Categorical Variables

1405 Instructor: Ruiqing (Sam) Cao

Categorical vs. Numeric Variables

- **Categorical variables** represent distinct categories or groups
  - Examples: IPO status, gender, country of residence
  - *Arithmetic* and *comparison* operators do not make sense on values of a categorical variable (e.g., brown+blue is undefined, and so is brown>blue)
  - *Summary statistics* like mean, sum, and standard deviation do not make sense on values of a categorical variable

- A **binary variable** are an important special case of categorical variables, which has exactly two categories often represented by `True` (1) and `False` (`0`)

  → also called a dummy, dummy variable, zero-one variable, or indicator variable

Relabel Values of a Categorical Column

	`DataFrame.col.map(dict)`
Arguments	`dict`: a dictionary mapping the original values to the new values
Returns	a Pandas Series with the original values replaced by new values

Relabel Values of a Categorical Column

- The column `month` has 12 unique values `{'01', '02',..., '12'}`
- To map `'01'→'jan'`, `'02'→'feb'`,... and store the result in new variable `month_char`, we can call `Series.map({old:new})`

```
pivoted['month_char'] = pivoted['month'].map(month_dict)
```

month
02
11
01
02
04
05
06

month_char
feb
nov
jan
feb
apr
may
jun

Discretize Numerical Data: Bins

	`pd.cut(Series, bins, labels)`
Arguments	`Series`: the column (variable) to be binned `bins`: the list indicating thresholds of the custom-defined bins `labels`: labels associated with each bin in the output Series
Returns	a Pandas Series with values from `labels` that indicate which bins the values in the original Series are in

Discretize Numerical Data: Bins

- Define `bins=[0,10,300,606]` as cutoffs for discretizing `count0`
- `pd.cut()` maps `count0` values between the 0th & 1st cutoffs → "Low", between the 1st & 2nd cutoffs → "Medium", and between the 2nd & 3rd cutoffs → "High"

```
pd.cut(pivoted.count0, bins=bins, labels=["Low", "Medium", "High"]
```

First 7 rows

count0
1
1
1
2
1
2
4

countBucket
Low
Low
Low
Low
Low
Low
Low

Discretize Numerical Data: Quantiles

	`pd.qcut(Series, q, labels)`
Arguments	`Series`: the column (variable) for which the quartiles are calculated `q`: an integer indicating the quartile `labels`: labels associated with each quartile in the output Series
Returns	a Pandas Series with values from `labels` that indicate which quartile the values in the original Series are in

Discretize Numerical Data: Quantiles

- Quartiles (q=4) splits values of `count0` into four equal groups
- `pd.qcut()` maps `count0` values into the four quartiles, with the label of each quartile given by `[1,2,3,4]`

```
pd.qcut(pivoted.count0, q=4, labels=[1,2,3,4])
```

Last 7 rows

count0
390
398
355
393
430
558
1

quartile
3
3
3
3
4
4
1

Generate Indicators from a Column

	`pd.get_dummies(Series, prefix)`
Arguments	`Series`: The column (variable) containing categorical values used to generate the indicator variables in the output DataFrame `prefix`: The prefix to be added to the names of the generated indicator variables
Returns	A DataFrame containing indicator variables, where each variable is set to `True` if the value in the original Series matches the name of the indicator variable (excluding the prefix) and set to `False` otherwise

Generate Indicators from a Column

```
pd.get_dummies(pivoted['countBucket'], prefix='bucket')
```

countBucket
Low
Low
Low
Low
Low
...
High
High
High
High
Low

	bucket_Low	bucket_Medium	bucket_High
0	True	False	False
1	True	False	False
2	True	False	False
3	True	False	False
4	True	False	False
...	...	...	...
76	False	False	True
77	False	False	True
78	False	False	True
79	False	False	True
80	True	False	False

Exercise: Categorical Variables

Run the provided code and obtain the DataFrame **pivoted**

1. Convert **month** from string format (e.g., '01', '12') to integer format (e.g., 1, 12)
2. Create a histogram to visualize the distribution of the **month** column
3. Add a new column named **quarter** using **pd.cut()**: quarter = 1 for months January, February, March (1, 2, 3); quarter = 2 for months April, May, June (4, 5, 6); quarter = 3 for months July, August, September (7, 8, 9); quarter = 4 for months October, November, December (10, 11, 12).
4. Convert **year** to integer. Add a new column named **period** using **pd.qcut()**: period = 'pre' for years below median; period = 'post' for years above median
5. (Prepare data for merging.) Create a new data table with two columns, **year** and **month**, containing all possible year-month combinations from February 2011 to August 2018. Ensure that the table includes every combination exactly once, with no duplicate rows.