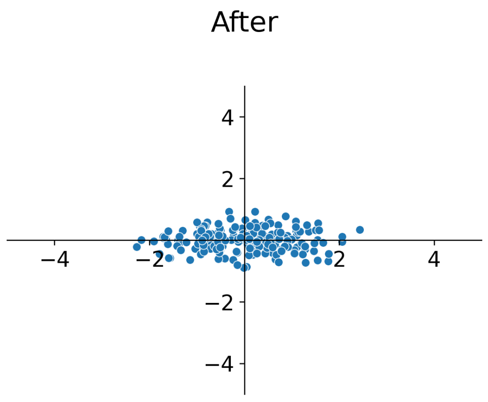
In this question, you will center and rotate this data.

(a) [2 Pts] Fill in the blank to **center** the data in `X`. Assume `pandas` is imported as `pd` and `numpy` as `np`.

`X = ______________________`

Solution: Either syntax is correct:

```
X - X.mean()
X - np.mean(X)
```



(b) [2 Pts] Next, you produce the following plot on the right labeled "After" that rotates the data, where **the two directions of greatest variance align with the coordinate axes**.

What values are plotted on the horizontal and vertical axes of this new visualization? Answer this question by stating appropriate labels for the two axes as English phrases.

Horizontal axis:

Solution: The first principal component of the data

Vertical axis:

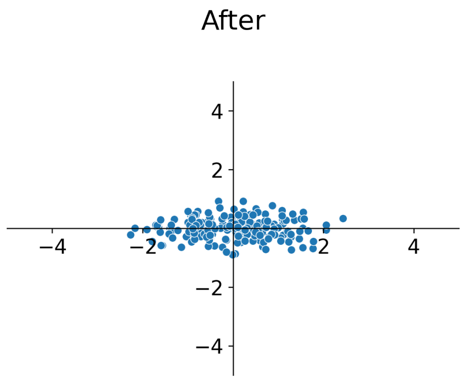
Solution: The second principal component of the data

The "After" plot of the rotated data, shown again below for convenience, was generated with the below code snippet:

```
U, S, VT = np.linalg.svd(X,
                         full_matrices=False)

# create a diagonal matrix from
# the values in the array S
Sigma = np.diag(S)

plt.scatter(x=(U@Sigma)[:, 0],
                y=(U@Sigma)[:, 1]);
```



(c) [2 Pts] Given the "After" plot of rotated data, which of the following matrices is **most likely** to represent the `Sigma` matrix?

○ $\begin{bmatrix} 0.5 & 0.45 \\ 0.45 & 0.75 \end{bmatrix}$ ○ $\begin{bmatrix} 0.75 & 0.45 \\ 0.45 & 0.5 \end{bmatrix}$ ○ $\begin{bmatrix} 10 & 0 \\ 0 & 8 \end{bmatrix}$ [✓ $\begin{bmatrix} 13 & 0 \\ 0 & 5 \end{bmatrix}$]

Solution: `Sigma` is the singular value matrix, where the singular values σ_i are given along the diagonal and the remaining entries in the matrix are 0. The variance captured by the ith principal component is given by $\frac{\sigma_i^2}{n}$. Examining the given plot, the spread of the data in the horizontal direction is well over twice the spread of the data in the vertical direction. This suggests that the first principal component captures substantially more variance than does the second principal component, so $\frac{\sigma_1^2}{n}/\frac{\sigma_2^2}{n}$ should be large (greater than 2, as a rough approximation). This condition is best satisfied by the final option.

Assume that `U`, `Sigma`, and `VT` are represented by the matrices U, Σ, and V^T, respectively. V^T is known to be an **orthonormal square matrix**.

(d) [1 Pt] Which of the following expressions is/are equivalent to $U\Sigma V^T$? Select all that apply.

[✓ □ X] [✓ □ $U\Sigma V^{-1}$] □ $(U\Sigma V^T)^{-1}$ □ XV

Solution: By the singular value decomposition, $X = U\Sigma V^T$. Because V^T is an orthonormal square matrix, $V^T = V^{-1}$.

(e) [1 Pt] Which of the following expressions is equivalent to $U\Sigma$?

○ ΣU [✓ ○ XV] ○ XV^T ○ VX

Solution: Using the singular value decomposition and the fact that $V^T = V^{-1}$:

$$\begin{aligned} X &= U\Sigma V^T \\ X &= U\Sigma V^{-1} \\ XV &= U\Sigma V^{-1}V \\ XV &= U\Sigma \end{aligned}$$

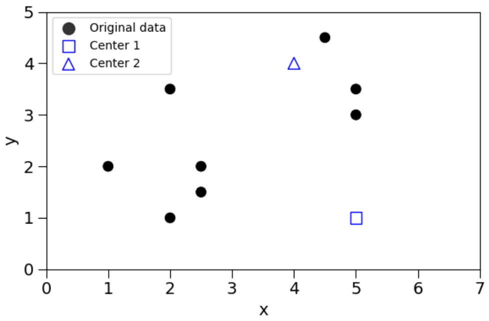
11 Cluster It All [6 Pts]

You are given the following eight (x, y) pairs to perform a clustering task:

$$\{(2,1),(2.5,1.5),(2.5,2),(1,2),(5,3),(4.5,4.5),(2,3.5),(5,3.5)\}$$

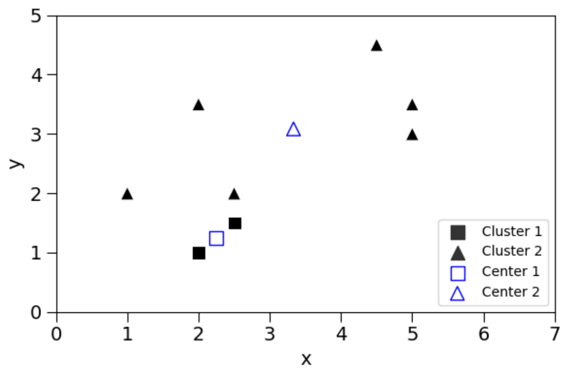
(a) [4 Pts] Suppose you ran the K-Means algorithm to identify TWO clusters in the data, with the initial assignment of cluster centers as Center 1: $(5, 1)$ and Center 2: $(4, 4)$.

In other words, you have the starting state to the right, where Center 1 and Center 2 are the outlined square and triangle markers, respectively.

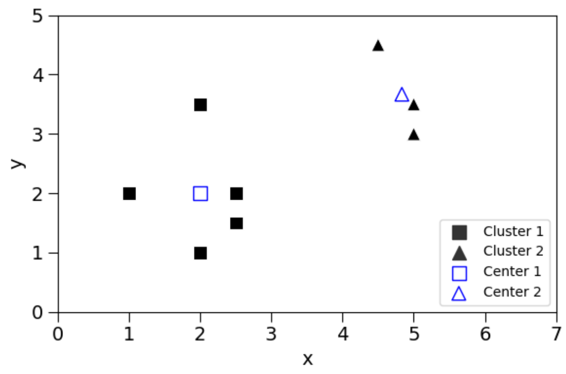


What is the result of K-Means after **ONE** iteration? All figures use the legend in Choice A, which lists the markers for the two clusters and two centers.

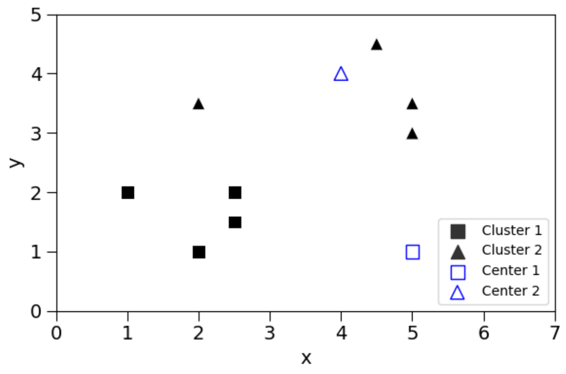
◯ A.



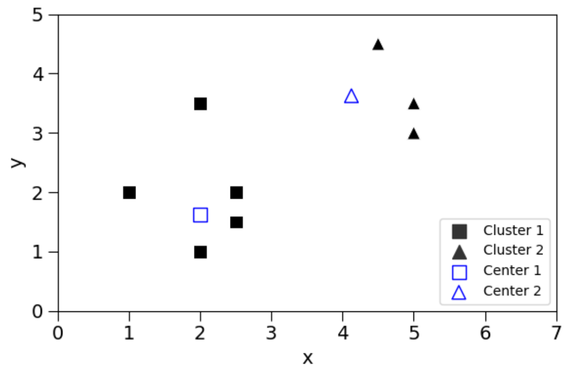
◯ B.



◯ C.



◯ D.



Solution: The two lowermost datapoints are closest to Center 1, while the remaining datapoints are closest to Center 2. After updating the labels and plotting the resulting cluster centers of the relabeled datapoints, Plot A is produced.

(b) [2 Pts] Which of the following statements about clustering algorithms is true?

○ All clustering algorithms are sensitive to initialization.

○ K-Means clustering always converges to the global optimum.

○ Hierarchical clustering is a better choice compared to K-Means on large datasets.

○ **The single linkage criterion considers the distance between two clusters as the minimum distance between a point in the first cluster and a point in the second.**

12 Be Our Guest (Lecture) [5 Pts]

(a) [2 Pts] Recall the March 21st lecture on climate and physical data presented by Professor Fernando Pérez. In this lecture, we explored the Keeling curve, which plots the mean fraction of carbon dioxide (CO_2) present in the air each month over the span of several years. What major trend could be observed from this data?

- ◯ CO_2 levels increased rapidly until the introduction of electric vehicles, then decreased gradually.
- ◯ **CO_2 levels fluctuated seasonally but tended to increase with each passing year.**
- ◯ CO_2 levels strictly increased at an increasing rate over time.
- ◯ CO_2 levels strictly increased at a decreasing rate over time.

(b) [3 Pts] Recall the March 16th lecture on the Cook County Assessor's Office (CCAO) presented by Dr. Ari Edmundson. In 1-2 sentences, describe some of the central problems with the property valuation model used by the CCAO to determine property taxes.

Solution: Answers will vary, but possible themes include:

- The CCAO systematically undervalued more expensive properties and overvalued less expensive properties, which resulted in a regressive tax system that placed a disproportionate tax burden on less wealthy homeowners
- Wealthy homeowners often had the resources to hire tax lawyers to appeal their tax assessments, while homeowners with fewer resources often could not contest their assessments in this way

13 Congratulations [0 Pts]

Congratulations! You have completed the Final.

- **Make sure that you have written your student ID number on *each 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.

[Optional, 0 pts] What should a Data 100 sticker look like? Draw a picture!

Spring 2023 Data C100/C200 Final Reference Sheet

Pandas

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

Function	Description
`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
`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

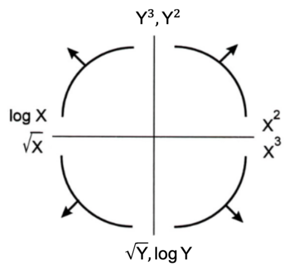
Function	Description
`s.str.len()`	Returns a Series containing length of each string
`s.str[a:b]`	Returns a Series where each element is a slice of the corresponding string indexed from `a` (inclusive, optional) to `b` (non-inclusive, optional)
`s.str.lower()/s.str.upper()`	Returns a Series of lowercase/uppercase 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

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 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. `displot` is similar but can optionally overlay a rug plot
`sns.rugplot(data=None, x=None)`	Adds a rug plot on the x-axis of variable x from `data`
`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`

Regular Expressions

Operator	Description	Operator	Description
`.`	Matches any character except `\n`	`*`	Matches preceding character/group zero or more times
`\`	Escapes metacharacters	`?`	Matches preceding character/group zero or one times
`|`	Matches expression on either side of expression; has lowest priority of any operator	`+`	Matches preceding character/group one or more times
`\d`, `\w`, `\s`	Predefined character group of digits (0-9), alphanumerics (a-z, A-Z, 0-9, and underscore), or whitespace, respectively	`^`, `$`	Matches the beginning and end of the line, respectively
`\D`, `\W`, `\S`	Inverse sets of `\d`, `\w`, `\s`, respectively	`( )`	Capturing group used to create a 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`

Modified lecture example for capture groups:

```
import re
lines = '169.237.46.168 - - [26/Jan/2014:10:47:58 -0800] "GET ... HTTP/1.1"'
re.findall(r'\[\d+\/(\w+)\/\d+:\d+:\d+:\d+ .+\]', line) # returns ['Jan']
```

Function	Description
`re.match(pattern, string)`	Returns a match if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns a match 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)
`re.sub(pattern, repl, string)`	Returns `string` after replacing all occurrences of `pattern` with `repl`

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	$\hat{\theta}_0 = \bar{y} - \hat{\theta}_1\bar{x} \qquad \hat{\theta}_1 = r\frac{\sigma_y}{\sigma_x}$
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	$R(\theta) = \frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2$	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)	$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y}$
Ridge Regression L2 Regularization	$\frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2 + \alpha\lvert\lvert\theta\rvert\rvert_2^2$	Squared L2 Norm of $\theta \in \mathbb{R}^d$	$\lvert\lvert\theta\rvert\rvert_2^2 = \sum_{j=1}^{d}\theta_j^2$
Ridge regression estimate (closed form)	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\alpha I)^{-1}\mathbb{X}^T\mathbb{Y}$		
LASSO Regression L1 Regularization	$\frac{1}{n}\lvert\lvert\mathbb{Y} - \mathbb{X}\theta\rvert\rvert_2^2 + \alpha\lvert\lvert\theta\rvert\rvert_1$	L1 Norm of $\theta \in \mathbb{R}^d$	$\lvert\lvert\theta\rvert\rvert_1 = \sum_{j=1}^{d}\lvert\theta_j\rvert$

Scikit-Learn

Package: `sklearn.linear_model`			
Linear Regression	**Logistic Regression**	**Function(s)**	**Description**
✓	-	`LinearRegression(fit_intercept=True)`	Returns an ordinary least squares Linear Regression model.
-	✓	`LogisticRegression( fit_intercept=True, penalty='l2', C=1.0)`	Returns an ordinary least squares Linear Regression model. Hyperparameter C is inverse of regularization parameter, C = 1/λ.
✓	-	`LassoCV()`, `RidgeCV()`	Returns a Lasso (L1 Regularization) or Ridge (L2 regularization) linear model, respectively, and picks the best model by cross validation.
✓	✓	`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.predict_proba(X)`	Returns predicted probabilities for the `X` passed in according to the fitted `model`. If binary classes, will return probabilities for both class 0 and 1.
✓	✓	`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`.

Package: `sklearn.model_selection`		
	Function	**Description**
	`train_test_split(*arrays, test_size=0.2)`	Returns two random subsets of each array passed in, with 0.8 of the array in the first subset and 0.2 in the second subset.

Probability

Let X have a discrete probability distribution $P(X = x)$. X has expectation $\mathbb{E}[X] = \sum_x xP(X = x)$ over all possible values x, variance $\text{Var}(X) = \mathbb{E}[(X - \mathbb{E}[X])^2]$, and standard deviation $\text{SD}(X) = \sqrt{\text{Var}(X)}$.

The covariance of two random variables X and Y is $\mathbb{E}[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])]$. If X and Y are independent, then $\text{Cov}(X, Y) = 0$.

Notes	Property of Expectation	Property of Variance
X is a random variable.		$\text{Var}(X) = E[X^2] - (E[X])^2$
X is a random variable, $a, b \in \mathbb{R}$ are scalars.	$\mathbb{E}[aX + b] = a\mathbb{E}[X] + b$	$\text{Var}(aX + b) = a^2\text{Var}(X)$
X, Y are random variables.	$\mathbb{E}[X + Y] = \mathbb{E}[X] + \mathbb{E}[Y]$	$\text{Var}(X + Y) = \text{Var}(X) + \text{Var}(Y) + 2\text{Cov}(X, Y)$
X is a Bernoulli random variable that takes on value 1 with probability p and 0 otherwise.	$\mathbb{E}[X] = p$	$\text{Var}(X) = p(1 - p)$

Central Limit Theorem

Let $(X_1, \ldots, X_n)$ be a sample of independent and identically distributed random variables drawn from a population with mean μ and standard deviation σ. The sample mean $\overline{X}_n = \sum_{i=1}^{n} X_i$ is normally distributed, where $\mathbb{E}[\overline{X}_n] = \mu$ and $\text{SD}(\overline{X}_n) = \sigma/\sqrt{n}$.

Parameter Estimation and Gradient Descent

Parameter Estimation

Suppose for each individual with fixed input x, we observe a random response $Y = g(x) + \epsilon$, where g is the true relationship and ϵ is random noise with zero mean and variance σ^2.

For a new individual with fixed input x, define our random prediction $\hat{Y}(x)$ based on a model fit to our observed sample $(\mathbb{X}, \mathbb{Y})$. The model risk is the mean squared prediction error between Y and $\hat{Y}(x)$: $\mathbb{E}[(Y - \hat{Y}(x))^2] = \sigma^2 + \left(\mathbb{E}[\hat{Y}(x)] - g(x)\right)^2 + \mathrm{Var}(\hat{Y}(x))$.

Suppose that input x has p features and the true relationship g is linear with parameter $\theta \in \mathbb{R}^{p+1}$. Then $Y = f_\theta(x) = \theta_0 + \sum_{j=1}^{p} \theta_j x_j + \epsilon$ and $\hat{Y} = f_{\hat{\theta}}(x)$ for an estimate $\hat{\theta}$ fit to the observed sample $(\mathbb{X}, \mathbb{Y})$.

Gradient Descent

Let $L(\theta, \mathbb{X}, \mathbb{Y})$ be an objective function to minimize over θ, with some optimal $\hat{\theta}$. 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 $\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_\theta L(\theta^{(t)}, \mathbb{X}, \mathbb{Y})$, where $\nabla_\theta L(\theta^{(t)}, \mathbb{X}, \mathbb{Y})$ is the partial derivative/gradient of L with respect to θ, evaluated at $\theta^{(t)}$.

SQL

SQLite syntax:

```
SELECT [DISTINCT]
    {* | expr [[AS] c_alias]
    {,expr [[AS] c_alias] ...}}
FROM tableref {, tableref}
[[INNER | LEFT ] JOIN table_name
    ON qualification_list]
[WHERE search_condition]
[GROUP BY colname {,colname...}]
[HAVING search_condition]
[ORDER BY column_list]
[LIMIT number]
[OFFSET number of rows];
```

Syntax	Description
`SELECT column_expression_list`	List is comma-separated. Column expressions may include aggregation functions (`MAX`, `FIRST`, `COUNT`, `AVG`, etc). `AS` renames columns. `DISTINCT` selects only unique rows.
`FROM s INNER JOIN t ON cond`	Inner join tables `s` and `t` using `cond` to filter rows; the `INNER` keyword is optional.
`FROM s LEFT JOIN t ON cond`	Left outer join of tables `s` and `t` using `cond` to filter rows.
`FROM s, t`	Cross join of tables `s` and `t`: all pairs of a row from `s` and a row from `t`
`WHERE a IN cons_list`	Select rows for which the value in column `a` is among the values in a `cons_list`.
`ORDER BY RANDOM LIMIT n`	Draw a simple random sample of `n` rows.
`ORDER BY a, b DESC`	Order by column `a` (ascending by default) , then `b` (descending).
`CASE WHEN pred THEN cons ELSE alt END`	Evaluates to `cons` if `pred` is true and `alt` otherwise. Multiple `WHEN`/`THEN` pairs can be included, and `ELSE` is optional.
`WHERE s.a LIKE 'p'`	Matches each entry in the column `a` of table `s` to the text pattern `p`. The wildcard `%` matches at least zero characters.
`LIMIT number`	Keep only the first `number` rows in the return result.
`OFFSET number`	Skip the first `number` rows in the return result.

Principal Component Analysis (PCA)

The i-th Principal Component of the matrix X is defined as the i-th column of $U\Sigma$ defined by Singular Value Decomposition (SVD).

$X = U\Sigma V^T$ is the SVD of X if U and V^T are matrices with orthonormal columns and Σ is a diagonal matrix. The diagonal entries of Σ, $[s_1, \ldots, s_r, 0, \ldots, 0]$, are known as singular values of X, where $s_i > s_j$ for $i < j$ and $r = \text{rank}(X)$.

Define the design matrix $X \in \mathbb{R}^{n \times p}$. Define the total variance of X as the sum of individual variances of the p features. The amount of variance captured by the i-th principal component is equivalent to s_i^2/n, where n is the number of datapoints.

Syntax	Description
`np.linalg.svd(X, full_matrices = True)`	SVD of `X` with shape `(M, N)` that returns `u`, `s`, `vt`, where `s` is a 1D array of `X`'s singular values. If `full_matrices=True`, `u` and `vt` have shapes `(M, M)` and `(N, N)` respectively; otherwise shapes are `(M, K)` and `(K, N)`, respectively, where `K = min(M, N)`.

Classification and Logistic Regression

Confusion Matrix

Columns are the predicted values $\hat{y}$ and rows are the actual classes y.

	$\hat{y} = 0$	$\hat{y} = 1$
$y = 0$	True negative (TN)	False Positive (FP)
$y = 1$	False negative (FN)	True Positive (TP)

Classification Performance

Suppose you predict n datapoints.

Metric	Formula	Other Names
Accuracy	$\frac{TP+TN}{n}$	
Precision	$\frac{TP}{TP+FP}$	
Recall/TPR	$\frac{TP}{TP+FN}$	True Positive Rate, Sensitivity
FPR	$\frac{FP}{FP+TN}$	False Positive Rate, Specificity

An ROC curve visualizes TPR vs. FPR for different thresholds T.

Logistic Regression Model: For input feature vector x, $\hat{P}_\theta(Y = 1|x) = \sigma(x^T\theta)$, where $\sigma(z) = 1/(1 + e^{-z})$. The estimate $\hat{\theta}$ is the parameter θ that minimizes the average cross-entropy loss on training data. For a single datapoint, define cross-entropy loss as $-[y \log(p) + (1 - y) \log(1 - p)]$, where p is the probability that the response is 1.

Logistic Regression Classifier: For a given input x and trained logistic regression model with parameter θ, compute $p = \hat{P}(Y = 1|x) = \sigma(x^T\theta)$. predict response $\hat{y}$ with classification threshold T as follows:

$$\hat{y} = \text{classify}(x) = \begin{cases} 1 & p \geq T \\ 0 & \text{otherwise} \end{cases}$$

Clustering

K-Means Clustering: Pick an arbitrary k, and randomly place k "centers", each a different color. Then repeat until convergence:

1. Color points according to the closest center (defined as squared distance).
2. Move center for each color to center of points with that color.

K-Means minimizes inertia, defined as the sum of squared distances from each datapoint to its center.

Agglomerative Clustering: Assign each datapoint to its own cluster. Then, recursively merge pairs of clusters together until there are k clusters remaining.

A datapoint's **silhouette score** S is defined as $S = (B - A)/\max(A, B)$, where A is the mean distance to other points in its cluster, and B is the mean distance to points in its closest cluster.