Back To Blog Posts

Feb 17, 2015

A List of Helpful Commands for Doing Machine Learning in Python

These commands assume you are using the standard `scikit-learn`, `pandas`, `statsmodels`, and `matplotlib`. If you do not use this ML pipeline, move on, stop reading now.

Cross-Validation

Cross-validation is a method used for splitting the training setup into `test` and `train` sets, and then training the algorithm on the `training` set and evaluating the algorithm on the `test` set. You do not need to divide these up manually, it can be done very easily with `scikit-learn`

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,
        y,
        test_size=0.4,
        random_state=0)
```

It’s important to understand that `train_test_split` does **NOT** preserve you percentage of samples for each class. If you want to perserve this (and if you have unbalanced classes, you most certainly do), use `StratifiedKFold`:

```
X_train, X_test, y_train, y_test = cross_validation.train_test_split(X,
        y,
        test_size=0.4,
        random_state=0)
```

Scaling/Normalizing Features

If different columns values have different magnitudes, you are going to need to normalize your features before you do any training. To do that, you can use `scikit-learn`:

```
 scaler = preprocessing.StandardScaler()
X = scaler.fit_transform(X)
```

From the **documentation**, you can read that `StandardScaler` subtracts the mean and scales to unit variance.

Converting A Pandas Data Frame To Numpy Matrix For Scikit-Learn

Scikit-learn does not currently accept Panda's Dataframes, but that is OK, because you can convert a dataframe into a numpy matrix easily enough with the following command:

```
X = df.as_matrix().astype(np.float)
```

Note this will only work if all of your data is `boolean` + `numerical` (no text)

Dropping All Rows That Contain NAN in a Given Column

If a certain feature is required, you will need to drop any rows in your dataframe that contain `NaN` values for that feature. To do this, you can execute something simple like the following

```
df = df[np.isfinite(df['FeatureColumn'])]
```

Dropping Useless Columns

Sometimes you have columns in your dataframe that you know are not useful for training a model, so drop them, in place:

```
df.drop(['UselessColumn1', 'UselessColumn2'], axis=1, inplace=True)
```

If you look at the columns in your dataframe, `UselessColumn1` and `UselessColumn2` should be gone.

Discretizing Columns In A Dataframe

Supervised learning algorithms usually like looking at numerical features, so if you need to convert a column that contains finite text classes (states, countries, etc.) to numbers, use scikit-learn:

Before:

```
In [72]:

df.State
Out[72]:
0    KS
1    OH
2    NJ
3    OH
4    OK
```

```
5    AL
6    MA
7    MO
8    LA
9    WV
10    IN
11    RI
12    IA
13    MT
14    IA
...
```

And After:

```
 from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
df['State'] = label_encoder.fit_transform(df['State'])

In [74]:

df.State
Out[74]:
0    16
1    35
2    31
3    35
4    36
5     1
6    19
7    24
8    18
```

```
9    49
10    15
11    39
12    12
13    26
14    12
...
3318    36
3319    50
3320    10
3321    46
3322    20
3323    15
3324    49
```

Boolean ANDs or ORs with two columns

Sometimes you want to filter a given set of rows in a dataframe by ANDing or ORing together two columns and take the result. This can be done like this

```
 valid_columns = (df.Column1) | (df.Column2)
df_filtered = df[valid_columns,]
```

The parentheses are very important, otherwise pandas will complain.

plotting multiple columns in pandas via subplots

```
df1[["column1","column1","column1"]].plot(kind='hist',subplots=True, figsize=(12, 6),bins=100);
```

Math, CS, Statsitics, and the occasional book review

Joe is a software engineer living in lower manhattan that specializes in machine learning, statistics, python, and computer vision.