MagikEDA Documentation

Release 0.1.3

Saul Berardo

September 29, 2015

Contents

MagikEDA is a python library which provides a set of useful functions to create graphs meant to be useful in Exploratory Data Analysis (EDA). It is concived to lay one layer above libraries such as Pandas and Basemap.

This documentation is work in progress. Please, come back later and perhaps there will be some descent docs here.

Contents:

CHAPTER 1

Install Instructions

1.1 Installing with PIP

Just type this:

```
pip install magikeda
```

1.2 Installing from Source

Just type this:

```
git clone https://github.com/saulberardo/MagikEDA.git magikeda
cd magikeda
python setup.py install
```

CHAPTER 2

Features

There are currently three packages available. The packages *univar* and *bivar* have functions for plotting one and two variables, respectively. The package *geo* has functions for calculating distances and plotting maps. The functions included are able of executing the following:

- Plotting a "data frame profile", which consists in a figure having one subplot for each variable in the dataset. Each subplot shows the variable distribution (histograms for numeric variables and bar charts for categorical).
- Plotting scatter plots with a tip box that is shown when points are hovered.
- Function to add an additional x axis bellow a figure.
- Function to compute the distance between a point and a segment (composed by two points).
- Function to plot a path (composed by a list o coordinates) over a map (the function automattically chooses a viewpoint to show the path (optionally with a padding).

CHAPTER 3

Modules

3.1 Univar

Functions to create graphs for ploting distributions of single variables.

univar.**add_extra_xaxis**(*fig*, *x*, *labels*, *padding=35*)

Add a x axis bellow the figure (indeed bellow the ax returned by fig.gca()) having the labels in the x positions.

The axis is added by first adding an entire new axes and the hiding all parts, except the xaxis.

Parameters

- **fig** (*Figure*) – The figure where to add the xaxis.
- **x** (*list*) – List of numbers specifying the x positions.
- **labels** (*list*) – List of strings specifying the labels to place in the x positions.
- **padding** (*int*) – How much space should be added between the figure and the new x axis bellow it.

Returns **new_ax** – Return the axes added to the image (where the x axis was plotted).

Return type Axes

univar.**plot_bar_chart**(*data*, *cmap='Accent'*, *color=None*, *xlabel=''*, *ylabel=''*, *title=''*, *width=0.9*, *xticks_rotation=0*, *series_legends=None*, *ax=None*)

Plot a barplot showing the distribution of a categorical variable (for one or multiple series of data).

It's similar to series.value_counts().plot(kind='bar') but can also be used with multiple serires at the same time (a lista of series in data).

Parameters

- **data** (*List*) – Pandas series (with categorical data) or list of series. For multiple series, each categories are grouped together.
- **cmap** (*str*) – Colormap to use for determining each category color. Default is 'Accent'.
- **color** (*Color*) – color to use in the graph (overrides colormap colors). Default is None, which means not to override colormap colors.
- **xlabel** (*str*) – Label of the x axis.
- **ylabel** (*str*) – Label of the x axis.
- **title** (*str*) – Figure title.
- **width** (*float*) – Width of each bar. Default is 0.9

- **xticks_rotation** (*float*) – Degrees to rotate x labels.
- **series_legends** (*List*) – Lgends to identify each series.
- **legend** (*bool*) – Whether to show or not the legend box.
- **ax** – Axes in which to plot the graph

Returns **ax** – Axes where the figure has been plotted.

Return type matplotlib.axes_subplots.AxesSubplot

univar.**plot_dataframe_profile**(*data_frame*, *include_cols=None*, *exclude_cols=None*, *shape=None*, *hspace=0.4*, *wspace=0.2*, *default_xticks_rotation=0*, *xticks_rotation={}*, *default_color='blue'*, *colors={}*, *bins=30*, *title=''*, *titles={}*, *default_ylabel=''*, *ylabels={}*, *default_xlabel=''*, *xlabels={}*)

Plot a grid of subplots in which each subplot shows the distribution of a variable of the data frame.

For numerical variables a histogram is shown. For categorical, a barplot whose bars lengths are proportional to the percentage of each category in the column. The subplots are drawn in the default column order in the dataframe, or in the order specified in include_cols.

Parameters

- **data_frame** (*pandas.Dataframe*) – The Pandas data frame whose columns will be plotted.
- **include_cols** (*list*) – A list with the column names that should be included in the plot. Default is None, which means all columns. If this parameter is used, just the columns specified are included, in the given order.
- **remove_cols** (*list*) – A list with the column names that should not be included in the plot. Columns listed here will be removed even if included in the include_cols list.
- **shape** (*tuple*) – A (rows, columns) tuples indicating the shape of the grid. Default is to use the square root of the number of variables as the number of columns, and as many rows as necessary to show all variables.
- **hspace** (*float*) – Vertical space between subplots.
- **wspace** (*float*) – Horizontal space betwwn subplots.
- **default_xticks_rotation** (*float*) – Default rotation in degrees of the category names in bar graphs.
- **xticks_rotation** (*dict*) – Dictionary associating column names to xtick label orientation (in degrees).
- **default_color** (*str*) – Default color name for all subplots.
- **colors** (*dict*) – Dictionary associating column names to colors (override defaults color).
- **title** (*str*) – General title to use above all subplots.
- **titles** (*dict*) – Dictionary associating column names to titles to use in each subplot. If a value is not specified for a column, the colum name is used as title.
- **default_ylabel** (*str*) – Default lable to use in the y axis.
- **ylabels** (*dict*) – Dictionary associating columnnames to y labels to use in each subplot. If a value is not specified for a column, the default_ylabel is used.
- **default_xlabel** (*str*) – Default lable to use in the y axis.

- **xlabels** (*dict*) – Dictionary associating columnnames to y labels to use in each subplot. If a value is not specified for a column, the default_ylabel is used.

Returns **gs** – Return the GridSpec created with the subplots.

Return type GridSpec

3.2 Bivar

Functions to create graphs for ploting distributions of two variables at the same time.

TODO: Instead of calling plot for each point, we should use directly plt.scatter function and then get the points and add the annotations.

bivar.**drawTipBox**(*x*, *y*, *text*, *color*, *ax*, *alpha=0.9*)

Adds a tootip box with text in he point on the ax specified with the color and alpha parameters.

bivar.**plot_mouse_over_scatter**(*x*, *y*, *tips*, *box_color=(0.95*, *0.9*, *1)*, *c=None*, *marker='o'*, *ax=None*, ***kwargs*)

Draw a scatter plot adding a tooltip to be show when the mouse is passes over the point.

The function calls plt.plot(...) method to each point separatedly and adds a hidden (by default) annotation over it. The annotation visibility is changed to True when the point is hovered.

Parameters

- **x** (*list (of floats)*) – List of x coordinates.
- **y** (*list (of floats)*) – List of y coordinates.
- **tips** (*list (of strings)*) – List of the string that will be shown when the mouse is over the point.
- **box_collor** (*color*) – RGB color of the box behind the text. Default is (0.95, 0.90, 1).
- **c** (*TODO*) – TODO
- **marker** (*Any valid marker symbol used in matplotlib scatter plots.*) – The marker symbol.
- **; Axes** (*ax*) – The ax where the scatter will be plotted. Default is None.
- **kwargs** (*dict or kwargs*) – Key word args to be passed to plt.plot(...) function.

3.3 Geo

Functions to plot maps and calculate distances.

TODO:

- Replace box value given in aspect_ratio by a box parameter
- make color attribute work with connect_points
- Return ax

geo.**distance_from_point_to_segment**(*px*, *py*, *x1*, *y1*, *x2*, *y2*)

Returns the minimum distance between a point (px, py) and a line segment (x1, y1, x2, y2).

Method adapted from Map Rantala, available at: https://nodedangles.wordpress.com/2010/05/16/measuring-distance-from-a-point-to-a-line-segment/

geo.**haversine**(*lon1*, *lat1*, *lon2*, *lat2*)

Calculate the great circle distance between two points on the earth (specified in decimal degrees)

This function found was found in many different sites on the internet.

geo.**plot_path_from_above**(*lon*, *lat*, *padding=1.0*, *show_map=True*, *color='red'*, *aspect_ratio=0.75*, *ax=None*)

Plot a path (a list of points) over a map using maps provided by basemap.

This functions uses very simple Basemap standard parameters. More options to customize the map should be added in the future.

Parameters

- **lon** (*pd.Series*) – Series containing the list of longitudes.
- **lat** (*pd.Series*) – Series containing the list of latitudes.
- **padding** (*float*) – Space to add arond path (in degrees). Default is 2 degrees.
- **show_map** (*boolean*) – Whether to draw the map around the path. Default is True.
- **color** (*color or list of colors.*) – The color of the path. Default is red. Optionally, if the parameter connect_poiints is False, a list of colors can be passed (having the same size as the number of dots). Example:

```
>>> colors = plt.cm.get_cmap('coolwarm', number_of_colors)
>>> color_idx = [ int(i) for i in color_values ]
>>> geo.plot_path_from_above(lon, lat, color=colors(color_idxs))
```

- **aspect_ratio** (*float or str or tuple*) – Aspect ration of the map. Default is 3/4. The values 'square' and 'fit_extremities' can also be specified. The first creates a square map. The second, adds just enough space to fit the extremities (still adding the padding). The cordinates of the box containing the map can also be specified as (min_lat, max_lat, min_lon, max_lon).
- **ax** (*AxesSubplot*) – Axes where to plot the map.

CHAPTER 4

Indices and tables

- genindex
- modindex

b

g

u

A

B

D

G

H

P

U