What is Feature Extraction

Edris Safari

Bellevue University, Nebraska U.S.A.

Abstract

Features in a typical dataset embody a massive amount of information. The information can be used to answer many questions and discover new things. Feature extraction is an exercise in the data science pipeline that ensures the integrity of the dataset for model creation which so heavily relies on the correct set of features. In this paper, we will introduce the idea of feature extraction, and show the techniques used to extract features from a dataset.

Keywords:

Features, Dataset, Data Science

What is Feature Extraction

Feature Extraction

Datasets typically come to the data scientists in raw format. They contain what are called observations for a set of features. For example, the observations could the record of customers at a bank and the features would be the customer's ID, first name, last name, date of birth, account type, balance, etc. The number of observations is the number of customers. In a more complex example, we could be faced with data from a textbook. In this case, the number features vary from each letter in the book, to unique words to sentences. In both the simple case and in the more complex case, we need to evaluate the feature set and minimize it to a point when specific models can be created to answer a specific question.

As a part of feature engineering, Feature extraction is a technique in reducing the number of features or dimension of the dataset. The book example above has a dimension the size or number of letters/words/sentences in the book. Depending on the use case, the extracted features will have the following properties:

- Extracted features are a subset of the original features
- Extracted features maintain the underlying information in the dataset.

These properties must ensure that the data generates high-quality models and predictions they generate. It should be noted that extracted features are not necessarily the same features in the dataset, but rather could be derived feature. They can also be derived from 0 or more existing features. This is all part of feature extraction exercise.

One of the techniques used in feature extraction is called the Principle Component Analysis. PCA is a dimensionality reduction algorithm that can be useful as a tool for visualization, noise filtering, feature extraction and essentially feature engineering, and much

more. An example of feature extraction and dimensionality reduction is given by Albon. As shown, the number of features we reduced by 10.

```
# Load libraries
from sklearn.preprocessing import StandardScaler

from sklearn.decomposition import PCA
from sklearn import datasets

# Load the data
digits = datasets.load_digits()
#print(digits)
# Standardize the feature matrix
features = StandardScaler().fit_transform(digits.data)

# Create a PCA that will retain 99% of variance
pca = PCA(n_components=0.99, whiten=True)

# Conduct PCA
features_pca = pca.fit_transform(features)

# Show results
print("Original number of features:", features.shape[1])
print("Reduced number of features:", features_pca.shape[1])
```

```
Original number of features: 64
Reduced number of features: 54
```

Another technique is using CountVectorizer in sklean package. This library creates a human unreadable feature set from text. The numerically valued features represent the number of times unique words are repeated/mentioned in the text. In the example below, we have 7 comments made on different days of the week. Note how CountVectorizer converted the 7 lines of text into a matrix of 7 rows (one row for each comment) and 22 columns. The algorithm found 22 unique words in the document and simply populated the matrix row/column position with the number of times the word(feature) was mentioned in the comment. For example, the first row of the matrix shows that the words ‘are’ and ‘you’ are each mentioned once in the 1st comment. In the 4th comment, the word ‘day is mentioned 2 times, and the words ‘bad’,’good’,’is’,’neither’ are mentioned once each. The new dataset is now the ‘df’ dataframe. From here further education can be made by performing classification algorithm based on what we want to discern from the data (i.e. sentimental analysis).

```
Addr1='DailyComments.csv'
dataframe = pd.read_csv(Addr1)
print(dataframe)
corpus = dataframe['comments']
```

```
  Day of Week                                  comments
0      Monday                       Hello, how are you?
1     Tuesday                     Today is a good day!
2   Wednesday  It's my birthday so it's a really special day!
3    Thursday       Today is neither a good day or a bad day!
4      Friday                      I'm having a bad day.
5    Saturday       There' s nothing special happening today.
6      Sunday                 Today is a SUPER good day!
```

25]:
```
vectorizer = CountVectorizer()
X = vectorizer.fit_transform(corpus)

print("vectorized words")
print(vectorizer.get_feature_names())
print("Identify Feature Words - Matrix View")
print( X.toarray())
print("Matrix shape")
print(X.toarray().shape)
```

```
vectorized words
['are', 'bad', 'birthday', 'day', 'good', 'happening', 'having', 'hello', 'how', 'is', 'it', 'my', 'neither', 'nothing', 'or',
'really', 'so', 'special', 'super', 'there', 'today', 'you']
Identify Feature Words - Matrix View
[[1 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1]
 [0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 0 0 1 0]
 [0 0 1 1 0 0 0 0 0 0 2 1 0 0 0 1 1 1 0 0 0 0]
 [0 1 0 2 1 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0 1 0]
 [0 1 0 1 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
 [0 0 0 0 0 1 0 0 0 0 0 0 0 1 0 0 0 1 0 1 1 0]
 [0 0 0 1 1 0 0 0 0 1 0 0 0 0 0 0 0 1 0 1 0]]
Matrix shape
(7, 22)
```

24]:
```
df = pd.DataFrame(X.toarray(),columns=vectorizer.get_feature_names())
df
```

24]:

	are	bad	birthday	day	good	happening	having	hello	how	is	...	neither	nothing	or	really	so	special	super	there	today	you
0	1	0	0	0	0	0	0	1	1	0	...	0	0	0	0	0	0	0	0	0	1
1	0	0	0	1	1	0	0	0	0	1	...	0	0	0	0	0	0	0	0	1	0
2	0	0	1	1	0	0	0	0	0	0	...	0	0	0	1	1	1	0	0	0	0
3	0	1	0	2	1	0	0	0	0	1	...	1	0	1	0	0	0	0	0	1	0
4	0	1	0	1	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	1	0	0	0	0	...	0	1	0	0	0	1	0	1	1	0
6	0	0	0	1	1	0	0	0	0	1	...	0	0	0	0	0	0	1	0	1	0

Conclusion

Feature extraction is not simply getting a subset of the dataset. It is much more than that. It is deriving and deciphering the dataset to make a dataset which can yield a high-quality modeling and prediction. This science is used heavily in natural language processing as well as visual science where the dataset has a very large number of features (i.e. color for each pixel in the image). There are algorithms and techniques to reduce this dataset to a dataset that can accurately predict (or recognize) an image based on the derived feature set.

References

1. A Quick Guide to Feature Engineering - https://www.kdnuggets.com/2019/02/quick-guide-feature-engineering.html
2. The Hitchhiker's Guide to Feature Extraction - https://www.kdnuggets.com/2019/06/hitchhikers-guide-feature-extraction.html
3. Albon, Chris. Machine Learning with Python Cookbook: Practical Solutions from Preprocessing to Deep Learning (p. 184). O'Reilly Media. Kindle Edition.