Data C100, Final 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 **80 points** spread out over **7 questions** and must be completed in the **110 minute** time period on August 10, 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** the box/circle to mark your answer. Do not use checkmarks or "$\times$"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: ____________________

1 Weepings of the Realm [10 Pts]

This question involves SQL databases. All code for this question 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.

Link, a heroic knight, has turned to data science to help him optimize his adventures.

He creates the SQL table `quests` to record the quests he has been offered on his journeys. The first few rows of `quests` are shown to the right. The table contains three columns:

`name`: the name of the person offering the quest
`location`: the location of the quest
`reward`: the monetary reward for completing the quest

name	location	reward
Hestu	Woodlands	10
Impa	Village	50
Sidon	Lake	20
Tulin	Mountain	5
Paya	Village	45
NULL	Underground	100

(a) [4 Pts] Link, being an honorable hero, only wants to accept quests from people he knows. Write a SQL query to return the name, location, and reward associated with the **one** quest with the **highest reward** offered by someone with a **non-NULL name**. The desired output is displayed below to the right.

```
SELECT *
FROM quests
____________________A____________________
____________________B____________________
____________________C____________________;
```

name	location	reward
Impa	Village	50

(i) Fill in Blank A:

Solution: `WHERE name IS NOT NULL`

(ii) Fill in Blank B:

Solution: `ORDER BY reward DESC`

(iii) Fill in Blank C:

Solution: `LIMIT 1`

Link is an honorable *and* practical hero.

He creates a second table, `travel`, to store the travel cost associated with journeying to each location. The first few rows of `travel` are shown to the right.

location	travel_cost
Village	3
Lake	25
Woodlands	10
Mountain	50

(b) [6 Pts] Fill in the blanks below to write a SQL query that returns a table with **two columns**:

`location`: a location with one or more quests
`avg_reward`: the average reward of all quests in that location

Only include locations **where the total reward** of all quests in the location is **greater** than the travel cost of going to that location. If a location is not present in one or both of Link's tables, this location should be excluded.

```
SELECT q.location, _________A_________ AS avg_reward
FROM quests AS q _________B_________ travel AS t
ON q.location = t.location
_______________C_______________
_______________D_______________;
```

(i) Fill in Blank A:

Solution: `AVG(reward)`

(ii) Fill in Blank B:

Solution: `INNER JOIN` (or `JOIN`, which defaults to an inner join)

(iii) Fill in Blank C:

Solution: `GROUP BY q.location` or `GROUP BY t.location`

(iv) Fill in Blank D:

Solution: `HAVING SUM(reward) > travel_cost`

2 Yodel Model [12 Pts]

It's time for the 100th annual Berkeley Yodeling Competition! Competitors will perform their best yodels (a type of singing) to win fame and glory. Too eager to wait for the final results, you decide to apply your modeling skills to predict the length of each competitor's longest yodel.

(a) [4 Pts] You collect the following training dataset of yodel lengths (y), time spent practicing (x_1), and lung capacity (x_2). You decide to fit a **constant model** $\hat{y} = \theta_0$ using **absolute (L1) loss** to compute the optimal parameter $\hat{\theta}_0$.

x_1	x_2	y
1	3	1
3/2	5	1
4	1	4

What is the **root mean squared error (RMSE)** of the constant model on the training data above when using the optimal parameter $\hat{\theta}_0$? Simplify your answer as much as possible.

Solution:

As shown in lecture, the optimal constant model parameter under L1 loss is the median of the target variable: $\hat{\theta} = \text{median}(y)$. This means that our prediction, $\hat{y}$, for each observation is the median of the provided y values: 1.

To determine the RMSE, we compute the squared error for each observation, then find the mean value across all three points and take the square root.

x_1	x_2	y	$(y-\hat{y})^2$
1	3	1	$(1-1)^2 = 0$
3/2	5	1	$(1-1)^2 = 0$
4	1	4	$(4-1)^2 = 9$

The RMSE is then $\sqrt{\frac{0+0+9}{3}} = \sqrt{3}$.

(b) [2 Pts] Your friend wants to create their own model and fit it using **ordinary least squares (OLS)**. For which of the following loss (empirical risk) functions can OLS be used to compute the optimal value of the parameters θ_i? Assume that x_{i1} and x_{i2} represent the first and second feature, respectively, of the ith observation.

- ☐ **A.** $R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_0)^2$
- ☐ B. $R(\theta) = \frac{1}{n}\sum_{i=1}^{n}|y_i - \theta_0 - \theta_1 x_{i1}|$
- ☐ **C.** $R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_1 x_{i1} - \theta_2 x_{i2})^2$
- ☐ **D.** $R(\theta) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \theta_1|x_{i1}| - \theta_2 x_{i2})^2$

Solution: The geometric properties of OLS assume that we are working with a **linear** model (with respect to θ) and **squared loss**. This means that we can only use OLS to minimize a loss function that includes a squared error term and no non-linear transformations of any parameter θ_i.

(c) [2 Pts] Your friend decides to take a different approach and instead fit a model using the following training data. Note that this dataset is identical to the original data on the previous page, with the addition of two new features x_3 and x_4.

x_1	x_2	x_3	x_4	y
1	3	2	1/2	1
3/2	5	2	7	1
4	1	8	3	4

Which of the following is/are true when fitting a model to this training data?

☐ **A. A linear regression model fitted using OLS would achieve a training mean squared error (MSE) of 0**

☐ **B. A linear regression model fitted using OLS would achieve a training mean absolute error (MAE) of 0**

☐ C. A ridge regression model with $\lambda > 0$ would achieve a training MSE of 0

☐ D. A ridge regression model with $\lambda > 0$ would achieve a training MAE of 0

☐ E. None of the above

Solution: Notice that the target y is a multiple of the feature x_3. This means that we can create a linear model of the form $\hat{y} = \frac{1}{2}x_3$ to perfectly predict each y value when $\hat{\theta}_3 = \frac{1}{2}$. In other words, the target vector $\mathbb{Y}$ lies in the span of the design matrix $\mathbb{X}$.

In OLS, there are no restrictions placed on model complexity; hence, any value of θ can be selected for the model. The OLS estimate will set $\hat{\theta}_3 = \frac{1}{2}$, leading to perfect predictions where $\hat{y} = y$ for all three datapoints. This means that the training MSE and MAE are both zero.

In ridge regression, the addition of the L2 regularization term to the objective function penalizes the model for selecting large magnitudes of θ, so we do not achieve the OLS optimal solution for $\hat{\theta}$. The ridge regression estimate for $\hat{\theta}_3$ will be set to some value other than $\frac{1}{2}$, so MSE and MAE will be non-zero.

(d) [4 Pts] You would rather use **gradient descent (GD)** to fit a model. You define the following loss function with one parameter θ_0. You decide to first sample a single datapoint from your dataset. You then run gradient descent considering **only this sampled datapoint** at **every** GD update step. This single sampled datapoint is displayed below.

$$L(\theta_0) = (y - \theta_0)^4$$

x_1	x_2	x_3	x_4	y
1	3	2	1/2	1

Which of the following is/are true when using gradient descent to compute the optimal model parameter $\hat{\theta}_0$ with a learning rate of $\alpha = \frac{1}{4}$ for your new choice of loss function and data? Assume a starting guess of $\theta_0^{(0)} = 0$.

☐ A. The loss function is non-convex, so gradient descent won't converge to a solution

☐ B. The procedure described above is stochastic gradient descent

☐ **C. The GD estimate for $\hat{\theta}_0$ will not update after the first iteration of the algorithm**

☐ **D. The solution for $\hat{\theta}_0$ produced by this gradient descent procedure achieves a training MSE of 0 on this single datapoint**

Solution: The convexity of a function can be tested by examining its second derivative. $\frac{d^2L}{d\theta_0^2} = 12(y - \theta_0)^2$, which is guaranteed to always be a positive number. This means that the loss function is convex, so A is incorrect.

In stochastic gradient descent, a new datapoint is sampled at *each* update of the estimate for θ. The procedure above uses the same sampled datapoint for all updates, so it is not stochastic gradient descent. Hence, B is incorrect.

To determine the gradient descent update for this loss function, find the derivative of loss with respect to the parameter θ_0, then plug it into the update rule: $\theta_0^{(t+1)} = \theta_0^{(t)} - \alpha\nabla_{\theta_0}L$.

$$\frac{dL}{d\theta_0} = -4(y - \theta_0)^3$$

$$\theta_0^{(t+1)} = \theta_0^{(t)} - 4\alpha(y - \theta_0^{(t)})^3$$

Now, substitute $\alpha = \frac{1}{4}$ and $\theta_0^{(0)} = 0$:

$$\theta_0^{(1)} = \theta_0^{(0)} - 4\frac{1}{4}(1 - \theta_0^{0})^3$$

$$\theta_0^{(1)} = 0 - 4\frac{1}{4}(1 - 0)^3 = 1$$

$\theta_0 = 1$ is the minimizing parameter value, so any subsequent gradient descent updates yield the same result (this can be confirmed by either computing $\theta_0^{(2)}$ or recognizing that $L(1) = (1 - 1)^4 = 0$). Therefore, C is true. When the constant model takes the form $\theta_0 = 1$, we perfectly predict the observation $y = 1$, making the MSE 0. This means that D is true.

3 Son of a Pitch(fork) [18 Pts]

Pitchfork is a music blog that reviews and rates music albums. For each music album, Pitchfork provides a rating (on a scale of 0 to 10), a review article, and a whether or not it deems the record "Best New Music". Adrianne obtains a DataFrame `reviews` that contains all reviews Pitchfork has published from 1997 to 2021. `reviews` contains 26,036 rows and 5 columns. The columns are:

`"URL"`: Simplified URLs of the review blog post, stored as `str`

`"Score"`: Pitchfork's rating of the record, on a scale of 0 to 10, stored as `float`

`"Best New Music?"`: Pitchfork's decision on whether or not the record is "the finest music of the current moment." Only possible values are integers `1` or `0`, with `1` representing "Yes" and `0` representing "No"

`"Review body"`: The review article, stored as `str`

`"Genre"`: the genre of the album, stored as `str`

The first 7 rows of `reviews` are shown below.

	URL	Score	Best New Music?	Review body	Genre
0	/reviews/albums/Eminem/928/Revival	5.0	0	Few texts in hip-hop...	Rap
1	/reviews/albums/Radiohead/2910/In_Rainbows	9.3	1	Like many music love...	Rock
2	/reviews/albums/Nicki_Minaj/6688/The_Pinkprint	7.5	0	Nicki Minaj is fed u...	Rap
3	/reviews/albums/Sufjan_Stevens/8020/Illinois	9.2	1	The best travel writ...	Folk/Country
4	/reviews/albums/Big_Thief/11808/U.F.O.F.	9.2	1	Big Thief are their ...	Rock
5	/reviews/albums/Ed_Sheeran/12622/÷	2.8	0	Ed Sheeran needs you...	Pop/R&B
6	/reviews/albums/100_gecs/18330/1000_gecs	7.4	0	If you're the type t...	Experimental

(a) [4 Pts] Adrianne realizes that there is no column for the title or the artist of each album. She notices that information about the title and artist is stored in the `"URL"` column of `reviews`. In particular, each entry in the `"URL"` column is in the format:

```
/reviews/albums/[artist]/[review id]/[album title]
```

You may assume that:

- The `review id` is a unique identifier for the review blogs, from 0 to 26035
- Spaces in the artists' names and record titles are replaced by underscores (`"_"`)
- There are no backslashes (`"/"`) in the artists' names and titles.

For example, the first entry in the previewed DataFrame above corresponds to an album named "Revival" from the artist "Eminem".

Complete the skeleton code to create a DataFrame named `artist_title` with two columns, one containing all the artists and one containing all the album titles.

```
pat = r"/reviews/albums/_____A_____"
artist_title = reviews["URL"].___B___
artist_title
```

The desired output for the first 7 rows of `reviews` is shown on the right.

	0	1
0	Eminem	Revival
1	Radiohead	In_Rainbows
2	Nicki_Minaj	The_Pinkprint
3	Sufjan_Stevens	Illinois
4	Big_Thief	U.F.O.F.
5	Ed_Sheeran	÷
6	100_gecs	1000_gecs

Note: you **do not** need to consider any edge cases beyond the example URLs shown in the first seven rows of `reviews`. As long as your code works for the `"URL"` strings shown in the DataFrame, and is not trivializing the problem, you are eligible for full credit.

(i) Fill in blank `A`:

Solution: `albums/([^/]+)/\d+/([^/]+)` or any equivalent pattern

(ii) Fill in blank `B`:

Solution: `str.extract(pat)`

(b) [3 Pts] Define the **length** of a review article as the number of characters in that article.

Create a DataFrame named `genre_lengths` that is **indexed** by `"Genre"` and contains **two columns**: `"Length"` and `"Review body"`. The `"Review body"` column should contain the review article that has the **longest length** out of all reviews for the corresponding genre. `"Length"` should contain the length of this longest review for the genre.

You may use as many or as few of the provided blanks below as you believe are necessary.

Solution:

```
reviews["Length"] = reviews["Review body"].str.len()
reviews = reviews.sort_values("Length", ascending=False)
genre_lengths = reviews.groupby("Genre") \
                          [["Length", "Review body"]].first()
```

Or any equivalent code. Note that we cannot simply take the maximum value of each group because we wish to return both the longest review *and* the review body text – it is not possible to take the max of a string.

Buck is also interested in the dataset and asks Adrianne to share it. Adrianne draws 100 records from `reviews` and gives the sample to Buck. For **all remaining parts** of this question, assume the records were drawn **uniformly at random with replacement** and **independently** of one another.

(c) [2 Pts] Let X_i denote the value of the `"Best New Music?"` column for entry i in Buck's sample, for $i \in \{1, 2, 3, \ldots, 100\}$. For example, if row 6 from the DataFrame on Page 11 is entry j in the sample, then $X_j = 0$.

`"Best New Music?"`

population mean	0.1
population SD	0.2

Adrianne also gives Buck some population parameters for this column in `reviews`, displayed above.

What is the distribution of $S = \sum_{i=1}^{10} X_i$? State the distribution's **name** and **parameters**.

Solution: Binomial$(10, 0.1)$

Each X_i can be thought of as a Bernoulli random variable. The population mean (an average of many 1 and 0 values) is equivalent to the proportion of the population that has a value of 1 – in other words, the probability that $X_i = 1$. The variable S therefore represents the sum of 10 Bernoulli variables with 0.1 probability of success, which is the Binomial distribution of $n = 10$ trials and $p = 0.1$.

(d) [3 Pts] Adrianne also gives Buck some **population parameters** for the `"Score"` column in `reviews`, displayed on the right. Let Y_i denote the `"Score"` value of entry i in Buck's sample, for $i \in \{1, 2, 3, \ldots, 100\}$.

`"Score"`

population mean	7.1
population SD	1.2

Assume the underlying data generating process for the `"Score"` column is a constant model: $Y = \theta + \epsilon$, where $\theta = 7.1$ is a fixed population parameter and ϵ is a random noise with expectation 0 and variance 1.44.

Buck uses the following model to fit his sample of Y_i's: $\hat{Y} = \hat{\theta}$.

Buck estimates θ using the mean of the first 10 `"Score"` values: $\hat{\theta} = \frac{1}{10}\sum_{i=1}^{10} Y_i$. What is his **model bias**? Show all work and simplify your answer as much as possible.

Solution: Recall the definition of model bias:

$$\begin{aligned}\text{model bias} &= E[\hat{Y} - \theta] \\ &= E[\hat{Y}] - \theta \\ &= E\left[\frac{1}{10}\sum_{i=1}^{10} Y_i\right] - 7.1 \\ &= \frac{1}{10}\sum_{i=1}^{10} E[Y_i] - 7.1 \\ &= 7.1 - 7.1 = 0.\end{aligned}$$

(e) [3 Pts] What is his **model variance**? Simplify your answer as much as possible.

Solution: Recall the definition of model variance:

$$\begin{aligned}\text{model variance} &= \text{Var}(\hat{Y}) \\ &= \text{Var}\left(\frac{1}{10}\sum_{i=1}^{10} Y_i\right) \\ &= \frac{1}{10^2}\sum_{i=1}^{10} \text{Var}(Y_i) && \text{since } Y_i\text{'s are i.i.d.} \\ &= \frac{1}{10^2} 10 \cdot 1.44 \\ &= \frac{1.44}{10}.\end{aligned}$$

(f) [3 Pts] What is his **model risk**? Simplify your answer as much as possible.

Hint: You might find the results from the previous two parts helpful.

Solution: Recall the decomposition of model risk:

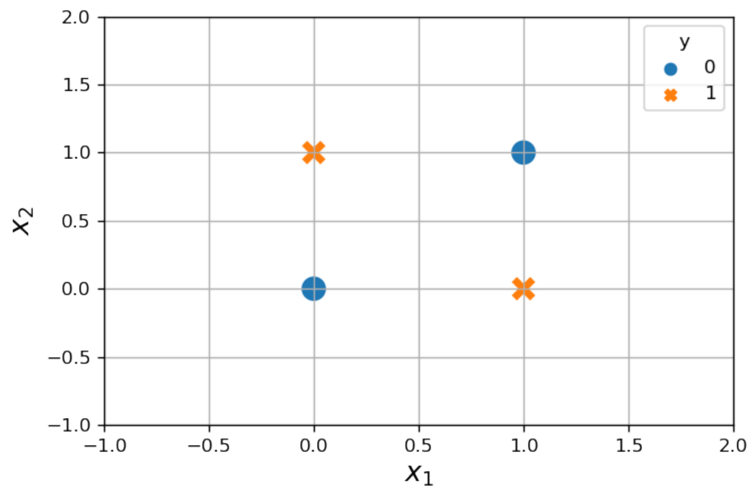
$$\begin{aligned}\text{model risk} &= \text{observation variance} + (\text{model bias})^2 + \text{model variance} \\ &= \text{Var}(\epsilon) + 0^2 + \frac{1.44}{10} \\ &= 1.44 + \frac{1.44}{10}\end{aligned}$$

4 Classifying XOR [12 Pts]

(a) [2 Pts] We want to use **logistic regression** to train a binary classifier on four data points with two features: x_1 and x_2.

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

(i) Mark the four data points in the empty plot below. Write down the corresponding class (0 or 1) next to each point.



(ii) Based on the plot from the previous part, is the dataset linearly separable?

○ Yes [✓ ○ **No**]

There is no straight line that can be drawn in terms of the features x_1 and x_2 that perfectly separates the two classes.

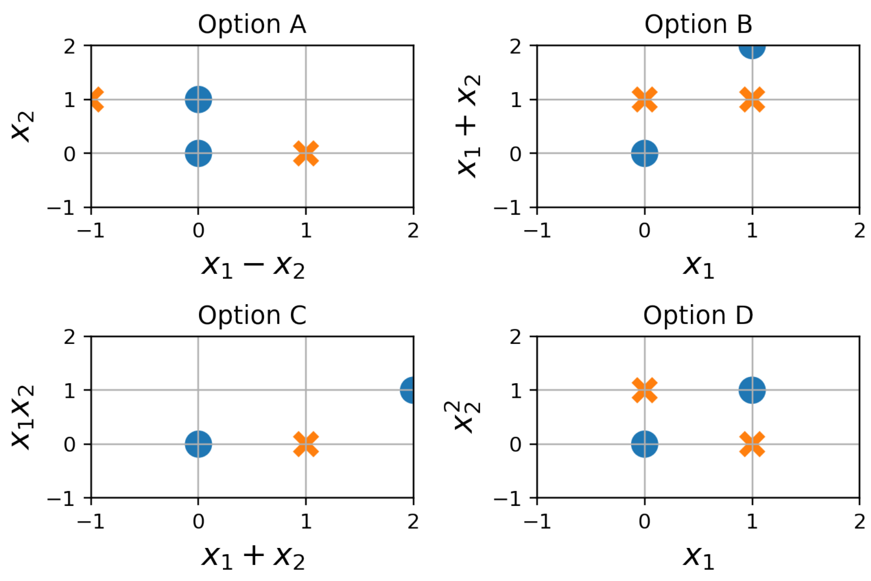
(b) [2 Pts] We consider generating new features by transforming the original features x_1 and x_2 from Part (a). Which set of features leads to a **linearly separable** dataset?

○ A. $\{x_1 - x_2,\ x_2\}$

○ B. $\{x_1,\ x_1 + x_2\}$

[✓ ○ **C.** $\{x_1 + x_2,\ x_1x_2\}$]

○ D. $\{x_1,\ x_2^2\}$



Option C is the only choice where a hyperplane can be used to separate the classes using a line in terms of the features.

(c) [4 Pts] We transform the original data to generate the new design matrix displayed below. We create a logistic regression model using a bias term, x_3, and x_4.

bias	x_3	x_4	y
1	1.0	-1.5	0
1	1.5	-2.0	0
1	0.5	3.0	1
1	0.0	1.0	0

We find that the optimal parameter vector for this model is as follows:

$$\hat{\theta} = [\log(1/2), \log(4/3), \log(1/3)]^\top$$

Now we want to classify a new data point: $x_* = [1, 1, -1]$. You may assume this new data point x_* is expressed in terms of the transformed features: [bias, x_3, x_4].

(i) Find the **predicted probability that point x_* is in class 1** $P(y_* = 1|x_*)$ and simplify your result as much as possible. Show all work for full credit. Recall that $\log$ denotes the natural logarithm with base e. Hint: $\log(a) + \log(b) = \log(ab)$ and $-\log(a) = \log(\frac{1}{a})$

Solution:

$$\begin{aligned} P(y_*|x_*) = \sigma(x_*^\top \hat{\theta}) &= \frac{1}{1+e^{-x_*^\top \hat{\theta}}} && (1) \\ &= \frac{1}{1+e^{-\log(1/2) - \log(4/3) + \log(1/3)}} && (2) \\ &= \frac{1}{1+e^{\log(1/2)}} && (3) \\ &= \frac{1}{1+1/2} && (4) \\ &= \frac{2}{3} && (5) \end{aligned}$$

(ii) For which threshold values T would the predicted class for x_* be $\hat{y}_* = 1$?

☐ **A. 0** ☐ **B. 0.3** ☐ C. 0.8 ☐ D. 1

The predicted class is $\hat{y} = 1$ whenever the predicted probability is greater than the threshold value T. Hence, we select all thresholds less than the predicted probability computed above.

(d) [2 Pts] Now that our classifier has been trained, we wish to assess its performance via classification metrics. Select all of the **correct statements** from the list below.

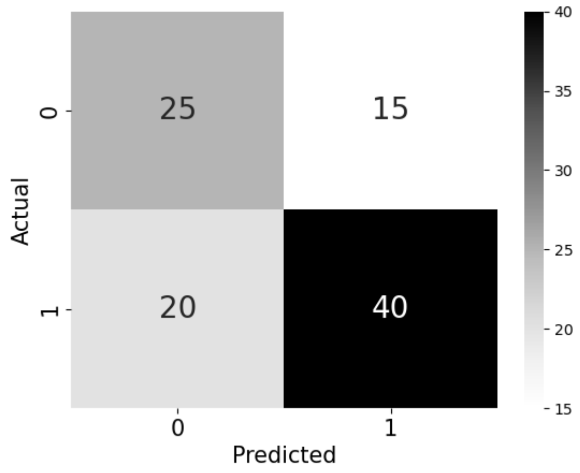
- ☐ **A. The area under a receiver operating characteristic (ROC) curve is always greater than or equal to 0.5**
- ☐ **B. The ROC curve of the ideal classifier passes through the (0, 1) point**
- ☐ C. Decreasing the threshold T can never decrease the number of true negative predictions
- ☐ **D. Increasing the threshold T can never decrease the number of false negative predictions**

Solution: The ROC curve of a perfect classifier passes through the upper lefthand corner of the plot at position (0, 1). The ROC curve of a random classifier passes diagonally through the plot, with AUC 0.5.

Any datapoint with a predicted probability greater than T will be classified as $\hat{y} = 1$. This means that as T decreases, the number of positive predictions will increase or stay constant, while the number of negative predictions will decrease or stay constant. With fewer negative predictions made overall, it is possible that the number of true negative predictions will decrease. Hence, C is incorrect.

As T increases, the number of positive predictions will decrease or stay constant, while the number of negative predictions will increase or stay constant. Because the total number of negative predictions is non-decreasing, the number of false negatives cannot increase. Hence, D is correct.

(e) [2 Pts] We use the fitted logistic regression classifier from Part (c) to predict the labels of 100 new data points. The **confusion matrix** corresponding to the predictions of this classifier on the new data points is presented below.



Compute the **recall** of this classifier. Show all work for full credit.

Solution:

$$\begin{aligned} \text{Recall} &= \frac{TP}{TP + FN} && (6) \\ &= \frac{40}{40 + 20} && (7) \\ &= \frac{2}{3} && (8) \end{aligned}$$

5 The Shape of U [11 Pts]

Suppose we have collected a design matrix of data X. We wish to use principal component analysis (PCA) to reduce the dimensionality of our data.

(a) [4 Pts] We use the **singular value decomposition (SVD)** to decompose the matrix X as follows:

$$X = U\Sigma V^\top = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 4 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1/\sqrt{2} & 0 & 1/\sqrt{2} \\ 0 & 1 & 0 \\ 1/\sqrt{2} & 0 & -1/\sqrt{2} \end{bmatrix}$$

Find the **second principal component (PC)** of X. Show all work for full credit.

Solution: Each column of $U\Sigma$ is a principal component.

$$U\Sigma = \begin{bmatrix} 0 & 2 & 0 \\ 4 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The second principal component is the second column of $U\Sigma$: $\begin{bmatrix} 2 \\ 0 \\ 0 \end{bmatrix}$

(b) [2 Pts] Which statements are **true** about principal component analysis performed on any matrix A with singular value decomposition $A = U\Sigma V^\top$? Consider the general case where U, Σ and $V^\top$ may not have the specific values given in the previous part.

☐ **A. All columns of the matrix A should have the same mean value to perform PCA**

☐ **B. Each principal component is a linear combination of the columns of A**

☐ **C. Each principal component is a linear combination of the columns of U**

☐ D. An $n \times d$ matrix A has d non-zero singular values

Solution: Before performing PCA, we must center the data such that each column has mean 0. Therefore, A is true.

Matrix multiplication can be interpreted as using the right matrix to create a linear combination of the columns of the left matrix. The principal components are computed as AV or $U\Sigma$. In both cases, we construct the principal component matrix by taking a linear combination of A or U. Hence, B and C are true.

The number of non-zero singular values for a matrix is equal to its rank. If an $n \times d$ matrix is not full rank, then it has fewer than d non-zero singular values.

(c) [3 Pts] Assume we store the matrix X from Part (a) as the NumPy array `X`. **Fill in the blanks** to create a **scree plot** for the variance ratio captured by each each principal component of X. Assume `numpy` and `matplotlib.pyplot` are imported as `np` and `plt`, respectively.

```
U, S, Vt = np.linalg.svd(X, full_matrices=False)
numbers = np.arange(len(S))
y = ______A______ / ______B______
plt.plot(x = numbers, y = y)
```

(i) Fill in blank `A`:

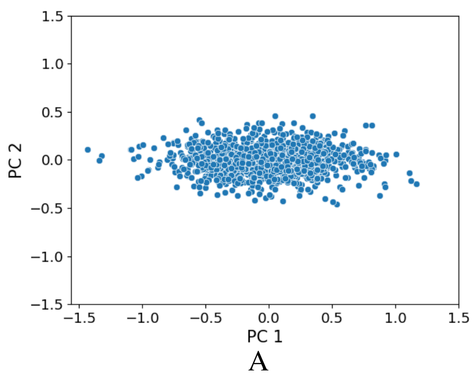
Solution: `S**2`

(ii) Fill in blank `B`:

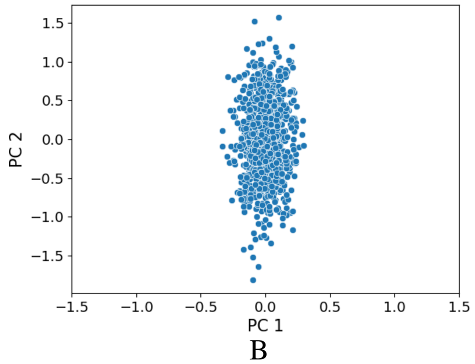
Solution: `sum(S**2)`

(d) [2 Pts] Suppose we project a new matrix A onto the **directions of its first two principal components**. Which of the following plots could **possibly** display the projected data with the first PC plotted along the **horizontal axis** and the second PC plotted along the **vertical axis**? Select all that apply.

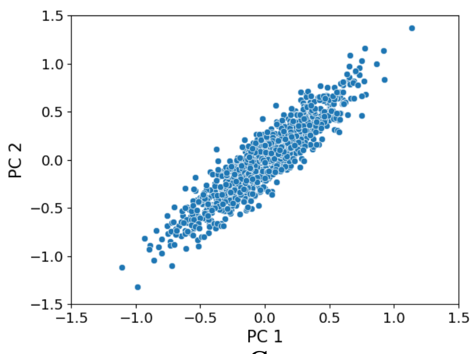
[✓ A] ☐ B ☐ C ☐ D



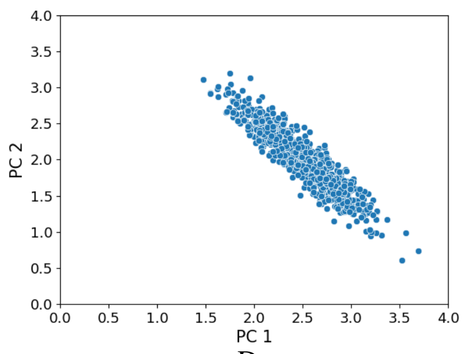
A



B



C



D

Solution: The first principal component captures the direction of most variance in the data, while the second principal component captures the direction of second-most variance. This means that a correct PC plot should be most ”spread out” along the horizontal axis (PC 1), and next most spread out along the vertical axis (PC 2). Only A satisfies this condition.

6 Yes or No [3 Pts]

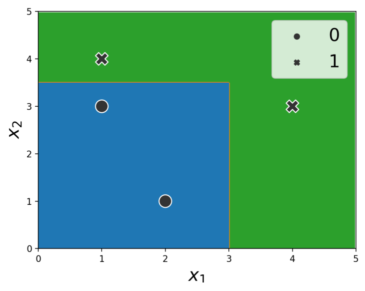
The following statements are about decision trees or random forests. Determine if each of them is true or false. Each subpart is worth 0.5 points.

For Parts (a) and (b), consider a decision tree fitted to the dataset of two classes (0 and 1) shown below.

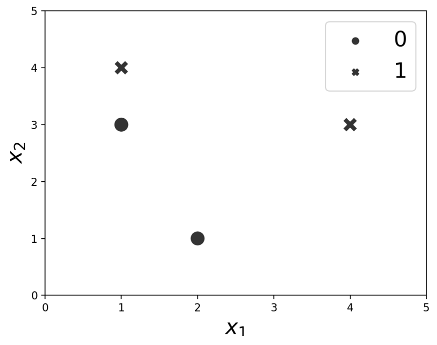
(a) A decision tree will generate a linear decision boundary on this dataset.

○ True [✓ **False**]

Solution: Looking at the data points, we cannot make a perfect classification using one of x_1 and x_2. The decision boundary is therefore not linear.



x_1	x_2	y
1	3	0
1	4	1
2	1	0
4	3	1



(b) The top (root) node of a decision tree for this dataset has an entropy of 1.

[✓ **True**] ○ False

Solution: There are two classes of points in the dataset: 0 and 1. The proportions of 0 and 1 are both $1/2$. Therefore, we can calculate the entropy using the formula:

$$S = -\sum_C p_C \log_2(p_C) = -(\frac{1}{2}\log_2\frac{1}{2}) - (\frac{1}{2}\log_2\frac{1}{2})$$
$$= -(\frac{1}{2}\cdot(-1)) - (\frac{1}{2}\cdot(-1)) = 1.$$

For Parts (c) to (f), consider the general behavior of decision trees and random forests on *any* data.

(c) A decision tree with no restrictions on maximum depth always achieves 100% training accuracy on a linearly separable dataset.

[✓ **True**] ○ False

Solution: From lecture, we know that decision trees (when unrestricted) will always have a perfect accuracy on the training data, except when there are samples from different categories with the exact same features. Since we specified the dataset is linearly separable, there will not be overlap between different classes.

(d) When training a decision tree, the maximum depth of the tree is a hyperparameter.

[✓ **True**] ○ False

Solution: Controlling the maximum depth is a form of limiting the complexity of decision trees. We don't decide this in the training process—we decide the maximum tree depth beforehand. This makes the maximum depth a hyperparameter.

(e) In any random forest, each tree uses a random subset of some number m features to train the entire tree.

○ True [✓ **False**]

Solution: In random forests, we use a random subset of m features in each **node** (not the entire tree) of each decision tree.

(f) A random forest will always achieve higher accuracy than a decision tree on the training set.

○ True [✓ **False**]

Solution: From lecture, decision trees will in general achieve close to 100% accuracy. However, it is not guranteed that random forest will always achieve 100% accuracy because of the randomness in both the Bootstrap and random feature selection. Therefore, we cannot say random forests always achieve higher accuracy than decision trees.

7 Cross(words)-Validation [13 Pts]

Ella, an avid crossword enthusiast, wants to build a model to help her analyze crossword puzzles.

(a) [2 Pts] Ella decides to build the following model with p numerical features and an intercept:

$$\hat{y} = \hat{\theta}_0 + \sum_{j=1}^{p} \hat{\theta}_j x_j$$

She uses **mean squared error with L1 regularization**, using a regularization hyperparameter of λ. Which of the following are correct objective function(s) for her model? Select all that apply.

- ☐ A. $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 + \lambda \sum_{j=1}^{p} \theta_j^2$
- ☐ B. $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| + \lambda \sum_{j=1}^{p}|\theta_j|$
- ☐ **C.** $\frac{1}{n}||\mathbb{Y} - \hat{\mathbb{Y}}||_2^2 + \lambda \sum_{j=1}^{p}|\theta_j|$
- ☐ D. $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| + \lambda \sum_{j=1}^{p} \theta_j^2$

Solution: This question tests understanding of the loss functions of regularization. Since we are using the **mean squared error**, the first part should be either $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$ or $\frac{1}{n}||\mathbb{Y} - \hat{\mathbb{Y}}||_2^2$. We can rule out options B and D. We are using L1 regularization, so the regularization part should be $\lambda \sum_{j=1}^{p}|\theta_j|$. We rule out A. The only correct option is C.

(b) [2 Pts] After performing feature engineering, Ella finds that the model's training error increases. Which of the following could **not** have been the reason? Select all that apply.

- ☐ A. Increasing the regularization penalty hyperparameter λ
- ☐ **B. Centering all features in the design matrix other than the bias column**
- ☐ **C. Adding polynomial features**
- ☐ D. Removing the intercept/bias column from the design matrix

Solution: Parts (b) and (c) test understanding of the bias-variance tradeoff curve: in general, increasing model complexity increases model variance and decreases model bias.
Option A is incorrect. Increasing the regularization penalty λ would decrease model complexity. As a result, this would increase model bias and equivalently the training error.
Option B is correct. Centering all the features besides the bias column would **not** affect the model and therefore the training error.
Option C is correct. Adding polynomial features would increase model complexity and therefore decrease model bias and training error.

Option D is incorrect. Removing the bias column decreases model complexity and therefore increases model bias and training error.

(c) [2 Pts] Ella is worried that the variance of her model may be too large. Which of the following will **not** increase Ella's model variance? Select all that apply.

- ☐ A. Decreasing the regularization penalty hyperparameter λ to 0
- [✓ **B. Increasing the number of data points used for training**]
- ☐ C. Using stochastic gradient descent to optimize the objective function instead of batch gradient descent
- ☐ D. Introducing a new categorical variable by adding one-hot encoded columns

Solution: Option A is incorrect. Decreasing the regularization penalty to 0 is equivalent to not regularizing the model; this increases model complexity and therefore increases model variance.
Option B is correct. Increasing the number of data points used for training will decrease model variance.
Option C is incorrect. Using stochastic GD instead of batch GD will increase model variance.
Option D is incorrect. Introducing a new column increases model complexity and therefore increases model variance.

(d) [2 Pts] Ella is not sure if she should use L1 or L2 regularization in her model. In which of the following scenarios is L1 regularization a **better** choice than L2? Select only one option.

- ○ A. When the columns of the design matrix are collinear
- [✓ **B. When we have many features but don't know which are most helpful for modeling**]
- ○ C. When we want a closed-form analytical solution for optimal model parameters
- ○ D. When we have more features than data points

Solution: Recall from lecture that L1 regularization has the tendency to set model coefficients to 0, so we can use L1 regularization to select useful features. L2 regularization does not such tendency. Therefore, L1 is better than L2 in this respect.
For scenarios in options A and D, L1 and L2 regularization will be similarly effective.
For the scenario in option C, L2 is better than L1 because L2 has a closed-form solution, but L1 does not.

Ella wants to use gradient descent to determine the optimal parameter values for her regularized model. To do so, she needs to select a value for α, the gradient descent learning rate.

She performs **4-fold cross-validation** using a dataset of 60 observations to help her choose from three possible values of α. The validation errors computed in each fold of her cross-validation (CV) procedure are shown below. Assume that she is not considering a test set, so all 60 observations are used in the cross-validation procedure.

	Fold #1	Fold #2	Fold #3	Fold #4
$\alpha = 0.1$	2	10	3	5
$\alpha = 1$	6	1	2	3
$\alpha = 10$	4	9	1	2

(e) [2 Pts] How many observations are in each validation fold of Ella's CV procedure?

Solution: In 4-fold cross-validation, each validation fold contains $1/4$ of the training data. This means that $60/4 = 15$ observations are in each validation fold.

(f) [3 Pts] Given the validation errors displayed above, which value of α should Ella choose?

○ A. $\alpha = 0.1$

[✓ ○ **B.** $\alpha = 1$]

○ C. $\alpha = 10$

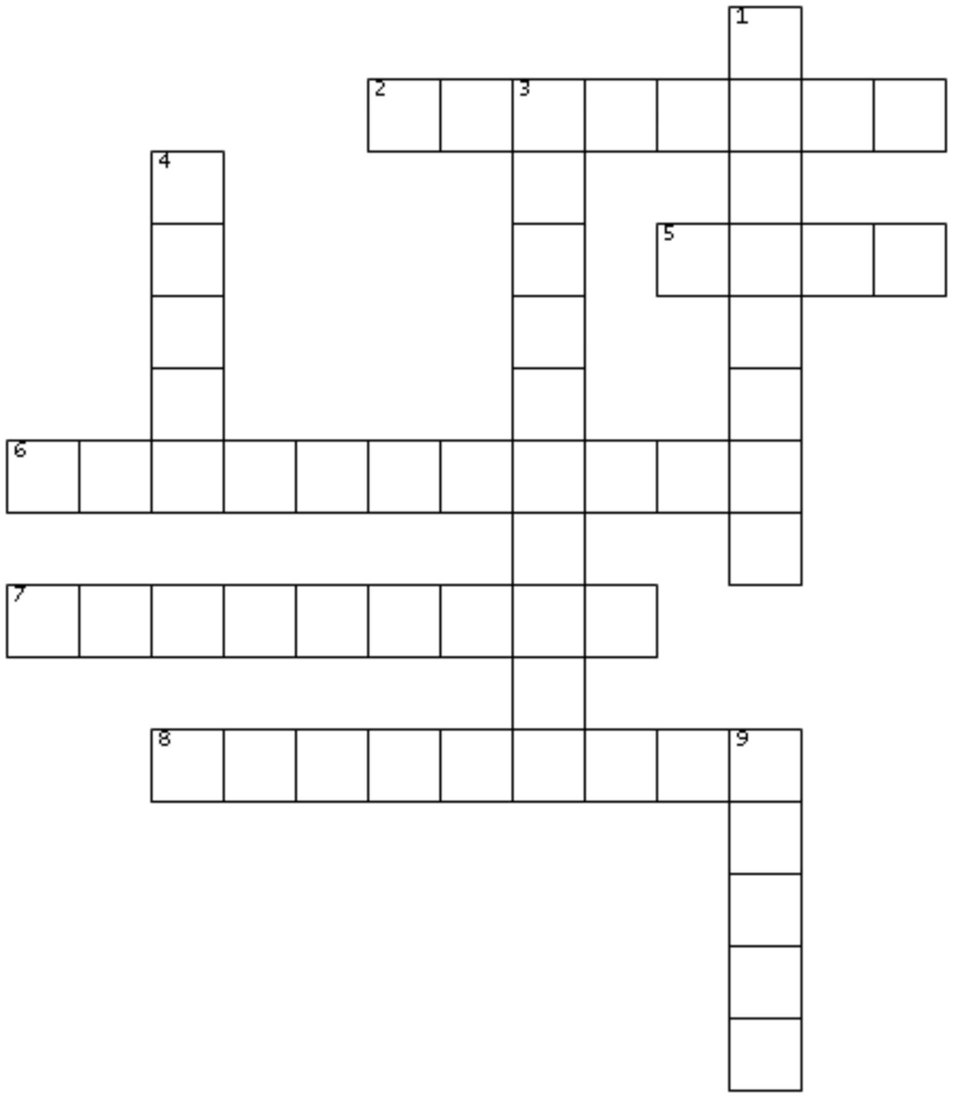
○ D. α should be determined using ordinary least squares, not cross-validation

Solution: Ella should select the value of α that results in the lowest cross-validation error, defined as the mean error across all validation folds. To determine the cross-validation error for each α value, find the row-wise average of the four validation errors for that choice of α.

	Fold #1	Fold #2	Fold #3	Fold #4	CV Error
$\alpha = 0.1$	2	10	3	5	5
$\alpha = 1$	6	1	2	3	3
$\alpha = 10$	4	9	1	2	4

The best choice of hyperparameter is $\alpha = 1$.

(g) [0 Pts] To celebrate completing her model, Ella creates the following crossword puzzle. Help her fill it out! Words can go across or down. Letters are shared when the words intersect. **This question is optional and worth 0 points.** You should only fill it out if you have completed all other questions in the exam.



Across

2. A metric for how much an estimator tends to vary from its mean value
5. A metric for how far off an estimator is from the true parameter
6. A tax system where the percentage tax rate is higher for individuals with higher net income
7. A numerical function of a random sample
8. A numerical function of a population

Down

1. When an estimator gets the population parameter right, on average
3. A tax system where the percentage tax rate is higher for individuals with lower net income
4. A regularization method that penalizes coefficients using L1 norm
9. A regularization method that penalizes coefficients using L2 norm

8 Congratulations [0 Pts]

Congratulations! You have completed the Final.

- **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.

Congrats on finishing the class! We're so happy to have spent the past few months with you.

[Optional, 0 pts] What did Data 100 "look" like this summer? Draw a picture!

Summer 2023 Data C100 Final 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.
`df.sample(n=1, replace=False)`	Returns `n` randomly sampled rows from `df`. By default, sampling is performed without replacement.

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 capturing group(s), outputs a DataFrame with one integer-named column for each group.
`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

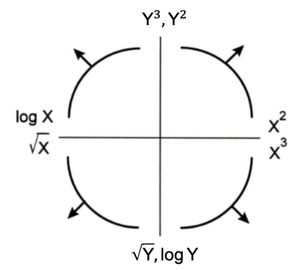
Function	**Description**
`re.match(pattern, string)`	Returns all matching characters if zero or more characters at beginning of `string` matches `pattern`, else None
`re.search(pattern, string)`	Returns all matching characters if zero or more characters anywhere in `string` matches `pattern`, else None
`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=False, 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	$\hat{\theta}_0=\bar{y}-\hat{\theta}_1\bar{x} \qquad \hat{\theta}_1=r\frac{\sigma_y}{\sigma_x}$
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	$R(\theta) = \frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2$	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)	$R^2 = \frac{\text{variance of fitted values}}{\text{variance of } y}$

Regularization

Concept	Formula	Concept	Formula
Ridge Regression L2 Regularization	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2 + \lambda\|\|\theta\|\|_2^2$	Squared L2 Norm of $\theta \in \mathbb{R}^p$	$\|\|\theta\|\|_2^2 = \sum_{j=1}^p \theta_j^2$
Ridge regression estimate (closed form)	$\hat{\theta}_{\text{ridge}} = (\mathbb{X}^T\mathbb{X} + n\lambda I)^{-1}\mathbb{X}^T\mathbb{Y}$		
LASSO Regression L1 Regularization	$\frac{1}{n}\|\|\mathbb{Y} - \mathbb{X}\theta\|\|_2^2 + \lambda\|\|\theta\|\|_1$	L1 Norm of $\theta \in \mathbb{R}^p$	$\|\|\theta\|\|_1 = \sum_{j=1}^p \|\theta_j\|$

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 $\theta^{(t)}$.

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:

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

Scikit-Learn

Package: `sklearn.linear_model`

Linear Regression	Logistic Regression	Class/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.
✓	-	`Lasso()`, `Ridge()`	Returns a Lasso (L1 Regularization) or Ridge (L2 regularization) linear model, respectively.
✓	✓	`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 `X` passed in according to the fitted `model`. If binary classes, returns probabilities for both classes 0 and 1.
✓	✓	`model.coef_`	Estimated coefficients for the linear model, not including the intercept.
✓	✓	`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 $\mathrm{Var}(X)=\mathbb{E}[(X-\mathbb{E}[X])^2]$, and standard deviation $\mathrm{SD}(X)=\sqrt{\mathrm{Var}(X)}$.

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 $\mathrm{Cov}(X,Y)=0$.

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

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 $\mathrm{SD}(\overline{X}_n)=\sigma/\sqrt{n}$.

Model Risk

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 $\hat{Y}(x)$: $\mathbb{E}[(Y-\hat{Y}(x))^2]=\sigma^2+\left(\mathbb{E}[\hat{Y}(x)]-g(x)\right)^2+\mathrm{Var}(\hat{Y}(x))$.

SQL

```
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`, `SUM`, `COUNT`, `AVG`, etc). `AS` renames columns. `DISTINCT` selects only unique rows.

`WHERE a IN cons_list`
`WHERE a IS NOT val`
`WHERE a LIKE 'p'`
	• Select rows for which the value in column `a` is among the values in a `cons_list`.
	• Selects rows for which the value in column `a` is not equal to `val` (of any data type).
	• Matches each entry in the column `a` to the text pattern `p`. The wildcard `%` matches at least zero characters. _ matches exactly one character.

`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.
`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)`.

Decision Trees

Suppose you have a **decision tree** classifier for k classes. For each node, define the probability for class $C \in \{1, \ldots, k\}$ as $p_C = d_C/d$, where d_C is the number of datapoints in class C (of the total d in the node). The entropy of the node (in bits) is defined as $S = -\sum_C p_C \log_2 p_C$, and the weighted entropy of the split is the average entropy of child nodes weighted by the number of datapoints in each.
Decision tree generation algorithm: all of the data starts in the root node. Repeat until every node is either pure or unsplittable.

- Pick the best feature x and best split value β to maximize the change in weighted entropy.
- Split data into two nodes, one where $x < \beta$, and one where $x \geq \beta$

A node that only has samples from one class is called a "pure" node. A node that has overlapping datapoints from different classes and thus cannot be split is called "unsplittable."

A **random forest** is a collection of many decision trees fit to variations of the same training data (e.g. bootstrapped samples, also called bagging). It is an ensemble method.