API

The present API describes a set of functions that will assist you in the development of the predictive model. The use of these functions will save you a lot of time!

The functions are divided into several categories:

* [Data Loading and Data Splitting Functions](#_52cc46ixwm4u)
* [Visualization Functions](#_rdkt6vfc192d)
* [Feature Engineering (FE) Functions](#_l3obzfphn9c)
* [Feature Scaling](#_75lr4jj5gd1p)
* [Training Functions](#_7q2r8cozdiv0)
* [Evaluation Functions](#_qnlc8kyzmdvr)

# Data Loading and Data Splitting Functions

## data\_read\_df()

The function returns the task's dataset

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
|  |  |  |

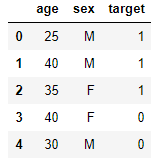
Returns:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | The task dataset |

Example:

df = read\_df()

df.head()



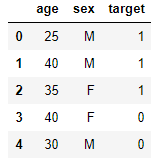
## data\_split\_train\_and\_test(df, test\_size)

This function receives DataFrame and test\_size and splits the DataFrame into train and test.

Note: This is carried out via stratified splitting to ensure that the target’s proportion is the same in both the train and test sets.

Example:

df =



X\_train, X\_test, y\_train, y\_test = data\_split\_train\_and\_test(df,0.3)

|  |  |
| --- | --- |
| X\_train |  |
| X\_test |  |
| y\_train |  |
| y\_test |  |

Please note: the indices are preserved from the DataFrame which is received as input (for error analysis tasks)

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame | DataFrame to split |
| test\_size | Float | It should be between 0.0 and 1.0 and represent the proportion of the dataset that should be included in the test split. |

Returns:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | Train’s examples (X\_train) |
| DataFrame | Test’s examples (X\_test) |
| Series | Train’s target (y\_train) |
| Series | Test’s target (y\_test) |

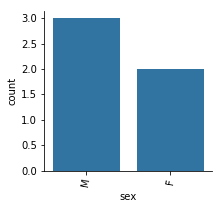
# Visualization Functions

## visual\_generate\_bar\_chart(df, feature\_name)

This function receives the DataFrame and feature name of a categorical variable and returns a plot that describes the count of each value.

Example:

visual\_generate\_count\_plot(df,'sex')



In this example:

There are 3 Males and 2 Females

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame | DataFrame to split |
| feature\_name | str | The name of the column a plot should be generated for |

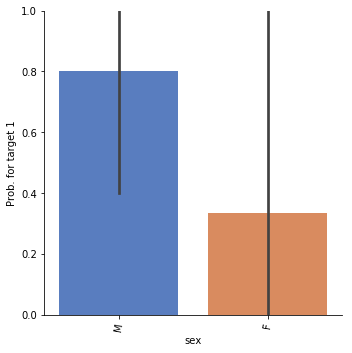
Returns:

A graph will be displayed

## visual\_generate\_category\_target\_prob\_plot(df, feature\_name)

This function receives the DataFrame and feature name of a categorical variable and returns a plot that describes the probability for target 1 for each value.

Example:



In this example, the target is 1 for 80% of the males

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame | DataFrame to split |
| feature\_name | str | The name of the column a plot should be generated for |

Returns:

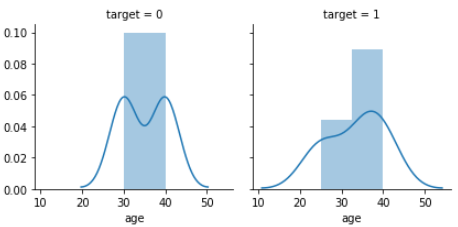
A graph will be displayed

## visual\_generate\_dis\_plot(df, feature\_name)

Plot a univariate distribution of observations as a function of the target (for numeric variables).

Example:

visual\_generate\_dis\_plot(df, 'age')



Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame | DataFrame to split |
| feature\_name | str | The name of the column a plot should be generated for |

Returns:

A graph will be displayed

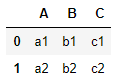
# Feature Engineering (FE) Functions

## FE\_encode\_values\_of\_categorial\_features(df, columns\_to\_encode)

This function receives a DataFrame and the names of columns whose values should be encoded into numbers.

Example:

df =



FE\_encode\_values\_of\_categorial\_features(df, ['A','B'])



Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame |  |
| columns\_to\_encode | list | The names of the columns that are to be encoded |

Return:

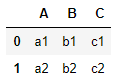
|  |  |
| --- | --- |
| Value | Description |
| DataFrame | The DataFrame produced after encoding the selected columns |

## FE\_create\_one\_hot\_encodeing(df, columns\_to\_encode)

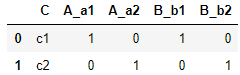
This function receives a DataFrame and the names of the columns whose values should be encoded using one-hot encoding.

Example:

df =



FE\_create\_one\_hot\_encodeing(df, ['A','B'])



Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame |  |
| columns\_to\_encode | list | The names of columns that are to be encoded |

Return:

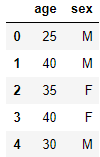
|  |  |
| --- | --- |
| Value | Description |
| DataFrame | The DataFrame produced after one-hot encoding the selected columns |

## FE\_divide\_numeric\_feature\_to\_ranges(df, column\_to\_divide\_to\_ranges, number\_of\_ranges)

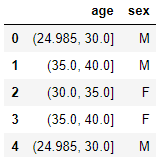
This function receives a DataFrame, a column name, and a number of ranges, and divides the selected columns into ranges.

Example:

df =



FE\_divide\_numeric\_feature\_to\_ranges(df\_temp, ‘age’,3)



Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| df | DataFrame |  |
| column\_to\_devide\_to\_ranges | str | The name of the column that is to be divided into ranges |

Returns:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | The DataFrame produced after dividing the selected column’s values |

# Feature Scaling Functions

## feature\_scaling(X\_train, X\_test, features\_to\_scale, scaling\_method)

This function receives X\_train, X\_test, columns to scale, and the scaling method (Normalizer / StandardScaler / MinMaxScaler), and returns a DataFrame after scaling the selected columns.

Example:

|  |  |
| --- | --- |
| X\_train |  |
| X\_test |  |

X\_train\_new, X\_test\_new = feature\_scaling(X\_train, X\_test, ['age'], 'StandardScaler')

|  |  |
| --- | --- |
| X\_train\_new |  |
| X\_test\_new |  |

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| X\_train | Train examples | The scaling is carried out in accordance with these examples’ distribution (and transformed on these examples too) |
| X\_test | Test examples | The values of these examples will be scaled according to the X\_train distribution |
| features\_to\_scale | list | Column names (the columns which should be scaled) |
| scaler\_method | str | The scaling will be carried out according to the given scaler\_methood.  The Scaler method should be one of the following:   * ‘Normalizer’ * ‘StandardScaler’ * ‘MinMaxScaler’   See the following links for exploring these methods:   * [Normalizer](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.Normalizer.html) * [StandardScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html) * [MinMaxScaler](https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html) |

Returns:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | The DataFrame produced after dividing the selected column’s values |

# 

# Training Functions

## train\_model(ml\_algo, df\_train, y\_train, params=None)

The function receives the name of a machine learning algorithm, the parameters for this algorithm (if not provided, it will use the default Sklearn parameters), and the train’s examples and targets as inputs and returns a trained classifier (after fitting it to train).

Inputs:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| ml\_algo | str | * 'LogisticRegression’ * 'DecisionTree' * 'KNN' * 'RandomForest' * 'LDA' * ‘MultinomialNB’ * ‘GaussianNB’ |
| df\_train | DataFrame | Train’s DataFrame |
| y\_train | Series | Train’s target |
| Params (optional) | dictionary | Description of the classifier parameter values (e.g., params[*‘penalty’*] =’l2’, params[‘*C*’]=1 when using LogisticRegression).  Please note: not all parameters are required (since, for example,. the default Sklearn value will be used for each parameter that is not passed).  See the following links in order to explore the optional parameters:   * [LogisticRegression](https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html)   [DecisionTree](https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html)   * [KNN](https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html) * [RandomForest](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html) * [LDA](https://scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.LinearDiscriminantAnalysis.html) * [MultinomialNB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.MultinomialNB.html) * [GaussianNB](https://scikit-learn.org/stable/modules/generated/sklearn.naive_bayes.GaussianNB.html) |

Returns:

|  |  |
| --- | --- |
| Value | Description |
| Sklearn classifier | Classifier after fitting to train’s examples |

# Evaluation Functions

## eval\_get\_score(classifier, X, y, metric)

Receives a trained classifier, examples and targets and returns a score (precision / recall / accuracy / auc / F1)

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| classifier | Sklearn classifier | A classifier after it has been fitted to train examples |
| X | DataFrame | Examples for calculating the chosen set (e.g. test set or train set) |
| y | Series | Target variable (e.g. y\_test or y\_train) |
| metric | str | Metric to return. It should be one of the following:   * ‘precision’ * ‘recall’ * ‘accuracy’ * ‘auc’ * f1 |

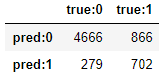
Returns:

|  |  |
| --- | --- |
| Value | Description |
| Float | The resulting score |

## eval\_get\_cm(classifier, X, y)

Receives a trained classifier, examples and targets. Returns the Confusion Matrix.

Example output:



Inputs:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| classifier | Sklearn classifier | A classifier after it has been fitted to train examples |
| X | DataFrame | Examples for calculating AUC (e.g. test set or train set) |
| y | Series | Target variable (e.g. y\_test or y\_train) |

Returns:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | Confusion Matrix |

## eval\_plot\_roc\_curve(classifier, X, y)

Plot ROC curve

Inputs:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| classifier | Sklearn classifier | A classifier after it has been fitted to train examples |
| X | DataFrame | Examples for calculating AUC (e.g. test set or train set) |
| y | Series | Target variable (e.g. y\_test or y\_train) |

Returns:

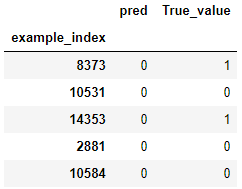
ROC curve plot

## eval\_get\_predictions(classifier, X, y)

This function receives an Sklearn classifier, examples for predicting (X) and the y\_true (y) of these examples. It returns a Dataframe with the predictions and y\_true.

Sample Output:

get\_predictions(classifier, X\_test, y\_test)



Please note: the example\_index is the original index of X (for error analysis tasks)

Inputs:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| classifier | Sklearn classifier | A classifier after it has been fitted to train examples |
| X | DataFrame | Examples to predict (e.g. X\_test or X\_train) |
| y | Series | Target variable (e.g. y\_test or y\_train) |

Returns:

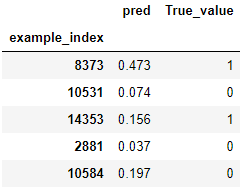
|  |  |
| --- | --- |
| Value | Description |
| DataFrame | DataFrame with the predictions. |

## eval\_get\_predictions\_proba(classifier, X, y)

The function receives an Sklearn classifier, examples for predicting (X) and the y\_true (y) of these examples. It returns a Dataframe with the probability predictions (probability for target 1) and y\_true.

Example Output:

get\_predictions\_proba(classifier, X\_test, y\_test)



Please note: the example\_index is the original index of X (for error analysis tasks)

Input:

|  |  |  |
| --- | --- | --- |
| Parameter | Value | Description |
| classifier | Sklearn classifier | A classifier after it has been fitted to train examples |
| X | DataFrame | Examples to predict (e.g. X\_test or X\_train) |
| y | Series | Target variable (e.g. y\_test or y\_train) |

Return:

|  |  |
| --- | --- |
| Value | Description |
| DataFrame | DataFrame with the predictions. |

# 