PyPart

Project Repository: https://github.com/anthonyam12/pypart

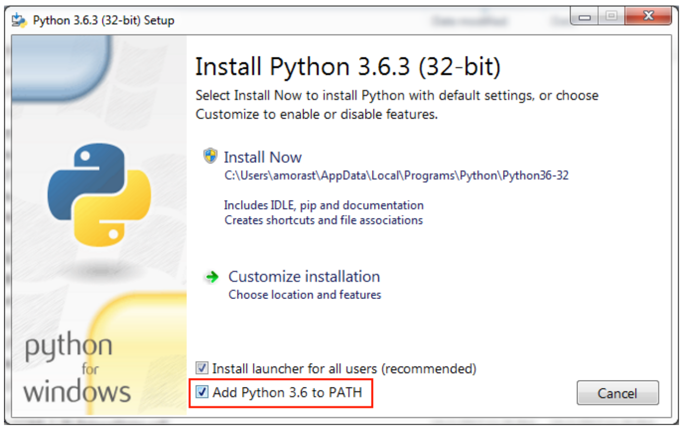
Python and Pandas Installation

The project was built with Python 3.6 and relies on the Pandas data analysis library for Python.

Python 3.6 can be installed from

https://www.python.org/downloads/

which provides an exe installer. When that's executed a screen similar to the one below is displayed, you'll want to make sure that the 'Add Python 3.6 to PATH' checkbox is selected so that pandas can be installed from Windows CMD, outlined in red below.



Python 3.6.3 (32-bit) Setup

Install Python 3.6.3 (32-bit)

Select Install Now to install Python with default settings, or choose Customize to enable or disable features.

Install Now
C:\Users\amorast\AppData\Local\Programs\Python\Python36-32

Includes IDLE, pip and documentation
Creates shortcuts and file associations

Customize installation
Choose location and features

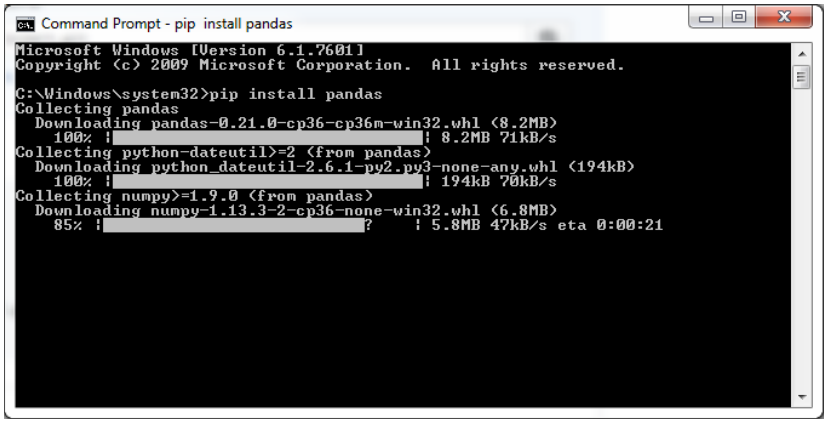
☑ Install launcher for all users (recommended)
☑ Add Python 3.6 to PATH

After the installation is complete the Windows' PATH Environment Variable should contain the install directory for Python so you're able to run 'python' and 'pip' commands in CMD and not receive an error about the command not being recognized.

With the environment variables set Pandas can be installed by opening CMD and typing

pip install pandas

which installs Pandas and all of its dependencies, as shown below.



Command Prompt - pip install pandas

```
Microsoft Windows [Version 6.1.7601]
Copyright (c) 2009 Microsoft Corporation.  All rights reserved.

C:\Windows\system32>pip install pandas
Collecting pandas
  Downloading pandas-0.21.0-cp36-cp36m-win32.whl (8.2MB)
    100% |████████████████████████████████| 8.2MB 71kB/s
Collecting python-dateutil>=2 (from pandas)
  Downloading python_dateutil-2.6.1-py2.py3-none-any.whl (194kB)
    100% |████████████████████████████████| 194kB 70kB/s
Collecting numpy>=1.9.0 (from pandas)
  Downloading numpy-1.13.3-2-cp36-none-win32.whl (6.8MB)
    85% |███████████████████████████?    | 5.8MB 47kB/s eta 0:00:21
```

Using PyPart

PyPart is most easily ran from the command line (CMD in Windows) via

```
python pypart.py <data file> <response variable name> [optional: <delayed value {0,1}>]
```

Here, the **<data file>** is the path to a csv file containing the dataset, **<response variable name>** is the column header for the response column, and an optional parameter **<delayed value>** taking a value of either 0 or 1. If no delayed value parameter is specified it is defaulted to 0.

An example run of the program on the cars dataset (cars.csv) for the non-delayed algorithm is as follows (assuming cars.csv is in the same directory as pypart.py)

```
python pypart.py cars.csv mpg 0
```

or

```
python pypart.py cars.csv mpg
```

To run the delayed algorithm the command is

```
python pypart cars.csv mpg 1
```

Output

The output of the pypart program is 4 CSV files and a '.tree' file, all of which are named after the dataset and are defined as follows,

1. **<dataset>.cptable.csv –** a CSV file holding CP table information. The contents of this file are what would be displayed if the user were to build a tree with rpart (call the output **tree**) and call **tree$cptable**.
2. **<dataset>.frame.csv** – similar to the CP table CSV, the frame CSV file contains the same information as an rpart object's **$frame** dataframe.
3. **<dataset>.splits.csv** – this CSV file will contain the information found in an rpart tree's **$splits** matrix.
4. **<dataset>.where.csv** – used for predicting the values of the original dataset, the where CSV file contains information found in an rpart object's **$where** vector. This is needed for the pruning of the tree in R.
5. **<dataset>.tree** – a file containing information about the tree and its splits. This will contain the same information as calling 'print(…)' on an rpart object in R.

In the cars example above, using **cars.csv** as the dataset, the 5 output files would be

1. cars.cptable.csv
2. cars.frame.csv
3. cars.spits.csv
4. cars.where.csv
5. cars.tree

Or, if the delayed algorithm is used, the tree file is renamed **cars.tree.delayed**, while the other file names stay the same.

Importing to R

The 4 CSV files listed above are needed to import the pypart output into R. In the project's root directory there is a subdirectory, **<project dir>/r**, containing an R script, **pypart_to_rpart.R**, that can be used to import the output of pypart into R and create (most of) an rpart object.

The script contains a single function call, **pypartToRpart(…)**, with usage,

pyTree <- pypartToRpart(<frame CSV>, <CP table CSV>, <where CSV>, <splits CSV>)

where the CSV files are the output of PyPart as defined above.

The scripts output is an object with the 'rpart' class to fool the rpart functions into accepting and manipulating the object.