**A person with a mustache

Description automatically generated with medium confidence**

**ZhongShan 中山**

**Package for Exploratory Data Analysis**

**Background**

ZhongShan, the native city of Chinese Revolutionary and Paramount Leader Sun Zhongshan (孙中山), is what this EDA (Exploratory Data Analysis Package) is named after. The purpose of this package is to streamline and reduce redundancy in EDA for Pandas DataFrame.

The naming was particularly chosen as EDA is the first step and foundation of every Data Science project, just as Sun Zhongshan laid the foundations to the Chinese Revolution against feudalism, colonialism and imperialism.

Apart from bulk calculating summary statistics, pairwise statistics between features and labels, and exporting data with only the selected features, it can also perform One Hot Encoding, Standardisation, and PCA transformation. Furthermore, the objects that are fitted in the training process can be collectively stored and exported as a SanMin object (redacted version of ZhongShan without the data), with built in methods allowing for future data (for predicting) to be run through and transformed exactly as the Training, Validation and Test data has been.

As the fundemental output of ZhongShan, SanMin’s name pays a tribute to the “Three Principles of the People” doctrine (三民主义) left behind by President Sun.

It is designed to handle data which has multiple labels (ending up with one set of selected features for each)

**Functions**

|  |  |
| --- | --- |
| Function | Purpose |
| setup\_directory(directories\_to\_create = ['notebooks',  'scripts',  'plots',  'models',  'data',  'data/raw',  'data/curated',  'presentables',  'can\_delete']  ) | Sets up the directory in line with requirements of a Data Science project.  Parameters:  directories\_to\_create – list – with default values |
| setup\_directory(directories\_to\_create) | Sets up directories upon request  Parameters:  directories\_to\_create – str/list |

**Class**

|  |  |
| --- | --- |
| Class | Purpose |
| ZhongShan | Object that stores, manipulates and displays EDA of data; allowing for outputs (plots and data) to be exported to directory  -Summary statistics and boxplots for Full Data  -Corr/NMI statistics, plots, and scatterplots of features with labels for Training Data  Able to perform:  -One Hot Encoding  -Train Val Test split  -PCA  -Standardisation |
| SanMin | Exported-to-file version of ZhongShan object, retaining all transformers that were built in  -does not export EDA data or raw data to prevent overusage of memory |

**Class: ZhongShan**

**Methods:**

|  |  |
| --- | --- |
| Methods | Purpose |
| ZhongShan(full\_data, toggle\_index = True) | Initialisation must input data  Toggle\_index determines whether to re-set all indices |
| fill\_na(df\_name, fill\_value=0) | Fills na values in df\_name dataframe with the fill\_value input  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’  fill\_value – any type – default = int(0) |
| one\_hot\_encode\_fit\_transform(col\_to\_ohe, output\_col\_name) | Fit and OHE transform col\_to\_ohe of data based on full data, and name the on hotted encoded columns by col\_to\_ohe’s items  Parameters:  col\_to\_ohe – str  output\_col\_name – list of strings |
| one\_hot\_encode\_transform(df\_name, col\_to\_ohe) | Transforms df\_name dataframe’s col\_to\_ohe column  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’  col\_to\_ohe – str |
| set\_columns(label\_columns, index\_columns, discarded\_columns) | Manually set which columns are for what purpose  Parameters:  ALL - lists of strings |
| view\_setted\_columns() | Print out the column purposes that are currently set |
| basic\_overview(df\_name, view\_how\_many=10) | Print out the first and last view\_how\_many columns of the selected dataframe, as well as its number of rows and columns  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’  view\_how\_many – int – default 10 |
| get\_full\_data\_analysis() | Compute basic summaries on Full Data (include mean, std, min, q1, q2, q3, max, number of nulls, 1.5 outlier bounds, 3 outlier bounds, and number of outliers using 1.5 and 3 bounds) |
| view\_column\_types() | Print out the column types of the current Full Data |
| get\_boxplot(col) | Gets boxplot of a single column col  Parameters:  col – str |
| view\_full\_data\_analysis() | Print the full Full Data IDE dataframe |
| view\_full\_data\_col\_analysis(col) | For one column col, print out the Full Data Summary  Parameters:  col – str |
| view\_full\_data\_col\_analysis(col) | For all columns, print out full summary |
| read\_in\_train\_test\_split(train\_data, val\_data, test\_data) | Read in Train Test Split data  Parameters:  train\_data – pd.DataFrame  val\_data - pd.DataFrame  test\_data - pd.DataFrame |
| train\_test\_split(train\_percentage, val\_percentage, test\_percentage, random = True, seed = 18661112) | Perform Train Test Split on Full Data that is already in ZhongShan  Parameters:  train\_percentage - float  val\_percentage - float  test\_percentage - float  random - bool  seed - int |
| pca\_fit(n\_components = 10) | Fit PCA using training data with n\_components  Parameters:  n\_components – int – default 10 |
| view\_pca\_explained\_variance\_ratio() | View dataframe that stores PCA explained variance |
| pca\_set\_final\_ncomponents(final\_ncomponents) | Set how many dimensions of PCA we want to use in the end (= final\_ncomponents)  Parameters:  final\_ncomponents - int |
| pca\_transform(df\_name) | PCA transform data using pre-trained PCA object  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’ |
| pca\_transform\_all() | Perform pca\_transform()on all of Full’ or ‘Train’ or ‘Validate’ or ‘Test’ |
| pca\_update\_features() | Update features list after adding PCA features |
| standardise\_fit() | Fit standardiser using Full Data |
| standardise\_transform(df\_name) | Transform df\_name data using pre-fitted standardiser  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’ |
| standardise\_transform\_all() | Standardise all of Full’ or ‘Train’ or ‘Validate’ or ‘Test’ |
| get\_abs\_corr() | Calculate correlation and absolute correlation matrix on Train Data |
| view\_corr\_matrix() | View the correlation matrix |
| view\_abs\_corr\_matrix() | View the absolute correlation matrix |
| get\_corr\_heatmap() | View the absolute correlation heatmap |
| view\_top\_corr() | View the sorted absolute correlation for features on each label |
| get\_nmi() | Calculate NMI matrix based on Train Data |
| view\_nmi\_matrix() | View the NMI matrix |
| get\_nmi\_heatmap() | View the NMI heatmap |
| view\_top\_nmi() | View the sorted NMI for features on each label |
| continuous\_scatter\_plot(col) | Plot scatter plot with feature on x axis and label on y axis for column col  Parameters:  col - str |
| categorical\_scatter\_plot(col) | Plot scatter plot with feature on y axis and label on x axis for column col  Parameters:  col - str |
| feature\_selection(cutoffs) | Read in feature selection cutoff value  Parameters:  cutoffs -float |
| view\_feature\_label\_analysis(col, scatter\_type='continuous') | View feature-label analysis for one feature  Parameters:  col – str  scatter\_type – str – either ‘continuous’ or ‘categorical’ |
| view\_all\_feature\_label\_analysis(scatter\_type = 'continuous') | Print the feature-label analysis for all features  Parameters:  scatter\_type – str – either ‘continuous’ or ‘categorical’ |
| export\_plot(scatter\_type, feature, label, address) | Save designated plot to address  Parameters:  scatter\_type – str - either ‘continuous’ or ‘categorical’  feature – str  label – str  address – str (include .png) |
| get\_selected\_features(metric = 'corr') | Extract lists that contain selected features for each label  Parameters:  metric – str – either ‘corr’ or ‘nmi’ |
| get\_feature\_selected\_data() | Apply the columns to our data for selection |
| export\_data(df\_name, label, address, index = False) | Export df\_name DataFrame for a certain label, to given address, without or with index  Parameters:  df\_name – str – either ‘Full’ or ‘Train’ or ‘Validate’ or ‘Test’  label- str  address – str – (include .csv)  index – bool – default False |

**Objects:**

|  |  |
| --- | --- |
| Objects | Purpose |
| full\_data | DataFrame |
| train\_data | DataFrame |
| val\_data | DataFrame |
| test\_data | DataFrame |
| OHE\_storage | Dictionary  -which stores dicts that contains OHE objects (key ‘OHE\_object’) and output column names (key ’output\_col\_names’)– with key being the column name |
| label\_columns | list |
| index\_columns | list |
| discarded\_columns | list |
| feature\_columns | list |
| retained\_columns | list |
| numeric\_cols | list |
| non\_numeric\_cols | list |
| full\_data\_IDE | DataFrame |
| full\_data\_IDE\_T | DataFrame  Full\_data\_IDE transposed |
| pca | pca object |
| pca\_explained\_variance\_ratio | Numpy array |
| final\_ncomponents | Int |
| standardiser\_objects | Dictionary  -which stores OHE objects – with key being the column name |
| corr\_matrix | DataFrame |
| abs\_corr\_matrix | DataFrame |
| corr\_heatmap | Matplotlib object |
| NMI\_matrix | DataFrame |
| nmi\_heatmap | Matplotlib object |
| cont\_scatter\_plot | Dictionary  -containg matplotlib objects with keys being ‘{col}:{label}’ |
| cat\_scatter\_plot | Dictionary  -containg matplotlib objects with keys being ‘{col}:{label}’ |
| final\_features | Dictionary  -of lists containing selected columns, with key being labels |
| feature\_selected\_full\_data | DataFrame |
| feature\_selected\_train\_data | DataFrame |
| feature\_selected\_val\_data | DataFrame |
| feature\_selected\_test\_data | DataFrame |

**Class: SanMin**

**Methods**

|  |  |
| --- | --- |
| Methods | Purpose |
| SanMin(zhongshan) | Initialisation must input a ZhongShan object  Parameters:  zhongshan - ZhongShan |
| import\_future\_data(future\_data, toggle\_index = True) | Reads in Future Data for transformation  Parameters:  future\_data – pd.DataFrame  toggle\_index – bool – default True |
| export\_data(label, address, index=False) | Export the manipulated future data to given address; with index denoting whether there should be an index  Parameters:  label – str  address – str (include .csv)  index – bool – default False |
| standardise\_transform() | Perform standardisation using pre-fitted standardisers for Future Data |
| fill\_na(fill\_value=0) | Perform na\_filling for Future Data using fill\_value  Parameters:  fill\_value – int – default 0 |
| one\_hot\_encode\_transform(col\_to\_ohe) | OHE transform one column of data using pre-trained OHE objects  Parameters:  col\_to\_ohe - str |
| pca\_transform() | PCA transform data using pre-trained PCA objects |
| get\_feature\_selected\_data() | Apply the pre-selected columns to Future Data |
| export\_SanMin(address) | Export SanMin object to address  Parameters:  address – str |

**Objects**

|  |  |
| --- | --- |
| Objects | Purpose |
| OHE\_storage | Dictionary  -which stores dicts that contains OHE objects (key ‘OHE\_object’) and output column names (key ’output\_col\_names’)– with key being the column name |
| pca | pca object |
| final\_ncomponents | int |
| standardiser\_objects | Dictionary  -which stores OHE objects – with key being the column name |
| final\_features | Dictionary  -of lists containing selected columns, with key being labels |
| retained\_columns | list |

**Dependencies**

pandas

numpy

matplotlib

seaborn

os

copy

pickle

itertools

sklearn