eflow

Eric Cacciavillani

Dec 13, 2020

CONTENTS

—

eflow.data_analysis.feature_analysis
eflow.data_analysis.null_analysis

CHAPTER

ONE

FEATUREANALYSIS

from eflow.data_analysis.feature_analysis import FeatureAnalysis

class FeatureAnalysis(*df_features*, *dataset_sub_dir=''*, *dataset_name=''*, *overwrite_full_path=None*, *notebook_mode=False*)

Analyzes the feature data of a pandas Dataframe object. (Ignores null data for displaying data and creates 2d graphics with 2 features. In the future I might add 3d graphics with 3 features.)

analyze_feature(*df*, *feature_name*, *dataset_name*, *target_feature=None*, *display_visuals=True*, *display_print=True*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *extra_tables=True*)

Generate's all graphic's for that given feature and the relationship to the target feature.

Args:

df: pd.Dataframe Pandas DataFrame object

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

target_feature: string Will create graphics involving this feature with the main feature 'feature_name'.

display_visuals: string Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Saves file if set to True; doesn't if set to False.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

Raises: Raises error if the json file's snapshot of the given dataframe doesn't match the given dataframe.

descr_table(*df*, *feature_name*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*)

Creates/Saves a pandas dataframe of a feature's numerical data. Standard deviation, mean, Q1-Q5, median, variance, etc.

Note Creates a png of the table.

Args:

df: pd.Dataframe Pandas DataFrame object

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Saves file if set to True; doesn't if set to False.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

group_by_feature_value_count_table(*df*, *feature_name*, *dataset_name*, *other_feature_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*)

Creates/Saves a pandas dataframe of features and their found types in the dataframe.

Note Creates a png of the table.

Args:

df: pd.Dataframe Pandas DataFrame object

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

other_feature_name: string Feature to compare to.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Saves file if set to True; doesn't if set to False.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

perform_analysis(*df*, *dataset_name*, *target_features=None*, *display_visuals=True*, *display_print=True*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*)

Performs all public methods that generate visualizations/insights about the data.

Note: Pretty much my personal lazy button for running the entire object without specifying any method in particular.

Args:

df: pd.Dataframe Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

target_features: collection of strings or None A feature name that both exists in the init df_features and the passed dataframe.

Note If init to 'None' then df_features will try to extract out the target feature.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

Raises: If an empty dataframe is passed to this function or if the same dataframe is passed to it raise error.

plot_count_graph(*df*, *feature_name*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *flip_axis=False*, *palette='PuBu'*)

Display a barplot with color ranking from a feature's value counts from the seaborn libary and save the graph in the correct directory structure.

Args:

df: pd.Dataframe Pandas dataframe object.

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string Name to give the file.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

display_print: bool Determines whether or not to print function's embedded print statements.

figsize: tuple Size for the given plot.

flip_axis: bool Flip the axis the ploting axis from x to y if set to 'True'.

palette: dict or string String representation of color pallete for ranking from seaborn's pallete.

Credit to seaborn's author: Michael Waskom Git username: mwaskom Link: http://tinyurl.com/y4pzrgcf

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_distance_graph(*df*, *feature_name*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *bins=None*, *norm_hist=True*, *hist=True*, *kde=True*, *colors=None*, *fit=None*, *fit_kws=None*)

Display a distance plot and save the graph in the correct directory.

Args:

df: pd.Dataframe Pandas dataframe object

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string Name to give the file.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

figsize: tuple The given size of the plot.

bins: int Specification of hist bins, or None to use Freedman-Diaconis rule.

norm_hist: bool If True, the histogram height shows a density rather than a count. This is implied if a KDE or fitted density is plotted.

hist: bool Whether to plot a (normed) histogram.

kde: bool Whether to plot a gaussian kernel density estimate.

colors [matplotlib color] Color to plot everything but the fitted curve in.

fit: functional method An object with fit method, returning a tuple that can be passed to a pdf method a positional arguments following an grid of values to evaluate the pdf on.

fit_kws [dictionaries, optional] Keyword arguments for underlying plotting functions.

Credit to seaborn's author: Michael Waskom Git username: mwaskom Doc Link: http://tinyurl.com/ycco2hok

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_jointplot_graph(*df*, *feature_name*, *dataset_name*, *other_feature_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *color=None*, *kind='scatter and kde'*, *ratio=5*)

Display a ridge plot and save the graph in the correct directory.

Args:

df: pd.Dataframe Pandas DataFrame object.

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

other_feature_name: string Feature to compare to.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

display_print: bool Determines whether or not to print function's embedded print statements.

figsize: tuple Tuple object to represent the plot/image's size. Because joinplot only accepts a single value for the figure; we just pull the greatest of the two values.

color: string Seaborn/maplotlib color/hex color for representing the graph

kind: string (scatter,reg,resid,kde,hex,scatter and kde) Kind of plot to draw.

ratio: Ratio of joint axes height to marginal axes height. (Determines distplot like plots dimensions.)

Credit to seaborn's author: Michael Waskom Git username: mwaskom Link: http://tinyurl.com/v9pxsoy

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_multi_bar_graph(*df*, *feature_name*, *dataset_name*, *other_feature_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *colors=None*, *stacked=False*)

Display a pie graph and save the graph in the correct directory.

Args:

df: Pandas DataFrame object.

feature_name: Specified feature column name.

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: Specify the sub directory to append to the pre-defined folder path.

save_file: Boolean value to whether or not to save the file.

dataframe_snapshot: Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

figsize: tuple Size of the plot.

colors: dict or string Dictionary of all feature values to hex color values.

stacked: bool Determines if the multi bar graph should be stacked or not.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_pie_graph(*df*, *feature_name*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *pallete=None*)

Display a pie graph and save the graph in the correct directory.

Args:

df: Pandas DataFrame object.

feature_name: Specified feature column name.

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: Specify the sub directory to append to the pre-defined folder path.

save_file: Boolean value to whether or not to save the file.

dataframe_snapshot: Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

figsize: tuple Size of the plot.

pallete: dict or string Dictionary of all feature values to hex color values.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_ridge_graph(*df*, *feature_name*, *dataset_name*, *other_feature_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *palette=None*)

Display a ridge plot and save the graph in the correct directory.

Args:

df: pd.Dataframe Pandas DataFrame object.

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

other_feature_name: string Feature to compare to.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

display_print: bool Determines whether or not to print function's embedded print statements.

figsize: tuple Tuple object to represent the plot/image's size.

palette: dict or string Dictionary of all feature values to hex color values.

Note - A large part of this was taken from: http://tinyurl.com/tuou2cn

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

plot_violin_graph(*df*, *feature_name*, *dataset_name*, *other_feature_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *figsize=(13, 10)*, *order=None*, *cut=2*, *scale='area'*, *gridsize=100*, *width=0.8*, *palette=None*, *saturation=0.75*)

Display a violin plot and save the graph in the correct directory.

Args:

df: pd.Dataframe Pandas dataframe object

feature_name: string Specified feature column name to compare to y.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

other_feature_name: string Specified feature column name to compare to x.

display_visuals: bool Boolean value to whether or not to display visualizations.

filename: string Name to give the file.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

display_print: bool Determines whether or not to print function's embedded print statements.

figsize: tuple Size of the given plot.

order: lists of strings Order to plot the categorical levels in, otherwise the levels are inferred from the data objects.

cut: float Distance, in units of bandwidth size, to extend the density past the extreme datapoints. Set to 0 to limit the violin range within the range of the observed data. (i.e., to have the same effect as trim=True in ggplot.)

scale: string {area, count, width} The method used to scale the width of each violin. If area, each violin will have the same area. If count, the width of the violins will be scaled by the number of observations in that bin. If width, each violin will have the same width.

gridsize: int Number of points in the discrete grid used to compute the kernel density estimate.

width: float Width of a full element when not using hue nesting, or width of all the elements for one level of the major grouping variable.

palette: dict or string Colors to use for the different levels of the hue variable. Should be something that can be interpreted by color_palette(), or a dictionary mapping hue levels to matplotlib colors.

saturation: float Proportion of the original saturation to draw colors at. Large patches often look better with slightly desaturated colors, but set this to 1 if you want the plot colors to perfectly match the input color spec.

Credit to seaborn's author: Michael Waskom Git username: mwaskom Doc link: http://tinyurl.com/y3hxxzgv

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

statistical_analysis_on_aggregates(*df*, *target_features*, *dataset_name*, *dataframe_snapshot=True*)

Aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

Args:

df: pd.Dataframe Pandas DataFrame object.

target_features: list of string Specified target features.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

Note: This function has a lot going on and it's infancy so I am going to purposely not give it suppress_runtime_errors so people will find problems with it and report it to me.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

value_counts_table(*df*, *feature_name*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*)

Creates a value counts table of the features given data.

Note Creates a png of the table.

Args:

df: pd.Dataframe Pandas DataFrame object

feature_name: string Specified feature column name.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

save_file: bool Saves file if set to True; doesn't if set to False.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

Creates/Saves a pandas dataframe of value counts of a dataframe.

Note - Creates a png of the table.

Raises: Raises error if the feature data is filled with only nulls or if the json file's snapshot of the given dataframe doesn't match the given dataframe.

CHAPTER

TWO

NULLANALYSIS

from eflow.data_analysis.null_analysis import NullAnalysis

class NullAnalysis(*df_features*, *dataset_sub_dir=''*, *dataset_name='Default Dataset Name'*, *overwrite_full_path=None*, *notebook_mode=False*)

Analyzes a pandas dataframe's object for null data; creates visuals like graphs and tables.

feature_analysis_of_null_data(*df*, *dataset_name*, *target_features=None*, *display_visuals=True*, *display_print=True*, *save_file=True*, *suppress_runtime_errors=True*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*, *nan_features=[]*)

Performs all public methods that generate visualizations/insights that feature analysis uses on an aggregation of null data in a feature.

Note: Pretty much my personal lazy button for running the entire object without specifying any method in particular.

Args:

df: pd.Dataframe Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

target_features: collection of string or None A feature name that both exists in the init df_features and the passed dataframe.

Note If init to 'None' then df_features will try to extract out the target feature.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

nan_features: collection of strings Features names that must contain nan data to aggregate on.

Raises: If an empty dataframe is passed to this function or if the same dataframe is passed to it raise error.

missing_values_table(*df*, *dataset_name*, *display_visuals=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *display_print=True*)

Creates/Saves a Pandas DataFrame object giving the percentage of the null data for the original DataFrame columns.

Args:

df: pd.Dataframe Pandas DataFrame object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

perform_analysis(*df*, *dataset_name*, *display_visuals=True*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *display_print=True*, *null_features_only=False*)

Perform all public methods of the NullAnalysis object. Except for feature_analysis_of_null_data.

Args:

df: pd.Dataframe Pandas Dataframe object.

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

null_features_only: bool Dataframe will pass on null features for the visualizations

plot_null_bar_graph(*df*, *dataset_name*, *display_visuals=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *display_print=True*, *null_features_only=False*, *figsize=(24, 10)*, *fontsize=16*, *labels=None*, *log=False*, *color='#072F5F'*, *inline=False*, *filter=False*, *n=0*, *p=0*, *sort=None*)

Desc (Taken from missingno): A bar graph visualization of the nullity of the given DataFrame then pushes the image to output folder.

Args:

df: pd.Dataframe Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

null_features_only: bool Dataframe will pass on null features for the visualizations

Please read the offical documentation for more about the parameters: Link - https://github.com/ResidentMario/missingno

Note - Changed the default color of the bar graph because I thought it was ugly.

plot_null_dendrogram_graph(*df*, *dataset_name*, *display_visuals=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *display_print=True*, *null_features_only=False*, *method='average'*, *filter=None*, *n=0*, *p=0*, *orientation=None*, *figsize=(24, 10)*, *fontsize=16*, *inline=False*)

Desc (Taken from missingno): Fits a *scipy* hierarchical clustering algorithm to the given DataFrame's variables and visualizes the results as a *scipy* dendrogram.

Args:

df: Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

null_features_only: bool Dataframe will pass on only null features for the visualizations

Please read the offical documentation for more about the parameters: Link: https://github.com/ResidentMario/missingno

plot_null_heatmap_graph(*df*, *dataset_name*, *display_visuals=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *display_print=True*, *inline=False*, *filter=None*, *n=0*, *p=0*, *sort=None*, *figsize=(24, 10)*, *fontsize=16*, *labels=True*, *cmap='RdBu'*, *vmin=-1*, *vmax=1*, *cbar=True*)

Desc (Taken from missingno): Presents a *seaborn* heatmap visualization of nullity correlation in the given DataFrame.

Args:

df: pd.Dataframe Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

Please read the offical documentation for more about the parameters: Link: https://github.com/ResidentMario/missingno

Note: Changed the default color of the bar graph because I thought it was ugly.

plot_null_matrix_graph(*df*, *dataset_name*, *display_visuals=True*, *display_print=True*, *filename=None*, *sub_dir=None*, *save_file=True*, *dataframe_snapshot=True*, *suppress_runtime_errors=True*, *null_features_only=False*, *filter=None*, *n=0*, *p=0*, *sort=None*, *figsize=(24*, *10)*, *width_ratios=(15*, *1)*, *color=(0.027*, *0.184*, *0.373)*, *fontsize=16*, *labels=None*, *sparkline=True*, *inline=False*, *freq=None*)

Desc (Taken from missingno): A matrix visualization of the nullity of the given DataFrame then pushes the image to output folder.

Args:

df: pd.Dataframe Pandas dataframe object

dataset_name: string The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

filename: string If set to 'None' will default to a pre-defined string; unless it is set to an actual filename.

sub_dir: string Specify the sub directory to append to the pre-defined folder path.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

null_features_only: bool Dataframe will pass on null features for the visualizations

Please read the offical documentation at for more about the parameters: Link: https://github.com/ResidentMario/missingno

Note: Changed the default color of the bar graph because I thought it was ugly.

eflow.data_pipeline_segments. data_encoder
eflow.data_pipeline_segments. feature_data_cleaner
eflow.data_pipeline_segments. feature_transformer
eflow.data_pipeline_segments. string_cleaner

CHAPTER

THREE

DATAENCODER

from eflow.data_pipeline_segments.data_encoder import DataEncoder

class DataEncoder(*segment_id=None*, *create_file=True*)

Attempts to convert features to the correct types. Will update the dataframe and df_features.

apply_value_representation(*df*, *df_features*, *_add_to_que=True*)

Translate features into most understandable/best representation

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

decode_data(*df*, *df_features*, *apply_value_representation=True*, *_add_to_que=True*)

Decode the data into non-numerical values for more descriptive analysis.

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

apply_value_representation: bool Translate features into most understandable/best representation/

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

encode_data(*df*, *df_features*, *apply_value_representation=True*, *_add_to_que=True*)

Encode the data into numerical values for machine learning processes.

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

apply_value_representation: bool Translate features into most understandable/best representation/

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

make_dummies(*df*, *df_features*, *qualitative_features=[]*, *_feature_values_dict=None*, *_add_to_que=True*)

Create dummies features of based on qualtative feature data and removes the original feature.

Note _feature_values_dict does not need to be init. Used for backend resource.

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

qualtative_features: collection of strings Feature names to convert the feature data into dummy features.

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

make_values_bool(*df*, *df_features*, *_add_to_que=True*)

Convert all string bools to numeric bool value

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

revert_dummies(*df*, *df_features*, *qualitative_features=[]*, *_add_to_que=True*)

Convert dummies features back to the original feature.

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

qualitative_features: collection of strings Feature names to convert the dummy features into original feature data.

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

revert_value_representation(*df*, *df_features*, *_add_to_que=True*)

Translate features back into worst representation

Args:

df: pd.Dataframe Pandas dataframe.

df_features: DataFrameTypes from eflow DataFrameTypes object.

_add_to_que: bool Hidden variable to determine if the function should be pushed to the pipeline segment.

CHAPTER

FOUR

FEATUREDATACLEANER

from eflow.data_pipeline_segments.feature_data_cleaner import FeatureDataCleaner

class FeatureDataCleaner(*segment_id=None*, *create_file=True*)

Designed for a multipurpose data cleaner.

drop_feature(*df*, *df_features*, *feature_name*, *_add_to_que=True*)

Drop a feature in the dataframe.

Args:

df: pd.Dataframe Pandas Dataframe

df_features: DataFrameType from eflow Organizes feature types into groups.

feature_name: string Name of the feature in the datatframe

_add_to_que: bool Pushes the function to pipeline segment parent if set to 'True'.

fill_nan_by_distribution(*df*, *df_features*, *feature_name*, *percentile*, *z_score=None*, *_add_to_que=True*)

Fill nan by the distribution of data.

Args:

df: pd.Dataframe Pandas Dataframe

df_features: DataFrameType from eflow Organizes feature types into groups.

feature_name: string Name of the feature in the datatframe

percentile: float or int

z_score:

_add_to_que: bool Pushes the function to pipeline segment parent if set to 'True'.

ignore_feature(*df*, *df_features*, *feature_name*, *_add_to_que=True*)

Ignore the given feature.

Args:

df: pd.Dataframe Pandas Dataframe

df_features: DataFrameType from eflow Organizes feature types into groups.

feature_name: string Name of the feature in the datatframe

_add_to_que: bool Pushes the function to pipeline segment parent if set to 'True'.

make_nan_assertions(*df*, *df_features*, *feature_name*, *_add_to_que=True*)

Make nan assertions for boolean features.

Args:

df: pd.Dataframe Pandas Dataframe

df_features: DataFrameType from eflow Organizes feature types into groups.

feature_name: string Name of the feature in the datatframe

_add_to_que: bool Pushes the function to pipeline segment parent if set to 'True'.

remove_nans(*df*, *df_features*, *feature_name*, *_add_to_que=True*)

Remove rows of data based on the given feature.

Args:

df: pd.Dataframe Pandas Dataframe

df_features: DataFrameType from eflow Organizes feature types into groups.

feature_name: string Name of the feature in the datatframe

_add_to_que: bool Pushes the function to pipeline segment parent if set to 'True'.

run_widget(*df*, *df_features*, *nan_feature_names=[]*)

df: A pandas dataframe object

df_features: DataFrameTypes object; organizes feature types into groups.

Returns: Returns a UI widget to create a JSON file for cleaning.

CHAPTER

FIVE

FEATURETRANSFORMER

from eflow.data_pipeline_segments.feature_transformer import FeatureTransformer

class FeatureTransformer(*segment_id=None*, *create_file=True*)

Combines, removes, scales, etc features of a pandas dataframe.

remove_features(*df*, *df_features*, *feature_names*, *_add_to_que=True*)

Removes unwanted features from the dataframe and saves them to the pipeline segment structure if _add_to_que is set to True.

Args:

df: Pandas Dataframe to update.

df_features: DataFrameTypes object to update.

feature_names: Features to remove

_add_to_que: Pushes the function to pipeline segment parent if set to 'True'.

CHAPTER

SIX

STRINGCLEANER

from eflow.data_pipeline_segments.string_cleaner import StringCleaner

class StringCleaner(*segment_id=None*, *create_file=True*)

eflow.foundation.data_frame_types
eflow.foundation.data_pipeline

eflow.model_analysis. classification_analysis
eflow.model_analysis.outlier_analysis
eflow.model_analysis. regression_analysis

CHAPTER

SEVEN

CLASSIFICATIONANALYSIS

from eflow.model_analysis.classification_analysis import ClassificationAnalysis

class ClassificationAnalysis(*dataset_name*, *model*, *model_name*, *feature_order*, *target_feature*, *pred_funcs_dict*, *df_features*, *sample_data*, *project_sub_dir='Classification Analysis'*, *overwrite_full_path=None*, *target_classes=None*, *save_model=True*, *notebook_mode=False*)

Analyzes a classification model's result's based on the prediction function(s) passed to it. Creates graphs and tables to be saved in directory structure.

classification_correct_analysis(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *aggerate_target=False*, *display_print=True*, *suppress_runtime_errors=True*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*)

Compares the actual target value to the predicted value and performs analysis of all the data.

Args:

X: np.matrix or lists of lists Feature matrix.

y: list or np.array Target data vector.

pred_name: str The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: str The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

feature_order: collection object Features names in proper order to re-create the pandas dataframe.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

classification_error_analysis(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *aggerate_target=False*, *display_print=True*, *suppress_runtime_errors=True*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*)

Compares the actual target value to the predicted value and performs analysis of all the data.

Args:

X: np.matrix or lists of lists Feature matrix.

y: list or np.array Target data vector.

pred_name: str The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: str The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

feature_order: collection object Features names in proper order to re-create the pandas dataframe.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

classification_metrics(*X, y, pred_name, dataset_name, thresholds=None, display_visuals=True, save_file=True, title='', custom_metrics_dict={}, ignore_metrics=[], average_scoring=['micro', 'macro', 'weighted']*)

Creates a dataframe based on the prediction metrics of the feature matrix and target vector.

Args:

X: Feature matrix.

y: list or np.array Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Display tables.

save_file: Determines whether or not to save the generated document.

title: Adds to the column 'Metric Score'.

custom_metrics_dict: Pass the name of metric(s) and the function definition(s) in a dictionary.

ignore_metrics: Specify the default metrics to not apply to the classification data_analysis.

- Precision
- MCC
- Recall
- F1-Score
- Accuracy

average_scoring:

Determines the type of averaging performed on the data.

- micro
- macro
- weighted

Returns: Return a dataframe object of the metrics value.

classification_report(*X, y, pred_name, dataset_name, thresholds=None, display_visuals=True, save_file=True*)

Creates a report of all target's metric evaluations based on the model's prediction output from the classification report from the sklearn.

Args:

X: Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

graph_model_importances(*feature_order*, *feature_importances*, *display_visuals=True*)

Graph given models's feature importances

Args:

feature_order: list Features names in proper order to re-create the pandas dataframe.

feature_importances: list List of floats that represent each corresponding features importance

display_visuals: bool Visualize graph if needed.

perform_analysis(*X, y, dataset_name, thresholds_matrix=None, classification_error_analysis=False, classification_correct_analysis=False, ignore_metrics=[], custom_metrics_dict={}, average_scoring=['micro', 'macro', 'weighted'], display_visuals=True*)

Runs all available analysis functions on the models predicted data.

Args:

X: Feature matrix.

y: Target data vector.

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds_matrix: List of list/matrix of thresholds.

If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

classification_error_analysis: bool Perform feature analysis on data that was incorrectly predicted.

classification_correct_analysis: bool Perform feature analysis on data that was correctly predicted.

figsize: All plot's dimension's.

ignore_metrics: Specify the default metrics to not apply to the classification data_analysis.

- Precision
- MCC

- Recall
- F1-Score
- Accuracy

custom_metrics_dict: Pass the name of metric(s) with the function definition(s) in a dictionary.

average_scoring: Determines the type of averaging performed on the data.

display_visuals: Controls visual display of error error data_analysis if it is able to run.

Returns: Performs all classification functionality with the provided feature data and target data.

- plot_precision_recall_curve
- classification_evaluation
- plot_confusion_matrix

plot_confusion_matrix(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *normalize=False*, *hide_zeros=False*, *hide_counts=False*, *x_tick_rotation=0*, *ax=None*, *figsize=(13, 10)*, *cmap='Blues'*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc Creates a confusion matrix plot based on the models predictions.

Args:

X:

Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

plot_cumulative_gain(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *ax=None*, *figsize=(13, 10)*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc Plots calibration curves for a set of classifier probability estimates.

Args:

X:

Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

plot_ks_statistic(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *ax=None*, *figsize=(13, 10)*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc Generates the KS Statistic plot from labels and scores/probabilities.

Args:

X:

Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

plot_lift_curve(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *ax=None*, *figsize=(13, 10)*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc The lift curve is used to determine the effectiveness of a binary classifier. A detailed explanation can be found at http://tinyurl.com/csegj9. The implementation here works only for binary classification.

Args:

X:

Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

plot_precision_recall_curve(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *plot_micro=True*, *classes_to_plot=None*, *ax=None*, *figsize=(13, 10)*, *cmap='nipy_spectral'*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc Plots precision recall curve plot based on the models predictions.

Args:

X: Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

plot_roc_curve(*X*, *y*, *pred_name*, *dataset_name*, *thresholds=None*, *display_visuals=True*, *save_file=True*, *title=None*, *ax=None*, *figsize=(13, 10)*, *title_fontsize='large'*, *text_fontsize='medium'*)

From scikit-plot documentation Link: http://tinyurl.com/y3ym5pyc Creates ROC curves from labels and predicted probabilities.

Args:

X: Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Visualize graph if needed.

save_file: Boolean value to whether or not to save the file.

CHAPTER

EIGHT

EFLOW.MODEL_ANALYSIS.OUTLIER_ANALYSIS

Functions

check_if_directory_exists(directory_path)	Checks if the given directory path exists. Raises an error if doesn't
convert_to_filename(filename[, ...])	Attempts to make the filename string valid.
correct_directory_path(directory_path)	Attempts to convert the directory path to a proper one by removing
create_dir_structure(directory_path, ...)	Creates required directory structures inside the parent
create_unique_directory(directory_path, ...)	Creates a unique folder in the proper directory structure.
dict_to_json_file(dict_obj, directory_path, ...)	Writes a dict to a json file.
get_all_directories_from_path(directory_path)	Gets directories names with the provided path.
get_all_files_from_path(directory_path[, ...])	Gets all filenames with the provided path.
get_unique_directory_path(directory_path, ...)	Iterate through directory structure until a unique folder name can be
json_file_to_dict(filepath)	Returns back the dictionary from of a json file.
load_pickle_object(file_path)	
pickle_object_to_file(obj, directory_path, ...)	Writes the object to a pickle file.
write_object_text_to_file(obj, ...[, ...])	Writes the object's string representation to a text file.
zcore_remove_outliers(df, feature_name, ...)	Any zscore that is between the negative and positive of the 'zscore_val' will be return as a pandas series object.

Classes

ModelAnalysis(dataset_name[, ...])	All objects in model_analysis folder of eflow are related to this object.
OutlierAnalysis(dataset_name, model, ...[, ...])	Analyzes a classification model's result's based on the prediction function(s) passed to it.

class OutlierAnalysis(*dataset_name*, *model*, *model_name*, *feature_order*, *df_features*, *project_sub_dir='Outlier Analysis'*, *overwrite_full_path=None*, *save_models=True*, *notebook_mode=False*)

Analyzes a classification model's result's based on the prediction function(s) passed to it. Creates graphs and tables to be saved in directory structure.

CHAPTER

NINE

REGRESSIONANALYSIS

from eflow.model_analysis.regression_analysis import RegressionAnalysis

class RegressionAnalysis(*dataset_name*, *model*, *model_name*, *feature_order*, *target_feature*, *pred_funcs_dict*, *df_features*, *project_sub_dir='Regression Analysis'*, *overwrite_full_path=None*, *save_model=True*, *notebook_mode=False*)

Analyzes a classification model's result's based on the prediction function(s) passed to it. Creates graphs and tables to be saved in directory structure.

perform_analysis(*X*, *y*, *dataset_name*, *regression_error_analysis=False*, *regression_correct_analysis=False*, *ignore_metrics=[]*, *custom_metrics_dict={}*, *display_visuals=True*, *mse_score=None*)

Runs all available analysis functions on the models predicted data.

Args:

X: Feature matrix.

y: Target data vector.

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

regression_error_analysis: bool Perform feature analysis on data that was incorrectly predicted.

regression_correct_analysis: bool Perform feature analysis on data that was correctly predicted.

ignore_metrics: Specify the default metrics to not apply to the classification data_analysis.

-
-
-
-
-

custom_metrics_dict: Pass the name of metric(s) with the function definition(s) in a dictionary.

display_visuals: Controls visual display of error error data_analysis if it is able to run.

Returns: Performs all classification functionality with the provided feature data and target data.

- plot_precision_recall_curve
- classification_evaluation
- plot_confusion_matrix

regression_correct_analysis(*X*, *y*, *pred_name*, *dataset_name*, *mse_score*, *display_visuals=True*, *save_file=True*, *display_print=True*, *suppress_runtime_errors=True*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*)

Compares the actual target value to the predicted value and performs analysis of all the data.

Args:

X: np.matrix or lists of lists Feature matrix.

y: collection object Target data vector.

pred_name: str The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: str The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

feature_order: collection object Features names in proper order to re-create the pandas dataframe.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

regression_error_analysis(*X*, *y*, *pred_name*, *dataset_name*, *mse_score*, *display_visuals=True*, *save_file=True*, *display_print=True*, *suppress_runtime_errors=True*, *aggregate_target_feature=True*, *selected_features=None*, *extra_tables=True*, *statistical_analysis_on_aggregates=True*)

Compares the actual target value to the predicted value and performs analysis of all the data.

Args:

X: np.matrix or lists of lists Feature matrix.

y: collection object Target data vector.

pred_name: str The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: str The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

feature_order: collection object Features names in proper order to re-create the pandas dataframe.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: bool Boolean value to whether or not to display visualizations.

display_print: bool Determines whether or not to print function's embedded print statements.

save_file: bool Boolean value to whether or not to save the file.

dataframe_snapshot: bool Boolean value to determine whether or not generate and compare a snapshot of the dataframe in the dataset's directory structure. Helps ensure that data generated in that directory is correctly associated to a dataframe.

suppress_runtime_errors: bool If set to true; when generating any graphs will suppress any runtime errors so the program can keep running.

extra_tables: bool

When handling two types of features if set to true this will generate any extra tables that might be helpful. Note -

These graphics may create duplicates if you already applied an aggregation in 'perform_analysis'

aggregate_target_feature: bool Aggregate the data of the target feature if the data is non-continuous data.

Note In the future I will have this also working with continuous data.

selected_features: collection object of features Will only focus on these selected feature's and will ignore the other given features.

statistical_analysis_on_aggregates: bool If set to true then the function 'statistical_analysis_on_aggregates' will run; which aggregates the data of the target feature either by discrete values or by binning/labeling continuous data.

regression_metrics(*X, y, pred_name, dataset_name, display_visuals=True, save_file=True, title='', custom_metrics_dict={}, ignore_metrics=[], multioutput=[None, 'uniform_average', 'variance_weighted']*)

Creates a dataframe based on the prediction metrics of the feature matrix and target vector.

Args:

X: Feature matrix.

y: Target data vector.

pred_name: The name of the prediction function in questioned stored in 'self.__pred_funcs_dict'

dataset_name: The dataset's name; this will create a sub-directory in which your generated graph will be inner-nested in.

thresholds: If the model outputs a probability list/numpy array then we apply thresholds to the ouput of the model. For classification only; will not affect the direct output of the probabilities.

display_visuals: Display tables.

save_file: Determines whether or not to save the generated document.

title: Adds to the column 'Metric Score'.

custom_metrics_dict: Pass the name of metric(s) and the function definition(s) in a dictionary.

ignore_metrics: Specify the default metrics to not apply to the classification data_analysis.

- Precision
- MCC
- Recall
- F1-Score
- Accuracy

average_scoring:

Determines the type of averaging performed on the data.

- micro
- macro
- weighted

Returns: Return a dataframe object of the metrics value.

CHAPTER

TEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

e

INDEX

O

P

R

S

V