DSC 201: Data Analysis & Visualization

Machine Learning

Dr. David Koop
Python, Pandas, and Time Zones

• Time series in pandas are **time zone native**
• The pytz module keeps track of all of the time zone parameters
  - even Daylight Savings Time
• Localize a timestamp using `tz_localize`
  - `ts = pd.Timestamp("1 Dec 2016 12:30 PM")`
  - `ts = ts.tz_localize("US/Eastern")`
• Convert a timestamp using `tz_convert`
  - `ts.tz_convert("Europe/Budapest")`
• Operations involving timestamps from different time zones become UTC
Downsampling

- Need to define **bin edges** which are used to group the time series into **intervals** that can be aggregated

- Remember:
  - Which side of the interval is closed
  - How to label the aggregated bin (start or end of interval)

```python
closed='left'  9:00  9:01  9:02  9:03  9:04  9:05

closed='right'  9:00  9:01  9:02  9:03  9:04  9:05
```

```
label='left'
label='right'
```
## Upsampling

- **No aggregation necessary**

### Code Examples

```python
In [222]: frame
```

<table>
<thead>
<tr>
<th></th>
<th>Colorado</th>
<th>Texas</th>
<th>New York</th>
<th>Ohio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000-01-05</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-12</td>
<td>-0.046662</td>
<td>0.927238</td>
<td>0.482284</td>
<td>-0.867130</td>
</tr>
</tbody>
</table>

```python
In [223]: df_daily = frame.resample('D').asfreq()
```

```python
In [224]: df_daily
```

<table>
<thead>
<tr>
<th></th>
<th>Colorado</th>
<th>Texas</th>
<th>New York</th>
<th>Ohio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000-01-05</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-06</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-07</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-08</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-09</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-10</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-11</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
<td>NaN</td>
</tr>
<tr>
<td>2000-01-12</td>
<td>-0.046662</td>
<td>0.927238</td>
<td>0.482284</td>
<td>-0.867130</td>
</tr>
</tbody>
</table>

```python
In [225]: frame.resample('D').ffill()
```

```python
Out[225]:
```

<table>
<thead>
<tr>
<th></th>
<th>Colorado</th>
<th>Texas</th>
<th>New York</th>
<th>Ohio</th>
</tr>
</thead>
<tbody>
<tr>
<td>2000-01-05</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-06</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-07</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-08</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-09</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-10</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-11</td>
<td>-0.896431</td>
<td>0.677263</td>
<td>0.036503</td>
<td>0.087102</td>
</tr>
<tr>
<td>2000-01-12</td>
<td>-0.046662</td>
<td>0.927238</td>
<td>0.482284</td>
<td>-0.867130</td>
</tr>
</tbody>
</table>
Interpolation

• Fill in the missing values with computed best estimates using various types of algorithms
• Apply after resample
Resampled Sales Data (ffill)
Resampled with Linear Interpolation (Default)
Resampled with Cubic Interpolation
Rolling Window Calculations

• Example:
  - df.rolling('180D').mean()
  - df.rolling('90D').sum()
Rolling Window Calculations

• Example:
  - `df.rolling('180D').mean()`
  - `df.rolling('90D').sum()`
Rolling Window Calculations

• Example:
  - `df.rolling('180D').mean()`
  - `df.rolling('90D').sum()`

```
12 8 7 4 9 13 4 11 3 8
```

7.8
Rolling Window Calculations

• Example:
  - `df.rolling('180D').mean()`  
  - `df.rolling('90D').sum()`

```
12 8 7 4 9 13 4 11 3 8

7.8 7.0
```
Rolling Window Calculations

• Example:
  - `df.rolling('180D').mean()`
  - `df.rolling('90D').sum()`

```python
[12  8  7  4  9 13  4 11  3  8]

7.8 7.0
```
Rolling Window Calculations

- Example:
  - `df.rolling('180D').mean()`
  - `df.rolling('90D').sum()`

```
12  8  7  4  9  13  4  11  3  8
```

```
7.8 7.0 8.3
```
180-Day Rolling Window (Mean)
Assignment 5

- Compare Hurricane Joaquin and Hurricane Maria
- Make sure you use DatetimeIndex for Part 3!
Final Exam

• Wednesday, Dec. 12 from 11:30am-2:30pm
• SENG 210
• Details online
Machine Learning

• Traditional Programming

Data -> Computer -> Output
Program

• Machine Learning

Data -> Computer
Output
...
Program
• Every machine learning algorithm has three components:
  - Representation
  - Evaluation
  - Optimization
Representation

- Decision trees
- Sets of rules / Logic programs
- Instances
- Graphical models (Bayes/Markov nets)
- Neural networks
- Support vector machines
- Model ensembles
- Etc.
Evaluation

- Accuracy
- Precision and recall
- Squared error
- Likelihood
- Posterior probability
- Cost / Utility
- Margin
- Entropy
- K-L divergence
- Etc.
Optimization

• Combinatorial optimization
  - E.g.: Greedy search
• Convex optimization
  - E.g.: Gradient descent
• Constrained optimization
  - E.g.: Linear programming
Types of Learning

• Supervised (inductive) learning
  - Training data includes desired outputs

• Unsupervised learning
  - Training data does not include desired outputs

• Semi-supervised learning
  - Training data includes a few desired outputs

• Reinforcement learning
  - Rewards from sequence of actions
Inductive Learning

• Given examples of a function \((X, F(X))\)
• Predict function \(F(X)\) for new examples \(X\)
  - Discrete \(F(X)\): Classification
  - Continuous \(F(X)\): Regression
  - \(F(X) = \text{Probability}(X)\): Probability estimation
Areas of Machine Learning

• Supervised learning
  - Decision tree induction
  - Rule induction
  - Instance-based learning
  - Bayesian learning
  - Neural networks
  - Support vector machines
  - Model ensembles
  - Learning theory

• Unsupervised learning
  - Clustering
  - Dimensionality reduction
How does this work in Python?

• Packages
  - scikit-learn: classical machine learning
  - keras: deep learning, works with Tensorflow and Theano
  - statsmodels: statistical models

• This lecture, we'll discuss scikit-learn

• Note the difference in the import name!
  - import sklearn
Features, Target, and Samples

Feature Matrix \((X)\)

- \(n_{\text{features}}\) →

Target Vector \((y)\)

- \(n_{\text{samples}}\) ←

- \(n_{\text{samples}}\) ←
Scikit-learn's Estimator API

1. Choose a class of model by importing the appropriate estimator class from Scikit-Learn.

2. Choose model hyperparameters by instantiating this class with desired values.

3. Arrange data into a features matrix and target vector following the discussion above.

4. Fit the model to your data by calling the fit() method of the model instance.

5. Apply the Model to new data:
   - For supervised learning, often we predict labels for unknown data using the predict() method.
   - For unsupervised learning, we often transform or infer properties of the data using the transform() or predict() method.

[Python Data Science Handbook]
Examples