PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Feature selection

Sarah Guido
Senior Data Scientist

What is feature selection?

- Selecting features to be used for modeling
- Doesn't create new features
- Improve model's performance

When to select features

city	state	lat	long
hico	tx	31.982778	-98.033333
mackinaw city	mi	45.783889	-84.727778
winchester	ky	37.990000	-84.179722

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Removing redundant features

Sarah Guido
Senior Data Scientist

Redundant features

- Remove noisy features
- Remove correlated features
- Remove duplicated features

Scenarios for manual removal

city	state	lat	long
hico	tx	31.982778	-98.033333
mackinaw city	mi	45.783889	-84.727778
winchester	ky	37.990000	-84.179722

Correlated features

- Statistically correlated: features move together directionally
- Linear models assume feature independence
- Pearson correlation coefficient

Correlated features

```
In [1]: print(df)

        A     B     C
0    3.06  3.92  1.04
1    2.76  3.40  1.05
2    3.24  3.17  1.03
3    3.49  3.45  0.86
...

In [2]: print(df.corr())

          A         B         C
A  1.000000  0.787194  0.543479
B  0.787194  1.000000  0.565468
C  0.543479  0.565468  1.000000
```

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Selecting features using text vectors

Sarah Guido
Senior Data Scientist

Looking at word weights

```
In [1]: print(tfidf_vec.vocabulary_)

{'200': 0,
 '204th': 1,
 '33rd': 2,
 'ahead': 3,
 'alley': 4,
 ...

In [4]: vocab = {v:k for k,v in
     tfidf_vec.vocabulary_.items()}

In [5]: print(vocab)

{0: '200',
 1: '204th',
 2: '33rd',
 3: 'ahead',
 4: 'alley',
 ...
```

```
In [2]: print(text_tfidf[3].data)

[0.19392702 0.20261085 0.24915279
 0.31957651 0.18599931 ...]

In [3]: print(text_tfidf[3].indices)

[ 31 102  20  70   5 ...]

In [6]: zipped_row =
dict(zip(text_tfidf[3].indices,
text_tfidf[3].data))

In [7]: print(zipped_row)

{5: 0.1597882543332701,
 7: 0.26576432098763175,
 8: 0.18599931331925676,
 9: 0.26576432098763175,
 10: 0.13077355258450366,
```

Looking at word weights

```
In [8]: def return_weights(vocab, vector, vector_index):

        zipped = dict(zip(vector[vector_index].indices,
                          vector[vector_index].data))

        return {vocab[i]:zipped[i] for i in vector[vector_index].indices}

In [9]: print(return_weights(vocab, text_tfidf, 3))

{'and': 0.15978825433332701,
 'are': 0.26576432098763175,
 'at': 0.18599931331925676,
 'audubon': 0.26576432098763175,
 'avenue': 0.13077355258450366,
 ...
```

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Dimensionality reduction

Sarah Guido
Senior Data Scientist

Dimensionality reduction and PCA

- Unsupervised learning method
- Combines/decomposes a feature space
- Feature extraction - here we'll use to reduce our feature space

- Principal component analysis
- Linear transformation to uncorrelated space
- Captures as much variance as possible in each component

PCA in scikit-learn

```
In [1]: from sklearn.decomposition import PCA

In [2]: pca = PCA()

In [3]: df_pca = pca.fit_transform(df)

In [4]: print(df_pca)

[88.4583, 18.7764, -2.2379, ..., 0.0954, 0.0361, -0.0034],
[93.4564, 18.6709, -1.7887, ..., -0.0509, 0.1331, 0.0119],
[-186.9433, -0.2133, -5.6307, ..., 0.0332, 0.0271, 0.0055]

In [5]: print(pca.explained_variance_ratio_)

[0.9981, 0.0017, 0.0001, 0.0001, ...]
```

PCA caveats

- Difficult to interpret components
- End of preprocessing journey

PREPROCESSING FOR MACHINE LEARNING IN PYTHON

Let's practice!