Lesson 06 Quiz 2

Due Feb 22 at 11:59pm **Points** 14 **Questions** 14 **Available** until Mar 1 at 11:59pm **Time Limit** None
Allowed Attempts 2

Instructions

Categorical Data

Overview | Labs | Assignment | Code Talk | Quiz 1 | Quiz 2

This quiz refers to the **Consolidation, Decoding, and Dummy Variables for Categorical Data** types.

You are allowed 2 attempts; your highest score will be kept. Correct answers will be shown after the 2nd attempt.

Attempt History

	Attempt	Time	Score
KEPT	Attempt 2	23 minutes	11 out of 14
LATEST	Attempt 2	23 minutes	11 out of 14
	Attempt 1	566 minutes	10 out of 14

Score for this attempt: **11** out of 14
Submitted Feb 25 at 12:50pm
This attempt took 23 minutes.

Question 1

0 / 1 pts

Consider the following code:

```
import numpy as np
x = np.array(["WA", "Washington", "Wash", "UT", "Utah", "Utah", "UT", "Utah", "IO"])
```

Simplify/Consolidate array **x** by relabeling categories.

WA = x[x == "Washington"]
WA = x[x == "Wash"]
UT = x[x == "Utah"]

Correct Answer

x[x == "Washington"] = "WA"
x[x == "Wash"] = "WA"
x[x == "Utah"] = "UT"

You Answered

x[x == "WA"] = "Washington"
x[x == "WA"] = "Wash"
x[x == "UT"] = "Utah"

x[x == "Wash"] = "Washington"
x[x == "WA"] = "Wash"
x[x == "UT"] = "Utah"

A relabel follows the pattern:
x[x == oldLabel] = newLabel
where x is an array of states/labels/categories/strings
oldLabel is the label that will be replaced
newLabel is the replacement label

Question 2

1 / 1 pts

Which of the following is best at tracking the degree of category consolidation in a column?

Correct!

- (selected) `len(Devices.loc[:,"Names"].unique())`
- `Devices.loc[:,"Names"]`
- `Devices.isnull().sum()`
- `Devices.head()`
- `len(Devices.loc[:,"Names"])`
- `scatter_matrix(Devices)`

Consolidation could be measured by number of categories.
Unique categories are returned in an array using `unique()`
The number of unique categories is determined by using the `len()` on the array.

Question 3

1 / 1 pts

Consider the following code in which you want to consolidate `DeviceTypes` into 4 categories.
The first consolidated category is **Phone**, the second consolidated category is **KitchenAppliance**, etc.

```
DeviceTypes = [
"Cell Phone", "Dish Washer", "Laptop", "Phone", "Refrigerator", "Server",
"Oven", "Computer", "Drill", "Server", "Saw", "Computer", "Nail Gun",
"Screw Driver", "Drill", "Saw", "Saw", "Laptop", "Oven", "Dish Washer",
"Oven", "Server", "Mobile Phone", "Cell Phone", "Server", "Phone"]
Devices = pd.DataFrame(DeviceTypes, columns=["Names"])
```

What is the **size** of the largest of the four consolidated categories?

- 6
- 4
- 10

Correct!

8

Consolidate into Phone, Computer, Tool, Appliance using pattern:

```
Devices.loc[Devices.loc[:, "Names"] == SomeCategory, "Names"] = ConsolidatedCategory
Devices.loc[:,"Names"].value_counts()
```

Question 4

1 / 1 pts

Consider the following lines of code:

```
DeviceTypes = [
"Cell Phone", "Dish Washer", "Laptop", "Phone", "Refrigerator", "Server",
"Oven", "Computer", "Drill", "Server", "Saw", "Computer", "Nail Gun",
"Screw Driver", "Drill", "Saw", "Saw", "Laptop", "Oven", "Dish Washer",
"Oven", "Server", "Mobile Phone", "Cell Phone", "Server", "Phone"]
Devices = pd.DataFrame(DeviceTypes, columns=["Names"])
```

What is the best way to present the **distribution** in the Names column?

Correct!

- `Devices.loc[:,"Names"].value_counts().plot(kind='bar')`
- `plt.hist(Devices)`
- `Devices.loc[:, "Names"]`
- `plt.hist(Devices.loc[:, "Names"])`

`Devices.loc[:,"Names"].value_counts().plot(kind='bar')` plots the frequency of the unique categories of the column.

Question 5

1 / 1 pts

Consider the following lines of code:

```
import pandas as pd
import numpy as np
from pandas.tools.plotting import scatter_matrix
import matplotlib.pyplot as plt
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
Auto = pd.read_csv(url, delim_whitespace=True, header=None)
Auto.columns = ["mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration", "model_year", "origin",
```

What is the best way to **present the distribution** in the weight column?

- `Auto.loc[:,"weight"].value_counts().plot(kind='bar')`

Correct!

- `plt.hist(Auto.loc[:, "weight"])`

- `Auto.groupby("weight").size().plot(kind='bar')`

- `plt.hist(Auto)`

`weight` is numeric and its distribution is best presented with a histogram.

`plt.hist(Auto)` is not appropriate because Auto is a data frame not a numeric array.

Question 6

1 / 1 pts

Consider the following code:

```
DeviceTypes = [
"Cell Phone", "Dish Washer", "Laptop", "Phone", "Refrigerator", "Server",
"Oven", "Computer", "Drill", "Server", "Saw", "Computer", "Nail Gun",
"Screw Driver", "Drill", "Saw", "Saw", "Laptop", "Oven", "Dish Washer",
"Oven", "Server", "Mobile Phone", "Cell Phone", "Server", "Phone"]
Devices = pd.DataFrame(DeviceTypes, columns=["Names"])
```

Which line of code is the most reasonable for **consolidation**?

- `Devices.loc[Devices.loc[:, "Names"] == "Cell Phone", "Margin"] = "Cell Phone"`

- `Devices.loc[Devices.loc[:, "Names"] == "Drill", "Margin"] = "Phone"`

Correct!

- `Devices.loc[Devices.loc[:, "Names"] == "Laptop", "Margin"] = "Computer"`

- `Devices.loc[Devices.loc[:, "Names"] == "Phone", "Margin"] = "Phone"`

- `Devices.loc[Devices.loc[:, "Names"] == "NewCategory", "Margin"] = "Phone"`

- `Devices.loc[Devices.loc[:, "Names"] == "Phone", "Margin"] = "Laptop"`

The category in the condition on the left-hand side of the assignment must be an existing category, otherwise the assignment is a no-op.

The **condcategory** in the condition on the left hand side of the assignment will be consolidated with a category on the right-hand side of the assignment.

The two categories must be different, otherwise it is a no-op.

Consider the following lines of code for the next 3 questions:

```
url = "https://archive.ics.uci.edu/ml/machine-learning-databases/auto-mpg/auto-mpg.data"
Auto = pd.read_csv(url, delim_whitespace=True, header=None)
Auto.columns = ["mpg", "cylinders", "displacement", "horsepower", "weight",
                "acceleration", "model_year", "origin", "car_name"]
```

Question 7

1 / 1 pts

What is an appropriate way to present the **distribution** in the origin column?

- histogram
- x vs y plot
- Correct! value counts as in "value_counts()"
- unique as in ".unique()"
- scatter plot

origin is not really a number. It is a code for a geography, which is best described as a category. `value_counts()` method lists the unique categories of the column and lists how often each category occurs

Question 8

1 / 1 pts

What is an appropriate way to present the **distribution** in the horsepower column?

- scatter plot
- unique as in ".unique()"
- value counts as in "value_counts()"
- Correct! histogram
- x vs y plot

horsepower is numeric and its distribution is best presented with a histogram.

horsepower needs to be converted to a numeric data type. For example: `Auto.loc[:, "horsepower"] = pd.to_numeric(Auto.loc[:, "horsepower"], errors='coerce')`

The NaNs or missing values need to be removed. For example: `np.isnan(Auto.loc[:, "horsepower"])` or `dropna()`

`value_counts` is not appropriate for numbers, since it does not order numbers numerically.

Question 9

1 / 1 pts

Which of the columns in Auto should be **decoded**?

- displacement
- weight
- mpg
- origin (Correct!)
- acceleration
- model_year
- horsepower
- car_name
- cylinders

Codes are typically low integers.
`Auto.dtypes, Auto.loc[:, "origin"].unique()` shows that origin is numeric and has the values 1, 2, and 3.

Question 10

0 / 1 pts

You need to decode a column of integer-coded categories if:

- The dataset is used in numeric analysis (Correct Answer)
- The code is a measure of the categories' inherent order (You Answered)
- The code uses random numbers
- The code is of type float and not of type integer
- The column is not marked as a coded column

Codes can be problematic because they are hard to understand, but that is not the main problem. The biggest problem with codes is that analyses and machine learning algorithms assume that the numeric code can be used numerically. For example: What is the meaning of the Shape's average in the Mamm data set? The answer is that there is no meaning of this average.

Question 11

1 / 1 pts

The following three arrays are the result of creating dummy variables (one-hot encoding):

```
isTinker = [1, 0, 1, 0, 1, 0, 0, 0]
isTailor = [0, 0, 0, 1, 0, 0, 0, 1]
isSoldier = [0, 1, 0, 0, 0, 0, 0, 0]
```

The original vector was called Profession.

What did Profession look like?

- `Profession = ["Tinker", "Soldier", "Tinker", "Tailor", "Tinker", "Spy", "Soldier", "Tailor"]`
- `Profession = ["Tinker", "Soldier", "Tinker", "Tailor", "Tinker", "Tailor"]`
- none of these options

Correct!

- `Profession = ["Tinker", "Soldier", "Tinker", "Tailor", "Tinker", "Spy", "Spy", "Tailor"]`
- `Profession = ["Tinker", "Soldier", "Tinker", "Taylor", "Tinker", "Soldier", "Soldier", "Taylor"]`

The original array has to be ["Tinker", "Soldier", "Tinker", "Tailor", "Tinker", "Spy", "Spy", "Tailor"]

Try the following code:

```
import numpy as np

Profession = ["Tinker", "Soldier", "Tinker", "Tailor", "Tinker", "Spy", "Spy", "Tailor"]

isTinker=(np.array(Profession)=="Tinker").astype(int)
isTailor=(np.array(Profession)=="Tailor").astype(int)
isSoldier=(np.array(Profession)=="Soldier").astype(int)
```

An array for "Spy" is not necessary. "Spy" is assumed if the other three are FALSE/0

Question 12

1 / 1 pts

Consider the following code:

```
ShirtColor = ["R", "G", "B", "R", "R", "R", "G"]
```

Why would we want to dummy encode this array called ShirtColor?

- Wavelengths cannot be ordered for red, green, and blue

Correct!

- We need a numeric array for data analysis
- Integer or Boolean columns require less memory than columns of string/object
- We need a code for Black
- "R", "G", and "B" cannot be assigned integer codes

The only reason to dummy encode a category column is to use the data in an analysis that requires numeric data. In essence, we convert category data to numeric data.

Question 13

1 / 1 pts

Consider the following code:

```
AB = ["B", "A", "B", "B", "B", "A"]
Test = pd.DataFrame(AB, columns=["AB"])
```

What code creates dummy variable(s) for column AB?

Correct!

- `Test.loc[:, "isA"] = (Test.loc[:, "AB"] == "A").astype(int)`
- `Test.loc[:, "isA"] = ("A" == "B").astype(int)`
- `Test.loc[:, "isA"] = Test.loc[:, "AB"].astype(int)`
- `Test.loc[:, "isA"] = Test.dropna()`

One-hot encoding or creating dummy variables follows the pattern: `isState = x == state`

Where

- x: is an array of states (categories, labels, strings, etc.)
- state: is a state or category like "A", "Red" or "Truck"
- isState: is a binary array of Booleans or preferably 1/0

Question 14

0 / 1 pts

The result of **one-hot encoding** (creating dummy variables) of a category array is a set of one or more arrays.

Each array has the following requirement:

- must be of type integer

must contain only the values 1 and 0

You Answered — must contain only two numbers

must contain only the values True and False

Correct Answer — must be of type Boolean

Dummy variables can be any two numbers. Boolean values count as numbers.

Quiz Score: **11** out of 14