CSE 163 **Name:** _____
Spring 2019
Exam 2
06/03/2019 **Student Number:** _____
Time Limit: 50 Minutes

Do not open the exam before the exam begins and close the booklet when time is called. Starting early or working after time is called will lead to a -10 deduction. You may write your name and Net ID on the front of the exam before the exam starts.

This exam contains 13 pages (including this cover page) and 4 questions. Some questions have sub-parts.

You are allowed to have one sheet of paper (both sides) with you as your cheat sheet. All other materials besides writing utensils should be put away before the exam starts. This includes all electronic devices like phones, calculators, and smart watches.

This exam is not, in general, graded on style and you do not need to include comments or imports for your code. Specific questions may specify restrictions about the style of your code that you must follow to receive full credit.

You may not abbreviate any code, such as "ditto" marks or "..." marks. You may write code to the side and indicate where it should be inserted. These markings must be unambiguous and any ambiguity when grading may result in a deduction if your code is not readable. All code and answers should remain within the provided boxes if possible.

You are allowed to ask for scratch paper after the exam starts to use as additional space when writing answers, but you must indicate on the original page for the problem that part of the answer is on scratch paper. Scratch paper must be stapled to the **END** of your exam after you finish the test. Failure to do so may result in your work on scratch paper not being accepted while grading.

Initials: _____

Initial above to indicate you have read and agreed to these rules. Failure to initial may result in your exam not being accepted for credit.

Question	Points	Score
1	15	
2	15	
3	20	
4	40	
Total:	90	

---

1. (15 points) For this problem, you will be writing a class that is a simplified classifier for `sklearn` named `SimpleClassifier`. The classifier is "simple" because it just learns the most frequent label and always uses that label as the prediction. You may not use any libraries to solve this problem. In the space provided, write a class named `SimpleClassifier` that has the following methods:

   - An initializer that sets up any initial state the class needs to solve the other tasks.
   - Write a method named `fit` that takes two parameters, a list of dictionaries dataset named `X` and a list of string labels `y` with the same length as `X` that indicates the label for each row. This function should learn which label is the most frequent to use for later prediction.

     You should not assume anything about the number of labels, the features being used in `X`, or the length of the lists (other than the length of `X` matches the length of `y`). You may assume that the lists and their contents are not `None`. Your method can break ties arbitrarily. This method should run in $\mathcal{O}(n)$ time where n is the number of rows in the dataset.
   - Write a method named `predict` that takes a list of dictionaries dataset `X` and returns a list of predicted labels for reach row in the dataset. If the `fit` function has not yet been called, the function should return `None`.

   The class should have private fields and your methods should not modify any of the input data.

   Here is an example program that uses the class you will write:

```
X_train = [
    {'feat1': 1, 'feat2': 2},
    {'feat1': 3, 'feat2': 4},
    {'feat1': 5, 'feat2': 6},
    {'feat1': 7, 'feat2': 8}
]
y = ['a', 'b', 'a', 'c']
X_test = [
    {'feat1': 3, 'feat2': 4},
    {'feat1': 9, 'feat2': 10}
]
model = SimpleClassifier()
print(model.predict(X_train))  # None
model.fit(X_train, y)
print(model.predict(X_train))  # ['a', 'a', 'a', 'a']
print(model.predict(X_test))   # ['a', 'a']
```

You should write your solution in the space on the next page.

Solution:

```
class SimpleClassifier:
    def __init__(self):
        self._label = None

    def fit(self, X, y):
        counts = {}
        for label in y:
            if label in counts:
                counts[label] += 1
            else:
                counts[label] = 1

        max_label = None
        max_count = None
        for label in counts:
            if max_count is None or counts[label] > max_count:
                max_count = counts[label]
                max_label = label
        self._label = max_label

    def predict(self, X):
        if self._label is None:
            return None
        else:
            return [self._label for _ in X]
```

2. (15 points total) This section has short answer questions on miscellaneous topics.

   (a) (5 points) Say we write a hash function for a class that has the following definition:

```
def __hash__(self):
    # randint is a function that returns a random integer
    # between the two numbers each time it is called
    return random.randint(0, 10)
```

   First, is this hash function correct functionally? This means that we can use it for elements in a hash table and it would have the correct behavior externally (but it may or may not be a **good** hash function).

   ◯ Yes

   √ **No**

   If you answered "Yes" to the previous question, describe in one or two sentences whether or not this is a good hash function and why. If you answered "No" to the previous question, describe in one or two sentences why it does not work. For either option, be specific by describing what would happen if we used it.

Solution: This hash function does not work because it returns a random number each time it's called. This means if you were to put an item in the hash table, you would get one random hash, and then a different random hash when you looked it up; we would not be able to find the values since the hash values aren't consistent.

   (b) (5 points) In class, we discussed the COMPAS system that uses machine learning to predict the rate that someone exiting jail would commit a crime again. As a reminder, we were looking at the fact that the system disproportionally gave higher likelihoods of repeat crime to people of color.

   Suppose the model used many features like name, age, race, home address, income, and many other features of the person and their crime history to make its prediction. To prevent this differential impact on people of color, one proposal is to remove the race column from the model.

   In the space on the next page, explain in two or three sentences whether this change to the model would remove the racist behavior and explain why or why not.

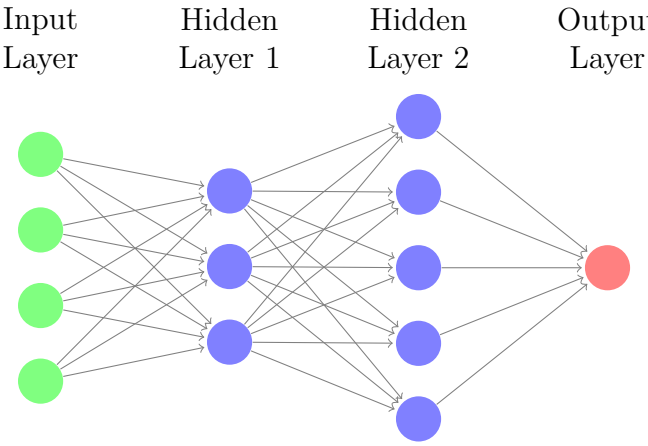
Use the space on the next page.

Solution: Removing the race column will not necessarily fix the problem, because there are other features in the dataset that can be correlated with race and can be used as a proxy. The differential behavior does not necessarily come from having a race column, but instead comes from inherent bias in the data and this can cause these correlations of race with other features.

(c) (5 points) Consider the task of using a neural network that takes images of hand-written digits (0-9) and predicts the digit. The input images are 10 pixels by 10 pixels and there are 10 possible digits. We want to use a neural network with two hidden layers, the first with 15 neurons and the second with 2 neurons.

In the space below, write out the number of weights that will need to be learned in this network. You should show your work for partial credit.

As an example, we show a very small network with two hidden layers below but these are **NOT** the number of weights we want you to compute; we want you to compute the number of weights for the example described in the first paragraph.



Solution:

- Number of weights between the input layer and first hidden layer: $100 * 15 = 1500$
- Number of weights between the first and hidden layers: $15 * 2 = 30$
- Number of weights between the second hidden layer and output layer: $2 * 10 = 20$

This means in total, there will

$$1500 + 30 + 20 = 1550$$

weights

3. (20 points total) For this problem, assume we have the following datasets. The first is a regular `DataFrame` stored in a variable named `df`, while the geo-spatial dataset is stored in a `GeoDataFrame` named `gdf`:

`df`

city	country	population
A	W	200
B	W	100
C	X	300
D	Y	50

`gdf`

name	geometry
W	Polygon1
Z	Polygon2
X	Polygon3

(a) (5 points) What is the resulting table when executing the following join?

```
df.merge(gdf, left_on='country', right_on='name', how='outer')
```

Make sure to include all of the column names. The order of the rows/columns does not matter in the result.

Solution:

city	country	population	name	geometry
A	W	200	W	Polygon1
B	W	100	W	Polygon1
C	X	300	X	Polygon3
D	Y	50	NaN	NaN
NaN	NaN	NaN	Z	Polygon2

(b) (15 points) Write a program that plots the map of the world with the countries colored by the **ratio** of the world population that resides within that country. The population of the country is defined as the sum of the populations of the cities in that country while the world population is the sum of the populations of all the countires. Countries in `gdf` without cities in `df` should be plotted with ratio 0.

The ratios should be between 0 and 1 (inclusive) and you should make sure that when plotting, the range of values shown on the map is between 0 and 1. The plot should have a legend and should be saved in a file named `'world_population.png'`

You do not need to write a function for your solution. You may make the following assumptions:

- The variables `df` and `gdf` as described above have been defined.
- There is a non-zero world population.
- There may be more rows in the variables than shown above, but they will have the columns described. Your code may modify the data to add columns.
- Operations that combine a `DataFrame` and a `GeoDataFrame` returns a `GeoDataFrame`.

Solution:

```
merged = gdf.merge(df, left_on='name', right_on='country',
                   how='left')
merged = merged.fillna(0)

grouped = merged.dissolve(by='name', aggfunc='sum')
total_pop = grouped['population'].sum()
grouped['ratio'] = grouped['population'] / total_pop
grouped.plot(column='ratio', legend=True, vmin=0, vmax=1)

plt.savefig('world_population.png')

# We would aso accept df.merge(gdf, ...) since we said
# you can assume merging operations return GeoDataFrames
```

4. (40 points total) The following problems relate to image processing.

   (a) (9 points total) Assume we have the following three `numpy` arrays defined. We show the name and the shape above each array

`a (4, 3)`

0	0	1
0	1	0
0	1	1
1	0	0

`b (3,)`

1	2	3

`c (2, 1)`

3
4

   In the spaces below, write out 1) the result of the operation and 2) explicitly write down the shape of the result. If the result is an error, indicate so and write a sentence or two explaining why it is an error.

   i. (3 points) `a + b`

Solution: Shape: `(4, 3)`

1	2	4
1	3	3
1	3	4
2	2	3

   ii. (3 points) `a + c`

Solution: This causes an error because the shapes are not compatible when broadcasting. The shapes have the same number of dimensions, but the first dimenions don't match and neither are 1 which is an error.

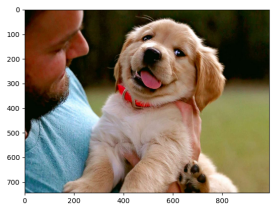
   iii. (3 points) `b + c`

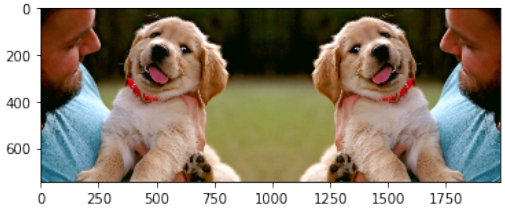
Solution: Shape: `(2, 3)`

4	5	6
5	6	7

(b) (13 points) Write a function called `mirror` that takes a color image (an `(n, m, 3)` `numpy` array) and returns a new color image that is the result of mirroring the image horizontally. Here is an example:

`puppy` `mirror(puppy)`





The resulting image should be twice as wide as the input and the original image should appear on the left side of the result while the image resulting from flipping the input image horizontally should appear on the right. Your method should not modify the input array. For full credit, your solution should have no loops. However, if you can't figure out how to do it with no loops, you can receive partial credit for a solution that has loops.

Solution:

```
def mirror(img):
    height, width, depth = img.shape
    result = np.zeros((height, 2 * width, depth),
                      dtype=np.uint8)
    # you need the dtype param if you actually want to
    # run the code, but we did not grade for having it

    result[:, :width, :] = img
    result[:, width:, :] = img[:, ::-1, :]
    return result
```

(c) The following problems have to do with using convolutions to implement an operator known as the "max pool". The max pool is essentially a convolution using a square window of a given size where we take the max of the numbers at each position. **One big difference is we will not allow the windows to overlap while convoling.**

For example, if we had the following matrix on the left and performed a max pool on it with window size 3, we would get a result shown below on the right.

`matrix`

1	2	3	4	5	6
2	1	2	3	4	5
3	2	1	2	3	4
4	3	2	1	2	3
5	4	3	2	1	2
6	5	4	3	2	1

`max_pool(matrix, 3)`

3	6
6	3

Notice that the result has shape `(2,2)` because we don't let the windows overlap.

i. (3 points) You will first mechanically trace the result of doing a max pool with window size 2 over the matrix below. This means you will make a square window of width/height 2 to slide over the matrix, computing the max at each point, and returning the values as a smaller matrix. Your answer should show

- The matrix resulting from the operation described above
- Explicitly write out the value of `result.shape`

1	2	3	1
2	1	2	2
2	2	1	1
2	2	1	7

Solution: Shape is `(2, 2)`

2	3
2	7

ii. (15 points) Next, you should write a function named `max_pool` that takes a 2d matrix and a window size `size` as parameters and performs this convolution using a square window of the given size over the given matrix, taking the max at each position to store in the result matrix that is returned by the function. Remember, the windows should not overlap in their positions.

You may assume the window size evenly divides the width and height of the image and that the image is square. Your method should not modify the given array. For full credit, your solution should have two loops.

If you are struggling to figure out how to solve this problem, we encourage you to think about the case when the window size `size=2`. If you write a method that works the given size is 2, you can get some partial credit for the problem. Similarly, if you are not sure how to solve this problem with only two loops, you can get partial credit by writing a solution that has more loops.

Solution:

```
def max_pool(img, size):
    height, width = img.shape
    result = np.zeros((height // size, width // size))

    for i in range(result.shape[0]):
        for j in range(result.shape[1]):
            curr = img[size*i:size*i+size, size*j:size*j+size]
            result[i, j] = np.max(curr)
    return result


# Another common solution
def max_pool(img, size):
    height, width = img.shape
    result = np.zeros((height // size, width // size))

    for i in range(0, img.shape[0], size):
        for j in range(0, img.shape[1], size):
            curr = img[i:i+size, j:j+size]
            result[i // size, j // size] = np.max(curr)
    return result
```

This page is left intentionally blank. You may use it as scratch paper.

- **Built-in Python functions**
  - `print(*strings)`
  - `range(end)` / `range(start, end, step?)`
  - `abs(v)`
  - `min(v1, v2)` / `max(v1, v2)`
  - `sum(v1, v2)`
  - `open(fname)`
  - `zip(l1, l2)`
  - Types: `int(v)`,`float(v)`,`str(v)`,`bool(v)`
- **String methods**
  - `upper()`, `lower()`
  - `find(s)`
  - `strip()`
  - `split()`
- **List methods**
  - Construct: `list()` or `[]`
  - `append(val)`
  - `extend(lst)`
  - `insert(idx, val)`
  - `remove(val)`
  - `pop(idx)`
  - `index(val)`
  - `reverse()`
  - `sort(key?)`
- **Set methods**
  - Construct: `set()`
  - `add(val)`
  - `remove(val)`
- **Dictionary methods**
  - Construct: `dict()` or `{}`
  - `keys()`
  - `values()`
  - `items()`
- **Special object methods**
  - `__init__`
  - `__repr__`
  - `__eq__`
  - `__hash__`
- **Pandas methods**
  - `mean()`
  - `min()` / `max()`
  - `idxmin()` / `idxmax()`
  - `count()`
  - `unique()`
  - `groupby(col)`
  - `apply(fun)`
  - `isnull()` / `notnull()`
  - `dropna()` / `fillna(v)`
  - `sort_values(col)` / `sort_index()`
  - `nlargest(n, col)`
  - `merge(df, left_on, right_on, how)`
- **Pandas fields**
  - `index`
  - `loc[row, col]`
- **Geopandas obnject methods**
  - `plot(column?, legend?, ax?, color?, vmin?, vmax?)`
  - `dissolve(by, aggfunc)`
  - Any of the pandas functions above
- **Geopandas module methods**
  - `geopandas.sjoin(left, right, op, how)`
- **matplotlib model classes**
  - `plt.subplots(nrows, ncols)`
  - `plt.show()`
  - `plt.savefig(f_name)`
- **numpy module methods**
  - `np.array(vals?)`
  - `np.arange(end)` / `np.arange(start, end, step?)`
  - `np.ones(shape)` / `np.zeros(shape)`
  - `np.dot(a1, a2)`
  - `np.sum(a)` / `np.min(a)` / `np.max(a)` / `np.mean(a)`
- **numpy array object methods**
  - `reshape(shape)`
  - `sum()` / `min()` / `max()` / `mean()`
  - `copy()`
- **numpy module methods**
  - `shape`

DO NOT WRITE ON THIS PAGE.
IT WILL NOT BE SCANNED WITH YOUR EXAM.