DS 100: Principles and Techniques of Data Science Date: February 15, 2019

Discussion #4 Solutions

Name:

Kernel Density Estimation

1. We wish to compare the results of kernel density estimation using a gaussian kernel and a boxcar kernel. For $\alpha > 0$, which of the following statements are true? Choose all that apply.

   Gaussian Kernel:
   $$K_\alpha(x, z) = \frac{1}{\sqrt{2\pi\alpha^2}} \exp\left(-\frac{(x-z)^2}{2\alpha^2}\right)$$

   Box Car Kernel:
   $$B_\alpha(x, z) = \begin{cases} \frac{1}{\alpha} & \text{if } -\frac{\alpha}{2} \leq x - z \leq \frac{\alpha}{2} \\ 0 & \text{else} \end{cases}$$

   A. Decreasing α for a gaussian kernel decreases the smoothness of the KDE.

   B. The gaussian kernel is always better than the boxcar kernel for KDEs.

   C. Because the gaussian kernel is smooth, we can safely use large α values for kernel density estimation without worrying about the actual distribution of data.

   D. The area under the box car kernel is 1, regardless of the value of α.

   E. None of the above.

   Solution:

   A. True

   B. False; if the α values are not carefully selected for the gaussian kernel, the box car kernel can provide a better kernel density estimate

   C. False; if we set α too high we potentially risk including too many points in our estimate, resulting in a flatter curve.

   D. True

Regular Expressions

2. Which strings contain a match for the following regular expression, `"1+1$"`? The character `"␣"` represents a single space.

   ◯ `What␣is␣1+1` [✓ `Make␣a␣wish␣at␣11:11`] ◯ `111␣Ways␣to␣Succeed`

   Solution: Recall that `1+` matches on *at least one* occurrence of the character `1`, and `$` marks the end of the string.

3. Given the text:

   ```
   "<record>␣Fernando␣Perez␣<fperez@berkeley.edu>␣Faculty␣</record>"
   "<record>␣Edward␣Fang␣<edward.fang@berkeley.edu>␣TA␣</record>"
   ```

   Which of the following matches exactly to the email addresses (including angle brackets)?
   ◯ `<.*@.*>` [✓ `<[^>]*@[^>]*>`] ◯ `<.*@\w+\..*>`

   Solution: Greediness matches too much in the first and third choices.

4. For each pattern specify the starting and ending position of the first match in the string. The index starts at zero and we are using closed intervals (both endpoints are included).

	`abcdefg`	`abcs!`	`ab␣abc`	`abc,␣123`
`abc*`	[0, 2]	[0, 2]	[0, 1]	[0, 2]
`[^\s]+`	[0, 6]	[0, 4]	[0, 1]	[0, 3]
`ab.*c`	[0, 2]	[0, 2]	[0, 5]	[0, 2]
`[a-z1,9]+`	[0, 6]	[0, 3]	[0, 1]	[0, 3]

5. Write a regular expression that matches strings (including the empty string) that only contain lowercase letters and numbers.

Solution:

```
^[a-z0-9]*$
```

6. Write a regular expression that matches strings that contain exactly 5 vowels.

Solution:

```
^([^aeiouAEIOU]*[aeiouAEIOU]){5}[^aeiouAEIOU]*$
```

7. Given that `address` is a string, use `re.sub` to replace all vowels with a lowercase letter "o". For example `"123␣Orange␣Street"` would be changed to `"123␣orongo␣Stroot"`.

Solution:

```
re.sub(r"[aeiuAEIOU]", "o", address)
```

8. Given `dates = "October␣10,␣November␣11,␣December␣12,␣January␣1"`, use `re.findall` to extract all the numbers in the string. The result should look like `["10", "11", "12", "1"]`.

Solution:

```
re.findall(r"[0-9]+", dates)
```

9. Given the following text in a variable `log`:

```
169.237.46.168 - - [26/Jan/2014:10:47:58 -0800]
"GET␣/stat141/Winter04/␣HTTP/1.1" 200 2585
"http://anson.ucdavis.edu/courses/"
```

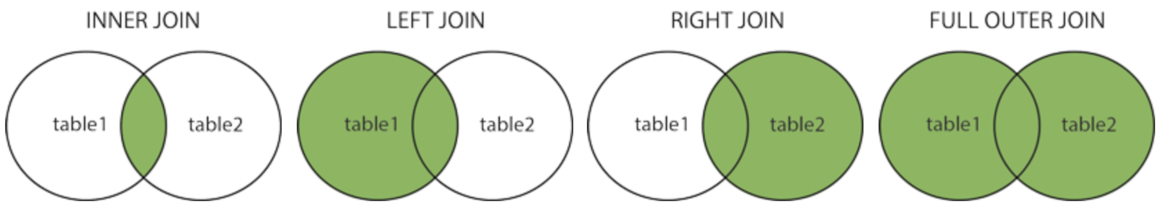
Fill in the regular expression in the variable `pattern` below so that after it executes, day is 26, month is Jan, and year is 2014.

```
pattern = ...
matches = re.findall(pattern, log)
day, month, year = matches[0]
```

Solution:

```
pattern = "\[(.+)\/(.+)\/([^:]+).*\]"
matches = re.findall(pattern, log)
day, month, year = matches[0]
```

SQL



Note: You do not always have to use the JOIN keyword to join sql tables. The following are equivalent:

```
SELECT column1, column2
FROM table1, table2
WHERE table1.id = table2.id;

SELECT column1, column2
FROM table1 JOIN table2
ON table1.id = table2.id;
```

10. Describe which records are returned from each type of join.

Solution:

(INNER) JOIN: Returns records that have matching values in both tables
LEFT (OUTER) JOIN: Return all records from the left table, and the matched records from the right table
RIGHT (OUTER) JOIN: Return all records from the right table, and the matched records from the left table
FULL (OUTER) JOIN: Return all records when there is a match in either left or right table

11. Consider the following real estate schema:

```
Homes(home_id int, city text, bedrooms int, bathrooms int,
area int)
Transactions(home_id int, buyer_id int, seller_id int,
transaction_date date, sale_price int)
Buyers(buyer_id int, name text)
Sellers(seller_id int, name text)
```

Fill in the blanks in the SQL query to find the id and selling price for each home in Berkeley. If the home has not ben sold yet, **the price should be NULL**.

```
SELECT ____H.home_id, T.sale_price____
FROM ____Homes H____
____LEFT OUTER____ JOIN ____Transactions T____
ON ____H.home_id = T.home_id____
WHERE ____H.city = 'Berkeley'____;
```

Solution: An alternate solution was to use Transactions in the FROM clause and perform a RIGHT OUTER JOIN with Homes.

```
SELECT H.home_id, T.sale_price
FROM Transactions T
RIGHT OUTER JOIN Homes H
ON H.home_id=T.home_id
WHERE H.city = 'Berkeley'
```