Exploratory data analysis

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

A closer look at features

```
print(df.columns)
```

```
['id', 'click', 'hour', 'C1', ... ]
```

```
print(df.dtypes)
```

```
id                   object
click                 int64
...
```

- `int` : an integer: `1` , `2` , etc.
- `float` : decimals: `3.02` , `4.56` , etc.
- `object` : string: `"hello"` , `"world"` , etc.
- `datetime` : datetime: `2018-01-01` , etc.

```
df.select_dtypes(
  include=['int', 'float'])
```

```
click                    int64
...
```

Missing data

```
df.info()
```

```
Data columns (total 24 columns):
id              50000 non-null object
```

```
df['id'].isnull()
```

```
[False, False, False, False, ... ]
```

```
df.isnull().sum(axis = 0)
```

```
dtype: object
id                    0
...
```

```
df.isnull().sum(axis = 0).sum()
```

```
0
```

Looking at distributions

```
df.groupby(['search_engine_type',
            'click']).size()
```

```
search_engine_type     click
1002               0          940
                   1          240
                 ...
```

```
df.groupby(['search_engine_type',
            'click']).size().unstack()
```

```
click                  0      1
search_engine_type
1002                 940    240
                 ...
```

Breakdown by CTR

```
df.reset_index()
```

```
click   search_engine_type      0      1
                      1002    940    240
```

```
df.rename(columns = {0: 'non_clicks'}, inplace = True)
```

```
click   search_engine_type  non_clicks  clicks
                      1002         940     240
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Feature engineering

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Dealing with dates

```
print(df.hour.head(1))
```

```
14102101
```

```
df['hour'] = pd.to_datetime(
  df['hour'], format = '%y%m%d%H')
df['hour_of_day'] = df['hour'].dt.hour
print(df.hour.head(1))
```

```
2014-10-21 01:00:0
```

```
print(df.groupby('hour_of_day')
      ['click'].sum())
```

```
             click
hour_of_day
1             1092
2             6546
```

Converting categorical variables via hashing

- Categorical features must be converted into a numerical format
- Hash function: maps arbitrary input to an integer output, returning exact same output for a given input
- Lambda function: `lambda x: f(x)`
- Apply hash function via `f(x) = hash(x)` as follows:

```
df['site_id'] = df['site_id'].apply(lambda x: hash(x), axis = 0)
```

```
83a0ad1a -> -9161053084583616050
85f751fd-> 8182420084941774600
```

A closer look at features

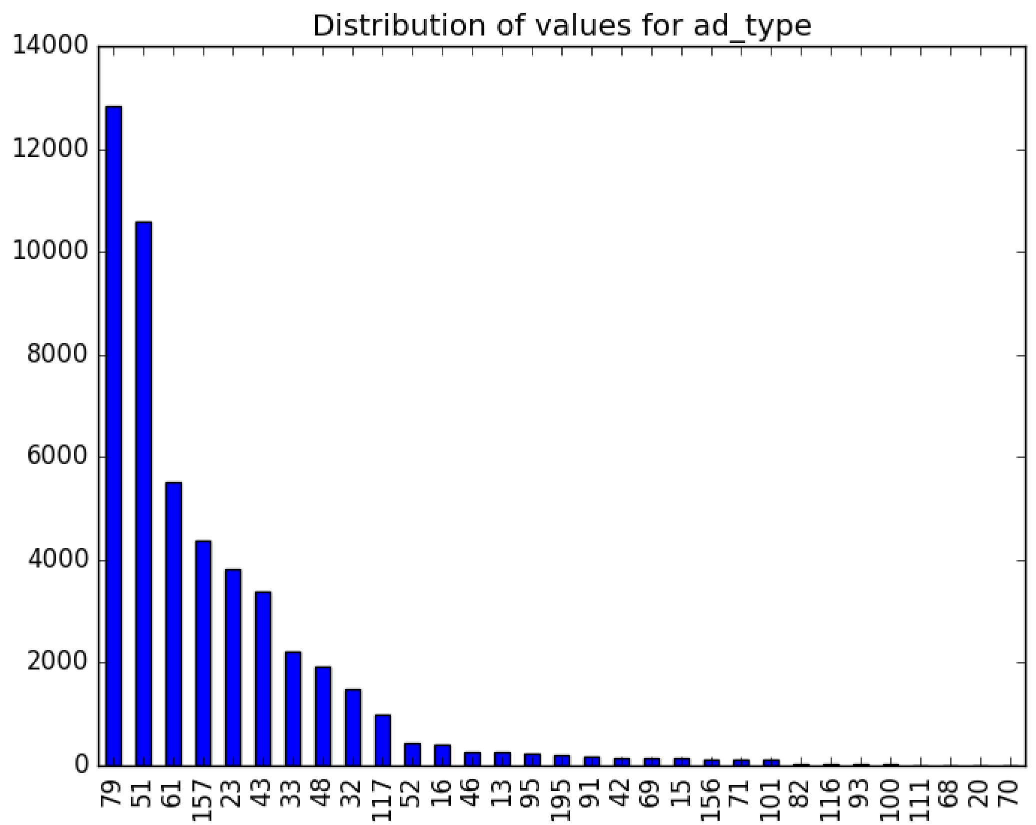
- Examples of `count()` and `nunique()` :

```
df['ad_type'].count()
```

```
50000
```

```
df['ad_type'].nunique()
```

```
31
```



Creating features

- Most of variables are categorical
- Adding more features is better for predictive power
- Example of new feature: impressions by `device_id` (user) and `search_engine_type` :

```
df['device_id_count'] = df.groupby('device_id')['click'].transform("count")
df['search_engine_type_count'] = df.groupby('search_engine_type')['click'].transform("count")
print(df.head(1))
```

```
...   device_id_count  search_engine_type_count
...             40862                     47710
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Standardizing features

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON

Kevin Huo
Instructor

Why standardization is important

- Standardization: ensuring your data fits assumptions that models have
- Certain features may have too high variance, which might unfairly dominate models
- Example: certain count have too large of a range of values due to one spam user
- Does not apply to categorical variables such as `site_id` , `app_id` , `device_id` , etc.

Log normalization

```
df.var()
```

```
click                    1.294270e-01
hour                     1.123316e-01
```

```
df.var().median()
```

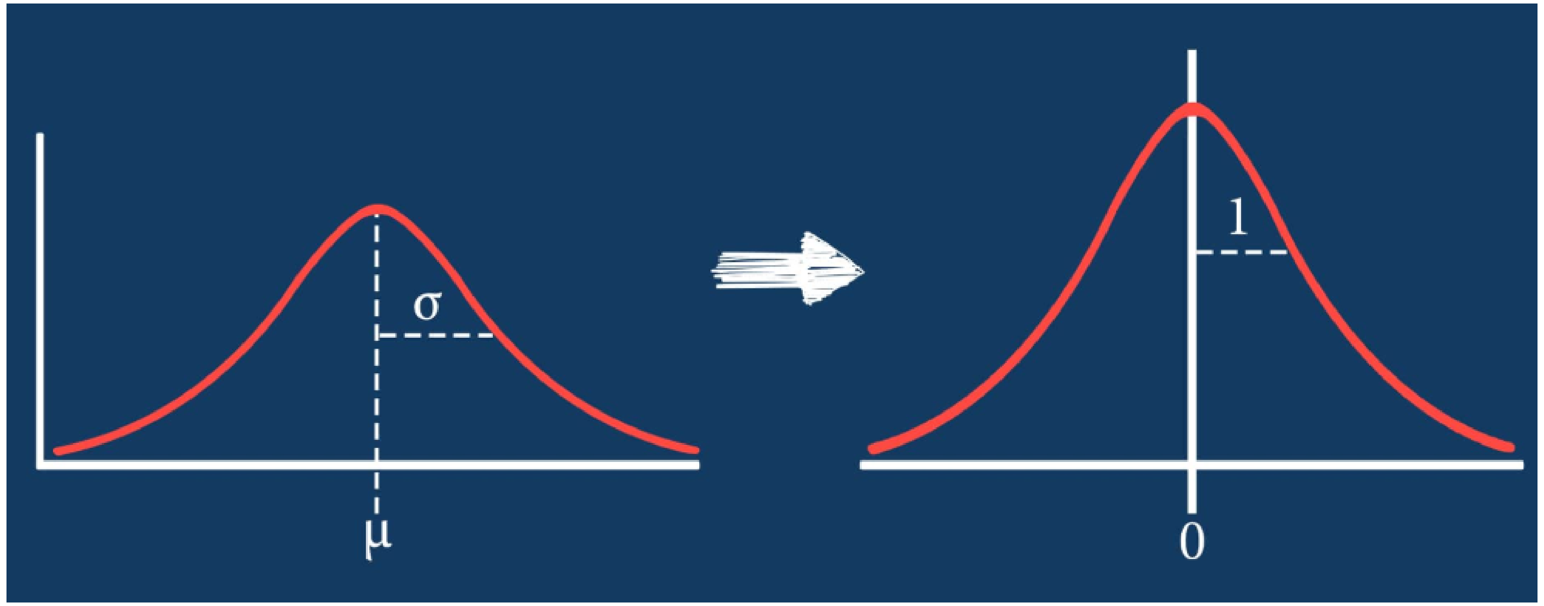
```
0.7108583771671939
```

```
print(df['click'].var())
df['device_id_count'] = df[
  'device_id_count'].apply(
  lambda x: np.log(x))
print(df['click'].var())
```

```
249362570.10134825
15.628476003312514
```

Scaling data

- Standard scaling converts all features to have mean of 0 and standard deviation of 1



- Generally a good practice for machine learning models

How to standard scale data

- Scaling can be done using `StandardScaler()` as follows:

```
scaler = StandardScaler()
X[numeric_cols] = scaler.fit_transform(X[numeric_cols])
```

```
dtype: float64
1     10.5 -> 0.85
2     32.3 -> 1.54
```

Let's practice!

PREDICTING CTR WITH MACHINE LEARNING IN PYTHON