Online 1. _______/16
Online 2. _______/24
Online 3. _______/15
Online 4. _______/12
Online 5. _______/8
Online 6. _______/25

Total _______/100

The final exam for 6.0002 is on MITx. **Do not write your answers on this paper.** Only fill out this front page with your name, username, and recitation.

You are allowed 2 pages (4 sides total) on which you can write anything. During the test, you are allowed to look at your own study code that you may have annotated with your own comments and changes. The comments and changes you may have made to the code should be related to the study code ONLY.

On your computer, you are only allowed to have the MITx website open and IDLE (or whatever Python IDE you prefer). Close everything except for these. **If the test proctors see that you have anything else open on the computer or if we believe you are trying to cheat, you will be asked to leave immediately and given a zero on the exam.**

**You have 120 minutes to complete the exam.**

**You must check out with an exam proctor after you have finished your exam but before you leave. Hand in this paper exam when you checkout.** Any student who does not do this will receive a 0 on the exam.
1) [16 points] Multiple Choice

1-1) Suppose you insert $100 \times N$ random numbers generated by `random.randint(0, 99)` in a hash table with 10 buckets whose hash function is $h(x) = x \mod 10$. Select the most accurate statement.

   a) If $N = 1$, each bucket in the hash table will have exactly 10 values.
   b) As $N$ goes to infinity, the number of items in the buckets will have a Gaussian distribution across all buckets.
   c) As $N$ goes to infinity, the number of items in the buckets will have a uniform distribution across all buckets.

1-2) A greedy optimization algorithm

   a) is typically efficient in time.
   b) always finds an answer faster than a brute force algorithm.
   c) always returns the same answer as the brute force algorithm.
   d) never returns the optimal solution to the problem.

1-3) Suppose you have a weighted directed graph and want to find a path between nodes A and B with the smallest total weight. Select the most accurate statement.

   a) If some edges have negative weights, depth-first search finds a correct solution.
   b) If all edges have weight 2, depth-first search guarantees that the first path found to be is the shortest path.
   c) If some edges have negative weights, breadth-first search finds a correct solution.
   d) If all edges have weight 2, breadth-first search guarantees that the first path found to be is the shortest path.

1-4) When $R^2$ is used to evaluate a model derived using a linear regression, select the most accurate statement.

   a) An $R^2$ value of 1 says that the model does not explain any of the variance in the data.
   b) An $R^2$ value of 1 says that the model fully explains the variance in the data.
   c) An $R^2$ value is a float between -1 and 1.
   d) An $R^2$ value is the sum of the distances between the data points and the best fit curve.

1-5) Select the most accurate statement.

   a) Training an algorithm on data set A and then testing it on a completely separate data set B is an example of unsupervised learning.
   b) Training an algorithm on data set A and testing it on the same data set A is an example of supervised learning.
   c) Unsupervised learning requires assigning some k data points randomly to an initial set of clusters.
   d) Supervised learning requires labeled data.
1-6) Select the most accurate statement.

a) Hierarchical and k-means clustering are both greedy algorithms.
b) Hierarchical and k-means clustering both use exhaustive enumeration (brute force).
c) Hierarchical clustering is a greedy algorithm and k-means uses exhaustive enumeration (brute force).
d) Hierarchical clustering uses exhaustive enumeration (brute force) and k-means is a greedy algorithm.

1-7) A p-value

a) is the probability of getting the observed sample results (or more extreme) when the null hypothesis is true.
b) is the probability that the null hypothesis is true.
c) that is large implies that the observations are correct.
d) that is large rejects the null hypothesis

1-8) John flipped 20 coins 100 times each. Coin number 3 came up heads 65 times. From this John concluded that coin number 3 is not a fair coin. Which of the following is an appropriate null hypothesis for computing a p-value for this hypothesis?

a) Coin 3 is fair.
b) At least one of the 20 coins is unfair.
c) With 95 confidence, coin 3 is fair.
d) With 95 confidence, at least one of the 20 coins is unfair.

2) [24 points] Answer the following questions in one or two sentences.

2-1) Consider the following function. Is it deterministic? Justify your answer in one sentence.

```python
def some_number():
    random.seed(0)
    return 10 * random.randint(1,2)
```

Yes because random.seed(0) gets called every time the function runs and the seed is used to give the same random number sequence.

2-2) We toss a biased, unfair coin where \( p(\text{heads}) = \frac{3}{4} \) and \( p(\text{tails}) = \frac{1}{4} \). We toss it 5 times and every time it comes up tails. What is the probability that the 6th toss comes up heads?

\[ \frac{3}{4} \]

2-3) When using supervised learning, would you expect the training error to be larger or smaller than the testing error? Explain in one sentence.

Training error is smaller because you are fitting a model to this data.
2-4) Suppose we are clustering N points into k clusters. If \( \text{coherence} = \frac{\text{sum}_{c \in \text{clusters}}(\text{variance}(c))}{k} \), where \( c \) is one of the \( k \) clusters, what happens to the coherence as \( k \) grows but \( N \) stays the same?

Coherence goes to 0.

2-5) I am thinking of a number between 1 and 10 (inclusive). I ask you to pick a number in that range and you get it right. Is the fact that you got it right one out of one times statistically significant at the 95% confidence interval? Support your answer with calculations.

No because \( p = \frac{1}{10} = 0.1 \)

Need value smaller than 0.05 to be statistically significant at 95% CI.

2-6) Half the students in 6.0002 are females. The professors told you that 25% of the students will get an A and that 75% of the students who get an A are female. What is the probability that you get an A in 6.0002 given that you are a female?

\[
0.75 \times 0.25 / 0.5 = 0.375 \quad \text{or} \quad \frac{3}{8}
\]

2-7) You train a model on some data. The plot on the left shows the curve of fit to the training data. The plot on the right shows the curve of fit to the test data. Is this an example of overfitting or underfitting on the training data? Explain in one sentence.

Overfitting because line of fit works with training data but not on test.

---

3) a) [15 points]

You are given the following simple code. It has functions to create a random graph and to find a path between two nodes. The graph is represented by a dictionary; integer keys represent all the nodes in the graph; each key has a list of integers representing the nodes that the key has a directed edge to. Write the missing function according to the specification.

```python
import random
```
# You are given this function - do not modify

def createRandomGraph():
    """ Creates a graph with 7 randomly chosen integer nodes from 0 to 9 and randomly chosen directed edges (between 10 and 20 edges) """
    g = {}
    n = random.sample([0,1,2,3,4,5,6,7,8,9], 7)
    for i in n:
        g[i] = []
    edges = random.randint(10,20)
    count = 0
    while count < edges:
        a = random.choice(n)
        b = random.choice(n)
        if b not in g[a] and a != b:
            g[a].append(b)
            count += 1
    return g

# You are given this function - do not modify

def findPath(g, start, end, path=[]):
    """ Uses DFS to find a path between a start and an end node in g. If no path is found, returns None. If a path is found, returns the list of nodes """
    path = path + [start]
    if start == end:
        return path
    if not g.has_key(start):
        return None
    for node in g[start]:
        if node not in path:
            newpath = findPath(g, node, end, path)
            if newpath: return newpath
    return None

# Code to test your function

g = createRandomGraph()
print g
a = g.keys()[0]
b = g.keys()[1]
p = findPath(g, a, b)
print a, '->', b, p
r = random.choice(g.keys())
m = allReachable(g, r)
print m

All reachable nodes from "r" are:

```markdown
[r, m]
```
4) [12 points] STUDY CODE

Recall the study code you were given. Answer the following questions about the study code in one or two sentences.

4-1) How many features does each tweet data point have?

6

4-2) Assume you cluster on TIME only. Approximately between which hours in the day are there the most clusters? Use the following ranges:

(A): midnight-4am   (D): noon-4pm
(B): 4am-8am   (E): 4pm-8pm
(C): 8am-noon   (F): 8pm-midnight

(A) midnight to 4am

4-3) Suppose your input tweet file consisted of ONLY these 4 tweets:

<table>
<thead>
<tr>
<th>@TWEET</th>
<th>TIME</th>
<th>LAT</th>
<th>LONG</th>
<th>LENGTH</th>
<th>HAPPY</th>
<th>SAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Focus on the positive love.</td>
<td>65383</td>
<td>39.28</td>
<td>-76.61</td>
<td>27</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>I love being alone!</td>
<td>9596</td>
<td>42.78</td>
<td>-71.07</td>
<td>20</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>I just LOVE Fridays #happytweet</td>
<td>59001</td>
<td>33.77</td>
<td>-84.40</td>
<td>31</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>My back hurts! #sadface</td>
<td>4122</td>
<td>41.54</td>
<td>-74.06</td>
<td>23</td>
<td>0</td>
<td>2</td>
</tr>
</tbody>
</table>

According to the study code given, what would be the scaled value of the SAD feature of the tweet "I love being alone!"?

0.5 or 1/2

4-4) In the function `plotRemovedErr` we separate the data set into training and holdout partitions by writing 

```python
(training, holdout) = randomPartition(points, 0.8)
```

Would changing this line to 

```python
(holdout, training) = randomPartition(points, 0.2)
```

change the expected behavior of `plotRemovedErr`? Explain in one sentence.

No because you are also switching the (training, holdout) tuple and checking that random number less than 0.2

5) [8 points] STUDY CODE

For a given tweet, if the code cannot find a (latitude, longitude) pair within any of the US state polygons, it marks the US state as "xx". For example, the whole name of the data point would be:

3:27:58 xx So happy I got great friends ;) [42.595500000000001,-82.874532700000003]

Assume a tweet came from inside a US state. How could we use the study code and clustering to reasonably guess which US state the tweet came from? Write a few sentences describing your method or conceptual process. Do not write code.

Set all feature weights to 0. Set latitude and longitude weights to 1. Run kmeans on data points with at least 50 clusters. Look at cluster containing the xx tweet in question. Look at what the majority of the states in that cluster is. Set xx to be that state.
6) [25 points] STUDY CODE

Recall the study code you were given. Write a function called `persistentTweets(numTrials, points, k, tweetName, cutoff=0.1)`, with the following specification:

```python
def persistentTweets(numTrials, points, k, tweetName, cutoff=0.1):
    
    Runs kmeans 'numTrials' times on 'points'. Looks at the cluster containing 'tweetName'. Considers another tweet from that cluster as part of the final group if that other tweet appears in all 'numTrials' clusters that 'tweetName' is in.
    Prints out the final group of tweets, including the original 'tweetName'
```

- `numTrials`: how many times to run kmeans
- `points`: list of Tweets
- `k`: number of clusters
- `tweetName`: string representing the tweet

Print format of the output tweet lines should be the name of the Tweet: for example: 19:04:17 SC I love happy people :)

Hint: Tweets within a cluster are separated by the newline '\n' character

Example: As an example, assume you have a data set consisting of 9 tweets:

<table>
<thead>
<tr>
<th>@TWEET</th>
<th>TIME</th>
<th>LAT</th>
<th>LONG</th>
<th>LENGTH</th>
<th>HAPPY</th>
<th>SAD</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>3</td>
<td>-1</td>
<td>-2</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>B</td>
<td>1</td>
<td>-2</td>
<td>-1</td>
<td>10</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>-1</td>
<td>-2</td>
<td>22</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>D</td>
<td>6</td>
<td>-4</td>
<td>-1</td>
<td>15</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>E</td>
<td>22</td>
<td>-1</td>
<td>-2</td>
<td>73</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>F</td>
<td>10</td>
<td>-1</td>
<td>-3</td>
<td>120</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>G</td>
<td>12</td>
<td>-3</td>
<td>-2</td>
<td>97</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>H</td>
<td>1</td>
<td>-1</td>
<td>-2</td>
<td>12</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>I</td>
<td>5</td>
<td>-1</td>
<td>-2</td>
<td>112</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>

Now you cluster into k = 4 clusters and are interested in running `persistentTweets` on tweet "D" `numTrials = 3` times. After each kmeans run, the clusters containing tweet "D" are:

1st kmeans run:
1:00 MA B
6:00 PA D
1:00 MA H
5:00 NJ I

2nd kmeans run:
3:00 PA C
6:00 PA D
5:00 NJ I
1:00 MA H

3rd kmeans run:
5:00 NJ I

Based on these cluster outcomes, the function will print tweets that occur in all 3 clusters (in any order):
6:00 PA D
5:00 NJ I

You can test your code on a smaller test file:
```
points = buildTweetPoints('testtweets.txt')
persistentTweets(3, points, 3, "I love happy people :")
```

Your output should have each tweet on a separate line or you can have an extra blank line between each tweet, but all tweets should NOT be on the same line.
def persistentTweets(numTrials, points, k, tweetName, cutoff=0.1):
    
    Runs kmeans 'numTrials' times on 'points'. Looks at the cluster containing 'tweetName'. Considers another tweet from that cluster as part of the final group if that other tweet appears in all 'numTrials' clusters that 'tweetName' is in.
    
    Prints out the final group of tweets, including the original 'tweetName'

    numTrials: how many times to run kmeans
    points: group of Tweets
    k: number of clusters
    tweetName: string representing the tweet

    Print format of the output tweet lines should be as they are stored by the Cluster class:
    time    state    tweet_string
    ex. 19:04:17    SC    I love happy people :)

    Hint: Tweets within a cluster are separated by the newline '\n' character

    persistentTweets = []
    # for each trial
    for trial in range(numTrials):
        # find the clusters
        clusters, maxDist = kmeans(points, k, cutoff, Tweet, toPrint=False)
        # find the cluster with tweetName in it
        for cluster in clusters:
            if cluster.containsTweetName(tweetName):
                # for that cluster:
                clusterPoints = cluster.getPoints()
                # if it is the first trial, all its tweets are persistent
                if trial == 0:
                    persistentTweets = clusterPoints[:]
                    # otherwise, any tweets seen before that aren't in it are not persistent
                else:
                    newPersistentTweets = []
                    for tweet in persistentTweets:
                        if tweet in clusterPoints:
                            newPersistentTweets.append(tweet)
                    persistentTweets = newPersistentTweets[:]
            for tweet in persistentTweets:
                print tweet.getName()