

Snorkel —

Programmatically

Build Training

Data in Python

Khuyen Tran

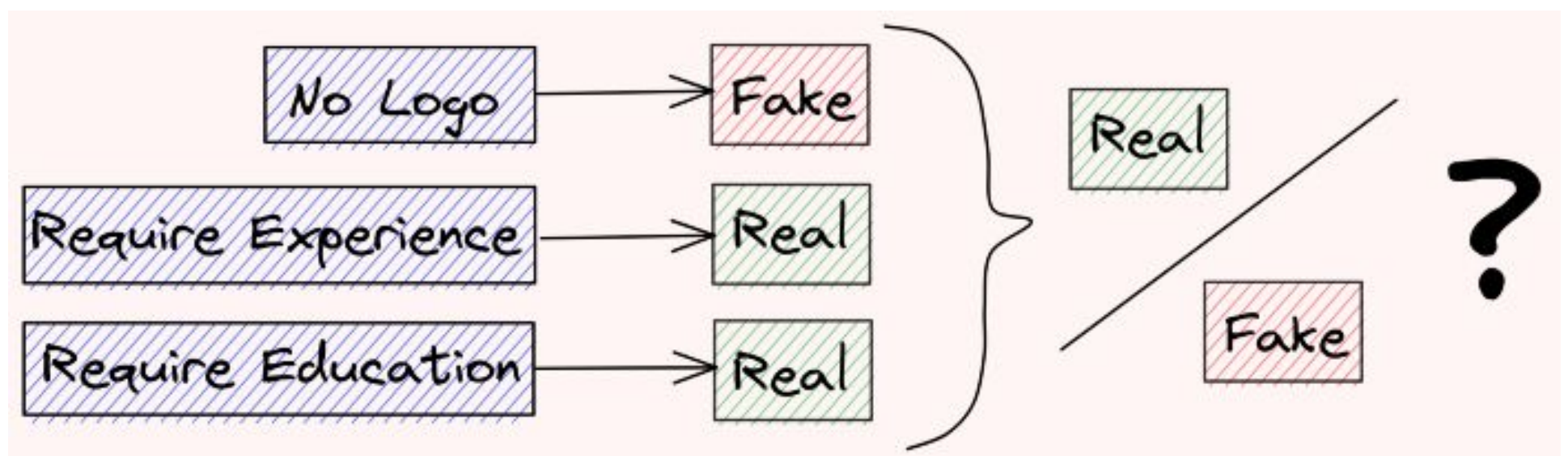


@khuyentran1401

Motivation

Imagine you try to determine whether a job posting is fake or not. You come up with some assumptions about a fake job posting.

How do you test which of these features are the most accurate in predicting fraud?

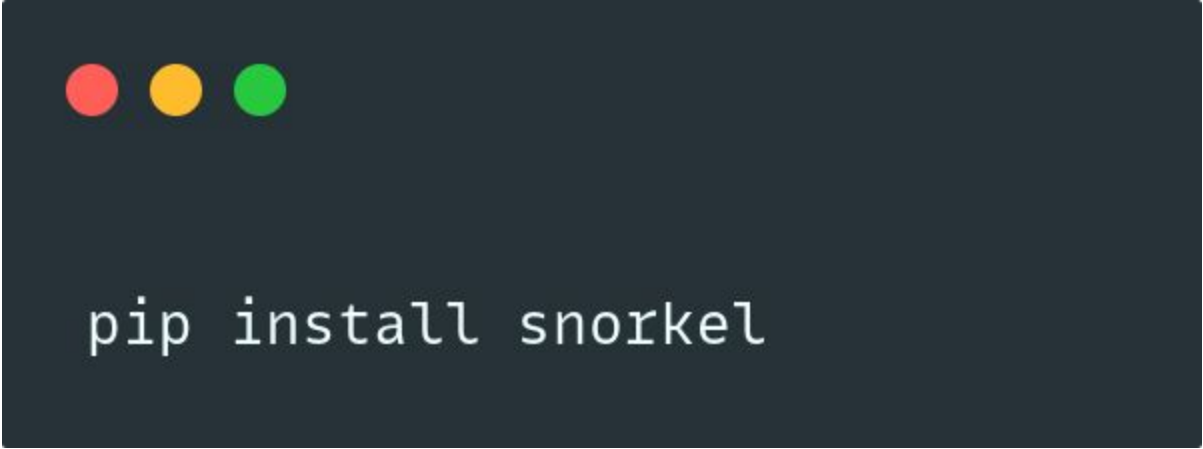


That is when Snorkel comes in handy.

What is Snorkel?

Snorkel is an open-source Python library for programmatically building training datasets without manual labeling. With Snorkel, users can create a training dataset in hours or days rather than manually labeling them over weeks or months.


To install Snorkel, type:



```
pip install snorkel
```

Load Data

We will use Snorkel on the dataset Real / Fake Job Posting Prediction from Kaggle to predict whether a job posting is fake or real. I preprocessed the data and split it into a train and test set.




```
import pandas as pd

train_df = pd.read_pickle(
    "https://github.com/khuyentran1401/"
    "Data-science/blob/master/"
    "feature_engineering/snorkel_example/"
    "train_fake_jobs.pkl?raw=true"
)
```

Give Meaning Names to Values

To learn how Snorkel works, start with giving a meaningful name to each value:



```
from snorkel.labeling import (labeling_function,  
                               PandasLFApplier,  
                               LFAAnalysis)  
  
FAKE = 1  
REAL = 0  
ABSTAIN = -1
```


Create Labeling Functions

We assume that:

- Fake companies don't have company profiles or logos
- Fake companies are found in a lot of fake job postings
- Real job postings often requires a certain level of experience and education

Let's test those assumptions using Snorkel's `labeling_function` decorator. The `labeling_function` decorator allows us to quickly label instances in a dataset using functions.

Create Labeling Functions



```
@labeling_function()
def no_company_profile(x: pd.Series):
    return FAKE if x.company_profile == "" else ABSTAIN

@labeling_function()
def no_company_logo(x: pd.Series):
    return FAKE if x.has_company_logo == 0 else ABSTAIN

@labeling_function()
def required_experience(x: pd.Series):
    return REAL if x.required_experience else ABSTAIN

@labeling_function()
def required_education(x: pd.Series):
    return REAL if x.required_education else ABSTAIN
```


Apply Labeling Functions to the Data

Next, we will use each of these labeling functions to label our training dataset:

```
lfs = [  
    no_company_profile,  
    no_company_logo,  
    required_experience,  
    required_education,  
]  
  
applier = PandasLFApplier(lfs=lfs)  
L_train = applier.apply(df=train_df)
```


Evaluate Labeling Functions

Now that we have created the labels using each labeling function, we can use LFAalysis to determine the accuracy of these labels.

```

LFAalysis(L=L_train, lfs=lfs).lf_summary(
    Y=train_df.fraudulent.values
)

```

	j	Polarity	Coverage	Overlaps	Conflicts	Correct	Incorrect	Emp. Acc.
no_company_profile	0	[1]	0.186204	0.186204	0.186204	459	2038	0.183821
no_company_logo	1	[1]	0.205742	0.205742	0.205742	459	2300	0.166365
required_experience	2	[0]	1.000000	1.000000	0.244295	12741	669	0.950112
required_education	3	[0]	1.000000	1.000000	0.244295	12741	669	0.950112

Explanation of the Statistics

Details of the statistics in the table:


- **Polarity:** The set of unique labels this LF outputs (excluding abstains)
- **Coverage:** The fraction of the dataset that is labeled
- **Overlaps:** The fraction of the dataset where this LF and at least one other LF agree
- **Conflicts:** The fraction of the dataset where this LF and at least one other LF disagree
- **Correct:** The number of data points this LF labels correctly
- **Incorrect:** The number of data points this LF labels incorrectly
- **Empirical Accuracy:** The empirical accuracy of this LF

Learn More

[Link to Snorkel.](#)

[My full article about Snorkel.](#)

Khuyen Tran

 @khuyentran1401