DataSci 400

lesson 5: dealing with categorical data

Seth Mottaghinejad

today's agenda

- what is and is not categorical data
- nominal vs ordinal
- high vs low cardinality
- `object` vs `category`
- dummy variable
- one-hot-encoding
- `fit` and `transform` for `OneHotEncoder`

categorical data

- technically any data that isn't strictly numeric is categorical
  - a **boolean** is categorical and **binary**
  - any grouping column is categorical
- sometimes numeric data should also be treated as categorical
  - example: 2-door car vs 4-door car / zip code
- a column that is mostly unique for each row is **not** categorical
  - a column of **raw text** text
  - an ID column or **primary key**

kinds of categorical data

- with **nominal** categorical columns the order of categories doesn't matter
  - zipcode
  - gender
- with **ordinal** categorical columns there is a natural ordering
  - Likert scale or any ranking
  - education level
  - age groups

`object` vs `category`

there are two column dtypes for categorical data in `pandas`

- `object` is the default type and is **free-form** (no restrictions on what the categories can be)
- `category` is for categorical data with **limited and pre-defined** categories, explicitly provided or as seen in data
  - any changes to data must **conform** to pre-defined schema
  - schema can change using **special methods** like `add_categories`, `remove_categories`, `remove_unused_categories`, etc.

one-hot encoding

- most ML algorithms don't **directly** handle categorical data
  - for ML data must be numeric and tabular
- one-hot encoding creates dummy variables, **one per category** of the categorical column
- **high-cardinality** categorical columns will result in very **wide** data sets, with lots and lots of dummy columns
  - this requires a lot of computational power
  - we could do one-hot encoding only on top k categories instead

one-hot encoding with `sklearn`

`fit` and `transform` is a common pattern in ML (even for data pre-processing steps like one-hot encoding)

```
from sklearn.preprocessing import OneHotEncoder

onehot = OneHotEncoder(sparse = False)

onehot.fit(data)

col_names = onehot.get_feature_names(bank_cat.columns)
bank_onehot =  pd.DataFrame(onehot.transform(bank_cat),
                            columns = col_names)
```

the end