Final Exam Solutions - DSC 80, Fall 2023

Instructions:

- This exam consists of 11 questions. A total of 160 points are available.
- Questions marked with (M) will be used for your midterm exam redemption.
- Write name in the top right of each page in the space provided.
- Please write neatly in the provided answer boxes. We will not grade work that appears elsewhere.
- Completely fill in bubbles and square boxes.

  ◯ A bubble means that you should only **select one choice**.

  ☐ A square box means you should **select all that apply**.
- You may refer to two 8.5" × 11" sheets of notes of your own creation. No other resources or technology (including calculators) are permitted.
- Do not turn the page until instructed to do so.

Last name	
First name	
Student ID number	
UCSD email	
Name of the person to your left	
Name of the person to your right	
All the work on this exam is my own. **(please sign)**	

This page is intentionally left blank. Feel free to use it as scratch paper.

Name: ______________________________

Question 1 . . . . *19 points*

(M) Fill in Python code below so that the last line of each code snippet evaluates to each desired result, using the `bus` and `stop` DataFrames described on Page 1 of the Reference Sheet. **You may not use `for` or `while` loops in any answer for this question.**

(a) (3 points) Compute the median minutes late for the `101` bus.

```
bus.loc[ bus['line'] == 101, 'late' ].median()
```

(b) (4 points) Compute a copy of `bus` with only the bus lines that made at least one stop containing the string "Myers Ln".

```
def f(x):
    return any(x['stop'].str.contains('Myers'))

bus.groupby( 'line' ). filter (f)
```

(c) (4 points) Compute the number of buses in `bus` whose next stop is "UTC".

```
x = stop.merge( bus ,
    on= ['line', 'stop'] , how= 'inner' )
x[ x['next'] == 'UTC' ].shape[0]
```

(d) (8 points) Compute the number of unique pairs of bus stops that are exactly two stops away from each other. For example, if you only use the first four rows of the `stop` table, then your code should evaluate to the number 2, since you can go from "Gilman Dr & Mandeville Ln" to "La Jolla Village Dr & Lebon Dr" and from "Gilman Dr & Mandeville Ln" to "Villa La Jolla Dr & Holiday Ct" in two stops.

Hint: The `suffixes=(1, 2)` argument to `merge` appends a `1` to column labels in the left table and a `2` to column labels in the right table whenever the merged tables share column labels.

```
m = stop .merge( stop ,
    left_on= 'next' , right_on= 'stop' , how= 'inner' ,
    suffixes=(1, 2))
(m[ ['stop1', 'next2'] ]
    .drop_duplicates().shape[0])
```

Question 2 . . . . *12 points*

Sunan wants to work with the `time` column in `bus`, but the times aren't consistently formatted. He writes the following code:

```
import re

def convert(y1, y2, y3):
    return int(y1), int(y2) if y2 else 0, y3

def parse(x):
    # Fill me in

bus['time'].apply(parse)
```

Sunan wants the last line of his code to output a Series containing tuples with parsed information from the `time` column. Each tuple should have three elements: the hour, minute, and "am"/"pm" for each time. For example, the first two values in the `time` column are `'12pm'` and `'1:15pm'`, so the first two tuples in the Series should be: `(12, 0, 'pm')` and `(1, 15, 'pm')`.

Select **all** the correct implementations of the function `parse`. Assume that each value in the `time` column starts with a one or two digits for the hour, followed by an optional colon and an optional two digits for the minute, followed by either "am" or "pm".

Hint: Calling `.groups()` on a regular expression match object returns the groups of the match as a tuple. For nested groups, the outermost group is returned first. For example:

```
>>> re.match(r'(..(...))', 'hello').groups()
('hello', 'llo')
```

- [x]
```
def parse(x):
    res = x[:-2].split(':')
    return convert(res[0], res[1] if len(res) == 2 else 0, x[-2:])
```

- [ ]
```
def parse(x):
    res = re.match(r'(\d+):(\d+)([apm]{2})', x).groups()
    return convert(res[0], res[1], res[2])
```

- [x]
```
def parse(x):
    res = re.match(r'(\d+)(:(\d+))?(am|pm)', x).groups()
    return convert(res[0], res[2], res[3])
```

- [ ]
```
def parse(x):
    res = re.match(r'(.+(.{3})?)(..)', x).groups()
    return convert(res[0], res[1], res[2])
```

Question 3 .... *24 points*

(M) Dylan wants to answer the following questions using hypothesis tests on the `bus` dataframe. For each test, select the **one** correct procedure to simulate a single sample under the null hypothesis, and select the **one** correct test statistic for the hypothesis test among the choices given. Assume that the `time` column of the `bus` dataframe has already been parsed into timestamps.

(a) Are buses more likely to be late in the morning (before 12pm) or the afternoon (after 12pm)?

Simulation procedure:

- ◯ `np.random.choice([0, 1], bus.shape[0])`
- ◯ `np.random.choice(bus['late'], bus.shape[0], replace=True)`
- ◯ **Randomly permute the `late` column.**

Test statistic:

- ◯ **Difference in means**
- ◯ Absolute difference in means
- ◯ **Difference in proportions**
- ◯ Absolute difference in proportions

(Both choices 1 and 3 were marked correct for this problem.)

(b) Are buses equally likely to be early or late?

Simulation procedure:

- ◯ **`np.random.choice([0, 1], bus.shape[0])`**
- ◯ `np.random.choice(bus['late'], bus.shape[0], replace=True)`
- ◯ Randomly permute the `late` column.

Test statistic:

- ◯ **Number of values below 0.**
- ◯ **`np.mean`**
- ◯ `np.std`
- ◯ TVD
- ◯ K-S statistic

(Both choices 1 and 2 were marked correct for this problem.)

(c) Is the `late` column MAR dependent on the `line` column?

Simulation procedure:

- ◯ `np.random.choice([0, 1], bus.shape[0])`
- ◯ `np.random.choice(bus['late'], bus.shape[0], replace=True)`
- ◯ **Randomly permute the `late` column.**

Test statistic:

- ◯ Absolute difference in means
- ◯ Absolute difference in proportions
- ◯ **TVD**
- ◯ K-S statistic

(d) Is the `late` column MAR dependent on the `time` column?

Simulation procedure:

- ◯ `np.random.choice([0, 1], bus.shape[0])`
- ◯ `np.random.choice(bus['late'], bus.shape[0], replace=True)`
- ◯ **Randomly permute the `late` column.**

Test statistic:

- ◯ Absolute difference in proportions
- ◯ TVD
- ◯ **K-S statistic**

Name: ____________________

Question 4 . . . . *9 points*

(M) Answer the following questions about missingness mechanisms.

(a) (3 points) What is the missingness mechanism for the `next` column in the `stop` dataframe?

◯ NMAR ◯ MAR ◯ MCAR ◯ **Missing by design**

(b) (3 points) Suppose that the missing values in `late` column from the `bus` dataframe are missing because Sam got suspicious of negative values and deleted a few of them. What is the missingness mechanism for the values in the `late` column?

◯ **NMAR** ◯ MAR ◯ MCAR ◯ Missing by design

(c) (3 points) Suppose that the missing values in `late` column from the `bus` dataframe are missing because Tiffany made an update to the GPS system at 8am and the system was down for 15 minutes afterwards. What is the missingness mechanism for the values in the `late` column?

◯ NMAR ◯ **MAR** ◯ MCAR ◯ Missing by design

Question 5 . . . . *8 points*

(M) Giorgia defines the following variables:

```
a = bus['late'].mean()            b = bus['late'].std()
```

She applies the imputation methods below to the `late` column, then recalculates `a` and `b`. For each imputation method, choose whether the new values of `a` and `b` will be lower (-), higher (+), exactly the same (=), or approximately the same (≈) as the original values of `a` and `b`.

(a) (4 points) Mean imputation:

a: ◯ - ◯ + ◯ **=** ◯ ≈ b: ◯ **-** ◯ + ◯ = ◯ ≈

(b) (4 points) Probabilistic imputation:

a: ◯ - ◯ + ◯ = ◯ **≈** b: ◯ - ◯ + ◯ = ◯ **≈**

Question 6 . . . . *9 points*

Consider three classifiers with the following confusion matrices:

Model A

Actual \ Predicted	Yes	No
Yes	40	10
No	10	40

Model B

Actual \ Predicted	Yes	No
Yes	80	0
No	10	10

Model C

Actual \ Predicted	Yes	No
Yes	80	10
No	5	5

i. (3 points) Which model has the highest accuracy?

◯ Model A ◯ **Model B** ◯ Model C

ii. (3 points) Which model has the highest precision?

◯ Model A ◯ Model B ◯ **Model C**

iii. (3 points) Which model has the highest recall?

◯ Model A ◯ **Model B** ◯ Model C

Name: ______________________________

Question 7 . . . . *12 points*

Alan set up a web page for his DSC 80 notes with the following HTML:

```
<html>
  <body>
    <div id="hero">DSC 80 NOTES</div>
    <div class="notes">
      <div class="notes">
        <p>Lecture 1: 5/5 stars!</p>
      </div>
      <div class="lecture notes">
        <p>Lecture 2: 6/5 stars!!</p>
      </div>
    </div>
    <div class="lecture">
      <p>Lecture 3: 10/5 stars!!!!</p>
    </div>
  </body>
</html>
```

Assume that the web page is parsed into a BeautifulSoup called `soup`.

Fill in each of the expressions below to evaluate to the desired string. Pay careful attention to the indexes after each call to `find_all()`!

(a) (4 points) `"Lecture 1: 5/5 stars!"`

`soup.find_all(` **'p'** `)[0].text`

(b) (4 points) `"Lecture 2: 6/5 stars!!"`

`soup.find_all(` **'div'** `)[3].text`

(c) (4 points) `"Lecture 3: 10/5 stars!!!!"`

`soup.find_all(` **class_='lecture'** `)[1].text`

Name: ______________________________

Question 8 . . . . *14 points*

Consider the following corpus:

Document number	Content
1	`'yesterday rainy today sunny'`
2	`'yesterday sunny today sunny'`
3	`'today rainy yesterday today'`
4	`'yesterday yesterday today today'`

(a) (6 points) Using a bag-of-words representation, which two documents have the largest dot product? Show your work, then write your final answer in the blanks below.

Solution: The bag-of-words representation for the documents is:

Document	`yesterday`	`rainy`	`today`	`sunny`
1	1	1	1	1
2	1	0	1	2
3	1	1	2	0
4	2	0	2	0

The dot product between documents 3 and 4 is 6, which is the highest among all pairs of documents.

Documents ______ **3** ______ and ______ **4** ______

(b) (4 points) Using a bag-of-words representation, what is the cosine similarity between documents 2 and 3? Show your work below, then write your final answer in the blank below.

Solution: The dot product between documents 2 and 3 is:

$$1 + 0 + 2 + 0 = 3 \tag{1}$$

The magnitude of document 2 is equal to document 3 and is:

$$\sqrt{1^2 + 0^2 + 1^2 + 2^2} = \sqrt{6} \tag{2}$$

So, the cosine similarity is:

$$\frac{3}{\sqrt{6} \times \sqrt{6}} = \frac{1}{2} \tag{3}$$

The cosine similarity between documents 2 and 3 is: ______ **0.5** ______.

(c) (4 points) Which words have a TF-IDF score of **0** for all four documents? Assume that we use base-2 logarithms. **Select all the words that apply.**

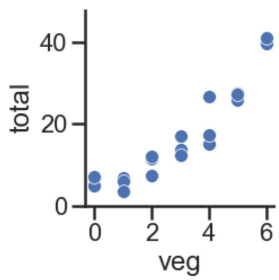
- ■ `yesterday`
- ☐ `rainy`
- ■ `today`
- ☐ `sunny`

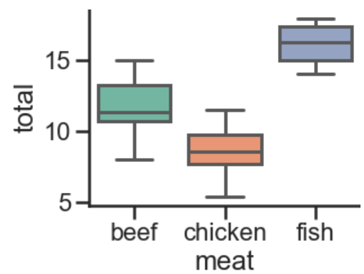
Name: ______________________________

Question 9 . . . . *35 points*

Every week, Lauren goes to her local grocery store and buys a varying amount of vegetables but always buys exactly one pound of meat (either beef, fish, or chicken). We use a linear regression model to predict her total grocery bill. We've collected a dataset containing the pounds of vegetables bought, the type of meat bought, and the total bill. Below we display the first few rows of the dataset and two plots generated using the entire training set.

veg	meat	total
1	beef	13
3	fish	19
2	beef	16
0	chicken	9





(a) Suppose we fit the following linear regression models to predict `total`. Based on the data and visualizations shown above, determine whether the fitted model weights are positive (+), negative (-), or exactly 0. The notation `meat=beef` refers to the one-hot encoded `meat` column with value 1 if the original value in the `meat` column was `beef` and 0 otherwise. Likewise, `meat=chicken` and `meat=fish` are the one-hot encoded `meat` columns for `chicken` and `fish`, respectively.

i. (3 points) $H(x) = w_0$

w_0: ◯ + ◯ - ◯ 0 ◯ Not enough info

ii. (4 points) $H(x) = w_0 + w_1 \cdot \texttt{veg}$

w_0: ◯ + ◯ - ◯ 0 ◯ Not enough info

w_1: ◯ + ◯ - ◯ 0 ◯ Not enough info

iii. (4 points) $H(x) = w_0 + w_1 \cdot (\texttt{meat=chicken})$

w_0: ◯ + ◯ - ◯ 0 ◯ Not enough info

w_1: ◯ + ◯ - ◯ 0 ◯ Not enough info

iv. (4 points) $H(x) = w_0 + w_1 \cdot (\texttt{meat=beef}) + w_2 \cdot (\texttt{meat=chicken})$

w_0: ◯ + ◯ - ◯ 0 ◯ Not enough info

w_1: ◯ + ◯ - ◯ 0 ◯ Not enough info

w_2: ◯ + ◯ - ◯ 0 ◯ Not enough info

v. (4 points) $H(x) = w_0 + w_1 \cdot (\texttt{meat=beef}) + w_2 \cdot (\texttt{meat=chicken}) + w_3 \cdot (\texttt{meat=fish})$

w_0: ◯ + ◯ - ◯ 0 ◯ **Not enough info**

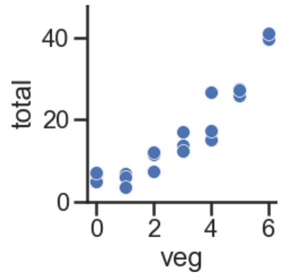
w_1: ◯ + ◯ - ◯ 0 ◯ **Not enough info**

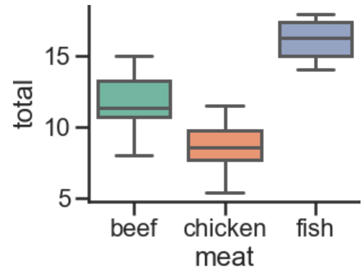
w_2: ◯ + ◯ - ◯ 0 ◯ **Not enough info**

w_3: ◯ + ◯ - ◯ 0 ◯ **Not enough info**

The data and plots from the previous page are reproduced here for convenience:

veg	meat	total
1	beef	13
3	fish	19
2	beef	16
0	chicken	9





Suppose we fit the model: $H(x) = w_0 + w_1 \cdot \texttt{veg} + w_2 \cdot (\texttt{meat=beef}) + w_3 \cdot (\texttt{meat=fish})$

After fitting, we find that $\vec{w} = [-3, 5, 8, 12]$.

(b) (2 points) What is prediction of this model on the **first** point in our dataset?

○ -3 ○ 2 ○ 5 ○ **10** ○ 13 ○ 22 ○ 25

(c) (2 points) What is the loss of this model on the **second** point in our dataset, using squared error loss?

○ 0 ○ 1 ○ 5 ○ 6 ○ 8 ○ 24 ○ **25** ○ 169

(d) (8 points) Determine how each change below affects model bias and variance compared to the model described at the top of this page. **Shade in all the boxes that apply.**

i. Add degree 3 polynomial features.

☐ Increase bias ■ **Decrease bias** ■ **Increase variance** ☐ Decrease variance

ii. Add a feature of numbers chosen at random between 0 and 1.

☐ Increase bias ☐ Decrease bias ■ **Increase variance** ☐ Decrease variance

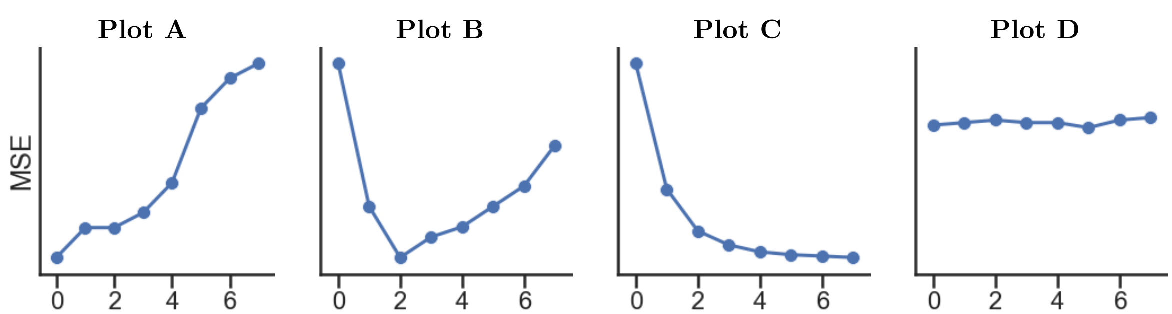
iii. Collect 100 more points for the training set.

☐ Increase bias ☐ Decrease bias ☐ Increase variance ■ **Decrease variance**

iv. Don't use the `veg` feature.

■ **Increase bias** ☐ Decrease bias ☐ Increase variance ■ **Decrease variance**

(e) (4 points) Suppose we predict `total` from `veg` using 8 models with different degree polynomial features (degrees 0 through 7). Which of the following plots display the training and validation errors of these models? Assume that we plot the degree of polynomial features on the x-axis, mean squared error loss on the y-axis, and the plots share y-axis limits.



Training error: ○ A ○ B ○ **C** ○ D

Validation error: ○ A ○ **B** ○ C ○ D

Name: ____________________

Question 10 .... ***18 points***

(a) (9 points) Suppose we fit decision trees of varying depths to predict `y` using `x1` and `x2`. The entire training set is shown in the table below. What is the:

x1	x2	y
A	1	0
A	2	1
B	3	0
B	4	1
A	1	0
A	2	1
B	3	0
B	4	1

The entropy of a node containing all the training points?

○ 0 ○ 0.5 ○ **1** ○ 2

Lowest possible entropy of a node in a fitted tree with depth 1 (two leaf nodes)?

○ **0** ○ 0.5 ○ 1 ○ 2

Lowest possible entropy of a node in a fitted tree with depth 2 (four leaf nodes)?

○ **0** ○ 0.5 ○ 1 ○ 2

(b) Suppose we write the following code:

```
hyperparameters = {
    'n_estimators': [10, 100, 1000], # number of trees per forest
    'max_depth': [None, 100, 10]     # max depth of each tree
}
grids = GridSearchCV(
    RandomForestClassifier(), param_grid=hyperparameters,
    cv=3, # 3-fold cross-validation
)
grids.fit(X_train, y_train)
```

Answer the following questions with a single number. Write your answer in the blank below each question.

i. (3 points) How many random forests are fit in total?

27

ii. (3 points) How many decision trees are fit in total?

9990

iii. (3 points) How many times in total is the first point in `X_train` used to train a decision tree?

6660

Name: ____________________

Question 11 . . . . ***0 points***

Optional: Draw a Picture About UCSD Data Science (or use this page for scratch work)