Simple Machine Learning Tool Kit

Release 2.2.7

Alessandra Bilardi

Mar 25, 2023

CONTENTS:

This package contains the classes to manage data preprocessing, testing and reporting.

CHAPTER

ONE

GETTING STARTED

smltk (Simple Machine Learning Tool Kit) package is implemented for helping your work during

- data preparation
- testing your model

The goal is to implement this package for each step of machine learning process that can simplify your code.

It is part of the educational repositories to learn how to write stardard code and common uses of the TDD.

1.1 Installation

If you want to use this package into your code, you can install by python3-pip:

```
pip3 install smltk
python3
>>> from smltk.metrics import Metrics
>>> help(Metrics)
```

The package is not self-consistent. So if you want to contribute, you have to download the package by github and to install the requirements

```
git clone https://github.com/bilardi/smltk
cd smltk/
pip3 install --upgrade -r requirements.txt
```

Read the documentation on readthedocs for

- API
- Usage
- Development

1.2 Change Log

See CHANGELOG.md for details.

1.3 License

This package is released under the MIT license. See LICENSE for details.

CHAPTER

TWO

API

2.1 Metrics

The class for testing models and reporting results

A collection of methods to simplify your code.

smltk.metrics.Metrics	

2.1.1 Targets and Predictions management

smltk.metrics.Metrics.split_tuples	Splits tuples of sample and its target in the relative lists
smltk.metrics.Metrics.prediction	Predicts with your model

2.1.2 Metrics management

smltk.metrics.Metrics.create_confusion_matrix	Creates and prints confusion matrix
smltk.metrics.Metrics.get_classification_metrics	Gets classification metrics
smltk.metrics.Metrics.scoring	Gets classification metrics after prediction
smltk.metrics.Metrics.modeling	Gets classification metrics after training and prediction
smltk.metrics.Metrics.print_metrics	Prints metrics

2.1.3 Model management

smltk.metrics.Metrics.save_model	Saves model
smltk.metrics.Metrics.resume_model	Resumes model

2.1.4 Detailed list

class smltk.metrics.**Metrics**

clean_binary_classification(*y_test*, *y_pred*)

Transforms the target and prediction in integer 0 and 1

Arguments:

y_test (list[])
list of targets

y_pred (list[])
list of predictions

Returns:
y_test and y_pred with only 0 and 1 values

create_confusion_matrix(*y_test*, *y_pred*, *is_test=False*)

Creates and prints confusion matrix

Arguments:

y_test (list[])
list of targets

y_pred (list[])
list of predictions

is_test (bool)
default is False

Returns:
confusion matrix

fit_exists(*model*)

Get a boolean True if fit method exists

Arguments:

model (obj)
object of your model

Returns:
boolean

get_classification_metrics(*params={}*)

Gets classification metrics

Arguments: params (dict) with the keys below

model (obj)
object of your model

X_train (list[]|list[tuple])
list of samples or list of tuples with sample and its target

y_train (list[])
list of targets

X_test (list[] | list[tuple])
list of samples or list of tuples with sample and its target

y_test (list[])
list of targets

y_pred (list[])
list of predictions

loss (str)
parameter of bias_variance_decomp, default mse

num_rounds (int)
parameter of bias_variance_decomp, default 200

random_seed (int)
parameter of bias_variance_decomp, default 3

Returns:
dictionary with Loss, Bias, Variance, MCC, ROC_AUC, Accuracy, Precision, Recall, Fscore

`is_binary_classification`(*y_test*, *y_pred*)

Gets if the classification is binary or not

Arguments:

y_test (list[])
list of targets

y_pred (list[])
list of predictions

Returns:
boolean

`modeling`(*model*, *X_train*, *y_train*, *X_test*, *y_test*)

Gets classification metrics after training and prediction

Arguments:

model (obj)
object of your model

X_train (list[]|list[tuple])
list of samples or list of tuples with sample and its target

y_train (list[])
list of targets

X_test (list[] | list[tuple])
list of samples or list of tuples with sample and its target

y_test (list[])
list of targets

Returns:
dictionary with Loss, Bias, Variance, MCC, ROC_AUC, Accuracy, Precision, Recall, Fscore

`predict_exists`(*model*)

Get a boolean True if predict method exists

Arguments:

model (obj)
object of your model

Returns:
boolean

`prediction`(*model*, *method*, *X_test*, *y_test=[]*)

Predicts with your model

Arguments:

model (obj)
object of your model

method (str)
name of method

X_test (list[]|list[tuple])
list of samples or list of tuples with sample and its target

y_test (list[])
list of targets

Returns:
tuple of list of targets and list of predictions

`print_metrics`(*metrics*)

Prints metrics

Arguments:

metrics (dict)
dictionary of metrics with their value

Returns:
only the print of metrics

`resume_model`(*filename*)

Resumes model

Arguments:

filename (str)
pathname and filename where you want to save your model

Returns:
object of your model

`save_model`(*model*, *filename*)

Saves model

Arguments:

model (obj)
object of your model

filename (str)
pathname and filename where you want to save your model

`scoring`(*model*, *X_test*, *y_test*)

Gets classification metrics after prediction

Arguments:

model (obj)
object of your model

X_test (list[] | list[tuple])
list of samples or list of tuples with sample and its target

y_test (list[])
list of targets

Returns:
dictionary with Loss, Bias, Variance, MCC, ROC_AUC, Accuracy, Precision, Recall, Fscore

`split_tuples`(*tuples*)

Splits tuples of sample and its target in the relative lists

Arguments:

tuples (list[tuple])
list of tuples with sample and its target

Returns:
tuple of list of samples and list of targets

2.2 PreProcessing

The classes for data preparation with nltk, sklearn and more

A collection of methods to simplify your code.

smltk.preprocessing.Ntk	The class Ntk contains the Natural Language Processing tool kit.

2.2.1 Natural Language Tool Kit

smltk.preprocessing.Ntk.get_tokens_cleaned	Tokenizes doc and filters tokens from punctuation, numbers, stop words, and words <= min_length and changes upper characters in lower characters and if is_lemma == True, also it lemmatizes
smltk.preprocessing.Ntk.get_doc_cleaned	Filters doc from punctuation, numbers, stop words, and words <= min_length and changes upper characters in lower characters and if is_lemma == True, also it lemmatizes
smltk.preprocessing.Ntk.get_stats_vocab	Gets statistics of vocabulary
smltk.preprocessing.Ntk.get_words_top	Gets words top for each target
smltk.preprocessing.Ntk.get_vocabs_cleaned	Cleans vocabs from common words among targets
smltk.preprocessing.Ntk.get_ngrams	Gets ngrams from doc or tokens
smltk.preprocessing.Ntk.get_ngrams_features	Gets ngrams features from doc or tokens
smltk.preprocessing.Ntk.get_features	Gets features
smltk.preprocessing.Ntk.get_features_from_docs	Gets features
smltk.preprocessing.Ntk.create_tuples	Creates tuples with sample and its target
smltk.preprocessing.Ntk.create_vocab_from_docs	Creates vocabulary from list of docs
smltk.preprocessing.Ntk.create_vocab_from_tuples	Creates vocabulary from list of tuples
smltk.preprocessing.Ntk.create_features_from_docs	Creates features from docs
smltk.preprocessing.Ntk.create_features_from_tuples	Creates features from tuples
smltk.preprocessing.Ntk.create_ngrams_features_from_docs	Creates ngrams features from docs
smltk.preprocessing.Ntk.create_ngrams_features_from_tuples	Creates ngrams features from tuples
smltk.preprocessing.Ntk.create_words_map	Creates the map of words
smltk.preprocessing.Ntk.create_words_cloud	Creates the cloud of words
smltk.preprocessing.Ntk.vectorize_docs	Vectorizes docs

2.2.2 Detailed list

class smltk.preprocessing.**Ntk**(*params=[]*)

The class Ntk contains the Natural Language Processing tool kit.

Arguments: params (dict) with the keys below

language (str)
default is english

lemmatizer (obj)
obj with method lemmatize() like WordNetLemmatizer()

min_length (int)
default is 1, so all words will be used

stop_words (list[str])
default is stopwords.words()

tag_map (dict)
default contains J, V, R

Here's an example:

```
>>> from smltk.preprocessing import Ntk
>>> doc = 'Good case, Excellent value.'
>>> ntk = Ntk()
>>> get_doc_cleaned = ntk.get_doc_cleaned(doc)
>>> print(get_doc_cleaned)
good case excellent value
```

add_doc_to_vocab(*doc*, *vocab*, *is_lemma=True*)

Adds tokens of that doc to vocabulary and updates vocabulary

Arguments:

doc (str)
text

vocab (collections.Counter)
dictionary of tokens with its count

is_lemma (bool)
default is True

Returns:
list of tokens of that doc and vocab updated

create_features_from_docs(*docs*, *target*, *is_lemma=True*, *words_top={}*, *degree=0*)

Creates features from docs

Arguments:

docs (list[str])
list of text

target (str)
target name of the docs

is_lemma (bool)
default is True

words_top (dict)
dictionary of the words top

degree (int)
degree of ngrams, default is 0

Returns:
list of tuples with features and relative target

create_features_from_tuples(*tuples*, *is_lemma=True*, *words_top={}*, *degree=0*)

Creates features from tuples

Arguments:

tuples (list[tuples])
list of tuples with sample and its target

is_lemma (bool)
default is True

words_top (dict)
dictionary of the words top

degree (int)
degree of ngrams, default is 0

Returns:
list of tuples with features and relative target

`create_ngrams_features_from_docs`(*docs*, *target*, *is_lemma=True*, *degree=2*)

Creates ngrams features from docs

Arguments:

docs (list[str])
list of text

target (str)
target name of the docs

is_lemma (bool)
default is True

degree (int)
degree of ngrams, default is 2

Returns:
list of tuples with features and relative target

`create_ngrams_features_from_tuples`(*tuples*, *is_lemma=True*, *degree=2*)

Creates ngrams features from tuples

Arguments:

tuples (list[tuples])
list of tuples with sample and its target

is_lemma (bool)
default is True

degree (int)
degree of ngrams, default is 2

Returns:
list of tuples with features and relative target

`create_tuples`(*docs=[]*, *target=[]*)

Creates tuples with sample and its target

Arguments:

docs (list[str])
list of texts

target (list[str])
list of targets

Returns:
list of tuples with sample and its target

`create_vocab_from_docs`(*docs*, *is_lemma=True*)

Creates vocabulary from list of docs

Arguments:

docs (list[str])
list of texts

is_lemma (bool)
default is True

Returns:
dictionary of tokens with its count in an object collections.Counter

`create_vocab_from_tuples`(*tuples*, *is_lemma=True*)

Creates vocabulary from list of tuples

Arguments:

tuples (list[tuples])
list of tuples with sample and its target

is_lemma (bool)
default is True

Returns:
dictionary of tokens with its count in an object collections.Counter

`create_words_cloud`(*words*, *is_test=False*)

Creates the cloud of words

Arguments:

words (str)
words

is_test (bool)
default is False

Returns:
only words cloud plot

`create_words_map`(*words*)

Creates the map of words

Arguments:

words (list[str])
words list

Returns:
string of all words

`get_doc_cleaned`(*doc*, *is_lemma=True*)

Filters doc from punctuation, numbers, stop words, and words <= min_length and changes upper characters in lower characters and if is_lemma == True, also it lemmatizes

Arguments:

doc (str)
text

is_lemma (bool)
default is True

Returns:
string cleaned

`get_features`(*doc*, *is_lemma=True*, *words_top={}*, *degree=0*)

Gets features

Arguments:

doc (str)
text

is_lemma (bool)
default is True

words_top (dict)
dictionary of the words top

degree (int)
degree of ngrams, default is 0

Returns:
dictionary of features extracted

`get_features_from_docs`(*docs*, *is_lemma=True*, *words_top={}*, *degree=0*)

Gets features

Arguments:

docs (list[str])
list of text

is_lemma (bool)
default is True

words_top (dict)
dictionary of the words top

degree (int)
degree of ngrams, default is 0

Returns:
dictionary of features extracted

`get_ngrams`(*degree=2*, *doc=''*, *tokens=[]*, *is_tuple=True*, *is_lemma=False*)

Gets ngrams from doc or tokens

Arguments:

degree (int)
degree of ngrams, default is 2

doc (str)
text, option if you pass tokens

tokens (list[str])
list of tokens, option if you pass doc

is_tuple (bool)
default is True

is_lemma (bool)
default is False

Returns:
list of tuples (n_grams) for that degree, or list of string (token)

`get_ngrams_features`(*degree=2*, *doc=''*, *tokens=[]*, *is_lemma=False*)

Gets ngrams features from doc or tokens

Arguments:

degree (int)
degree of ngrams, default is 2

doc (str)
text, option if you pass tokens

tokens (list[str])
list of tokens, option if you pass doc

is_lemma (bool)
default is False

Returns:
dictionary of ngrams extracted

`get_stats_vocab`(*vocab*, *min_occurance=1*)

Gets statistics of vocabulary

Arguments:

vocab (collections.Counter)
dictionary of tokens with its count

min_occurance (int)
minimum occurance considered

Returns:
tuple of tokens number with >= min_occurance and total tokens number

`get_tokens_cleaned`(*doc*, *is_lemma=True*)

Tokenizes doc and filters tokens from punctuation, numbers, stop words, and words <= min_length and changes upper characters in lower characters and if is_lemma == True, also it lemmatizes

Arguments:

doc (str)
text

is_lemma (bool)
default is True

Returns:
list of tokens cleaned

`get_vocabs_cleaned`(*vocabs*)

Cleans vocabs from common words among targets

Arguments:

vocabs (dict)
keys are targets, values are vocabularies for that target

Returns:
vocabs cleaned from common words among targets

`get_words_top`(*vocab*, *how_many*)

Gets words top for each target

Arguments:

vocab (collections.Counter)
dictionary of tokens with its count

how_many (int)
how many words in your top how_many list

Returns:
dictionary of the top how_many list

`lemmatize`(*tokens*)

Lemmatizes tokens

Arguments:

tokens (list[str])
list of words

Returns:
list of tokens lemmatized

`tokenize_and_clean_doc`(*doc*)

Tokenizes doc and filters tokens from punctuation, numbers, stop words, and words <= min_length and changes upper characters in lower characters

Arguments:

doc (str)
text

Returns:
list of words filtered

`vectorize_docs`(*docs*, *is_count=True*, *is_lemma=False*, *is_test=False*)

Vectorizes docs

Arguments:

docs (list[str])
list of texts

is_count (bool)
default is True

is_lemma (bool)
default is True

is_test (bool)
default is False

Returns:
list of scipy.sparse.csr.csr_matrix, one for each doc

`word_tokenize`(*doc*)

Splits document in each word

Arguments:

doc (str)
text

Returns:
list of words

2.3 DataVisualization

The class for managing the data of the main repositories

A collection of methods to simplify your code.

smltk.datavisualization.DataVisualization	

2.3.1 Features management

smltk.datavisualization.DataVisualization.get_df	Create a DataFrame from the data of the main repositories
smltk.datavisualization.DataVisualization.get_inference_df	Create a DataFrame from the data of the main repositories

2.3.2 Images management

smltk.datavisualization.DataVisualization.get_inference_objects	Rescale boxes with probability greater than the threshold
smltk.datavisualization.DataVisualization.get_inference_objects_df	Create a DataFrame from the prediction of object detection
smltk.datavisualization.DataVisualization.plot_inference_objects	Plot image with boxes

2.3.3 Detailed list

class smltk.datavisualization.**DataVisualization**(**args*, ***kwargs*)

bboxes_cxcywh_to_xyxy(*bboxes*)

Concatenate a sequence of tensors along a new dimension

Arguments:

bboxes (sequence of Tensors)
list of boxes Tensors

Returns:
sequence of Tensors

get_df(*data*)

Create a DataFrame from the data of the main repositories

Arguments:

data (mixed)
data loaded from one of the main repositories

Returns:
Pandas DataFrame

`get_inference_df`(*data*, *x_test*, *y_test*, *y_pred*)

Create a DataFrame from the data of the main repositories

Arguments:

x_test (Pandas DataFrame)
features used for the prediction

y_test (list of str)
list of the targets

y_pred (list of str)
list of the predictions

Returns:
Pandas DataFrame

`get_inference_objects`(*image*, *prediction*, *threshold=0.7*)

Rescale boxes with probability greater than the threshold

Arguments:

image (PIL Image)
object of type PIL Image

prediction (dict)
prediction of the model with pred_logits and pred_boxes

threshold (float)
probability value used like threshold, default 0.7

Returns:
tuple of sequences of Tensors about probabilities and boxes

`get_inference_objects_df`(*probability*, *boxes*)

Create a DataFrame from the prediction of object detection

Arguments:

probability (sequence of Tensors)
list of probabilities Tensors

boxes (sequence of Tensors)
list of boxes Tensors

Returns:
Pandas DataFrame

`plot_inference_objects`(*image*, *probability*, *boxes*)

Plot image with boxes

Arguments:

image (PIL Image)
object of type PIL Image

probability (sequence of Tensors)
list of probabilities Tensors

boxes (sequence of Tensors)
list of boxes Tensors

Returns:
plot

rescale_bboxes(*bboxes*, *size*)

Rescale boxes on image size

Arguments:

bboxes (sequence of Tensors)
list of boxes Tensors

size (tuple)
width and height of image

Returns:
sequence of Tensors

CHAPTER

THREE

USAGE

3.1 Preprocessing and Metrics

The package only contains methods to simplify your code, so it doesn't cover all steps of machine learning process, but only those that save you lines of code.

[1]:
```
#!pip install smltk==2.2.2

!pip install nltk
!pip install wordcloud
%pip install mlxtend --upgrade

# import os
# import sys
# sys.path.insert(1, os.getcwd() + '/../../../')
# print(os.getcwd())
# print(sys.path)

import numpy as np
from smltk.preprocessing import Ntk
nlp = Ntk()
from smltk.metrics import Metrics
mtr = Metrics()
```

```
Requirement already satisfied: nltk in /Users/alessandrabilardi/.env/lib/python3.10/site-
↪packages (3.8.1)
Requirement already satisfied: click in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (from nltk) (8.1.3)
Requirement already satisfied: regex>=2021.8.3 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from nltk) (2022.9.13)
Requirement already satisfied: tqdm in /Users/alessandrabilardi/.env/lib/python3.10/site-
↪packages (from nltk) (4.64.0)
Requirement already satisfied: joblib in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (from nltk) (1.2.0)

[notice] A new release of pip available: 22.2.2 -> 23.0.1
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: wordcloud in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (1.8.2.2)
Requirement already satisfied: numpy>=1.6.1 in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from wordcloud) (1.23.5)
```

(continues on next page)

(continued from previous page)

```
Requirement already satisfied: matplotlib in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from wordcloud) (3.7.1)
Requirement already satisfied: pillow in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (from wordcloud) (9.4.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (1.4.4)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (3.0.9)
Requirement already satisfied: packaging>=20.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (21.3)
Requirement already satisfied: fonttools>=4.22.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (4.37.3)
Requirement already satisfied: python-dateutil>=2.7 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (2.8.2)
Requirement already satisfied: contourpy>=1.0.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib->wordcloud) (1.0.5)
Requirement already satisfied: cycler>=0.10 in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from matplotlib->wordcloud) (0.11.0)
Requirement already satisfied: six>=1.5 in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (from python-dateutil>=2.7->matplotlib->wordcloud) (1.16.0)

[notice] A new release of pip available: 22.2.2 -> 23.0.1
[notice] To update, run: pip install --upgrade pip
Requirement already satisfied: mlxtend in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (0.21.0)
Requirement already satisfied: pandas>=0.24.2 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from mlxtend) (1.5.3)
Requirement already satisfied: matplotlib>=3.0.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from mlxtend) (3.7.1)
Requirement already satisfied: numpy>=1.16.2 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from mlxtend) (1.23.5)
Requirement already satisfied: scipy>=1.2.1 in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from mlxtend) (1.9.1)
Requirement already satisfied: setuptools in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from mlxtend) (65.4.1)
Requirement already satisfied: joblib>=0.13.2 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from mlxtend) (1.2.0)
Requirement already satisfied: scikit-learn>=1.0.2 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from mlxtend) (1.1.0)
Requirement already satisfied: contourpy>=1.0.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (1.0.5)
Requirement already satisfied: kiwisolver>=1.0.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (1.4.4)
Requirement already satisfied: packaging>=20.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (21.3)
Requirement already satisfied: pyparsing>=2.3.1 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (3.0.9)
Requirement already satisfied: fonttools>=4.22.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (4.37.3)
Requirement already satisfied: cycler>=0.10 in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from matplotlib>=3.0.0->mlxtend) (0.11.0)
Requirement already satisfied: python-dateutil>=2.7 in /Users/alessandrabilardi/.env/lib/
```

(continues on next page)

(continued from previous page)

```
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from matplotlib>=3.0.0->mlxtend) (9.4.0)
Requirement already satisfied: pytz>=2020.1 in /Users/alessandrabilardi/.env/lib/python3.
↪10/site-packages (from pandas>=0.24.2->mlxtend) (2022.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /Users/alessandrabilardi/.env/lib/
↪python3.10/site-packages (from scikit-learn>=1.0.2->mlxtend) (3.1.0)
Requirement already satisfied: six>=1.5 in /Users/alessandrabilardi/.env/lib/python3.10/
↪site-packages (from python-dateutil>=2.7->matplotlib>=3.0.0->mlxtend) (1.16.0)

[notice] A new release of pip available: 22.2.2 -> 23.0.1
[notice] To update, run: pip install --upgrade pip
Note: you may need to restart the kernel to use updated packages.
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package omw-1.4 to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package omw-1.4 is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package vader_lexicon to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package vader_lexicon is already up-to-date!
```

3.1.1 Example with nltk library

[2]:
```
import nltk
nltk.download('movie_reviews')
```

```
[nltk_data] Downloading package movie_reviews to
[nltk_data]     /Users/alessandrabilardi/nltk_data...
[nltk_data]   Package movie_reviews is already up-to-date!
```

[2]:
```
True
```

[3]:
```
docs_pos = [nltk.corpus.movie_reviews.raw(review) for review in nltk.corpus.movie_
↪reviews.fileids(categories=["pos"])]
docs_neg = [nltk.corpus.movie_reviews.raw(review) for review in nltk.corpus.movie_
↪reviews.fileids(categories=["neg"])]
```

Preprocessing

[4]:
```
# data reduction
vocab_pos = nlp.create_vocab_from_docs(docs_pos)
#vocab_pos = nlp.create_vocab_from_docs(nltk.Text(nltk.corpus.movie_reviews.
↪words(categories=["pos"])))
vocab_neg = nlp.create_vocab_from_docs(docs_neg)
#vocab_neg = nlp.create_vocab_from_docs(nltk.Text(nltk.corpus.movie_reviews.
↪words(categories=["neg"])))
vocabs = {'pos': vocab_pos, 'neg': vocab_neg}
```

[5]:
```
# distribution
for target in vocabs.keys():
  print(target)
  print(nlp.get_stats_vocab(vocabs[target], 2))
  nlp.create_words_cloud(nlp.create_words_map(vocabs[target]))
```

```
pos
(15791, 27546)
```


```
neg
(14665, 25738)
```

[6]:
```
# data cleaning
vocabs_cleaned = nlp.get_vocabs_cleaned(vocabs)
```

[7]:
```
# data wrangling
top_100_pos = nlp.get_words_top(vocabs_cleaned['pos'], 100)
top_100_neg = nlp.get_words_top(vocabs_cleaned['neg'], 100)

features = nlp.create_features_from_docs(docs_pos, 'pos', words_top = top_100_pos)
features.extend(nlp.create_features_from_docs(docs_neg, 'neg', words_top = top_100_neg))
```

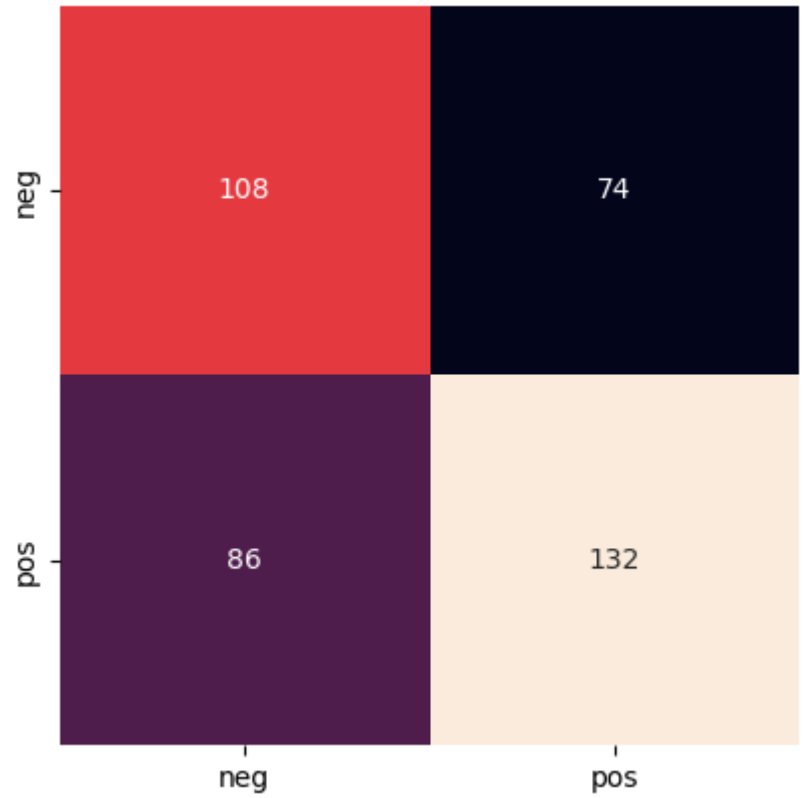
[8]:
```
# splitting
from nltk.sentiment.util import split_train_test
X_train, X_test = split_train_test(features)
```

Modeling

[9]:
```
# fitting
classifier = nltk.NaiveBayesClassifier.train(X_train)
```

[10]:
```
# testing
y_test, y_pred = mtr.prediction(classifier, 'classify', X_test)
mtr.create_confusion_matrix(y_test, y_pred)
```

[10]:
```
array([[108,  74],
       [ 86, 132]])
```



[11]:
```
mtr.get_classification_metrics({
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[11]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.19819315931825934,
 'ROC_AUC': 0.5994555902812784,
 'Accuracy': 0.6,
 'Precision': array([0.55670103, 0.6407767 ]),
 'Recall': array([0.59340659, 0.60550459]),
 'Fscore': array([0.57446809, 0.62264151]),
 'Support': array([182, 218])}
```

Prediction

```
[12]: classifier.classify(nlp.get_features(docs_pos[0]))
```

```
[12]: 'pos'
```

```
[13]: classifier.classify(nlp.get_features(docs_neg[0]))
```

```
[13]: 'neg'
```

3.1.2 Example with sklearn library - wine dataset

```
[14]: from sklearn.datasets import load_wine
      data = load_wine()
```

Preprocessing

```
[15]: # splitting
      from sklearn.model_selection import train_test_split
      X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.
      ↪2, random_state=5)
```

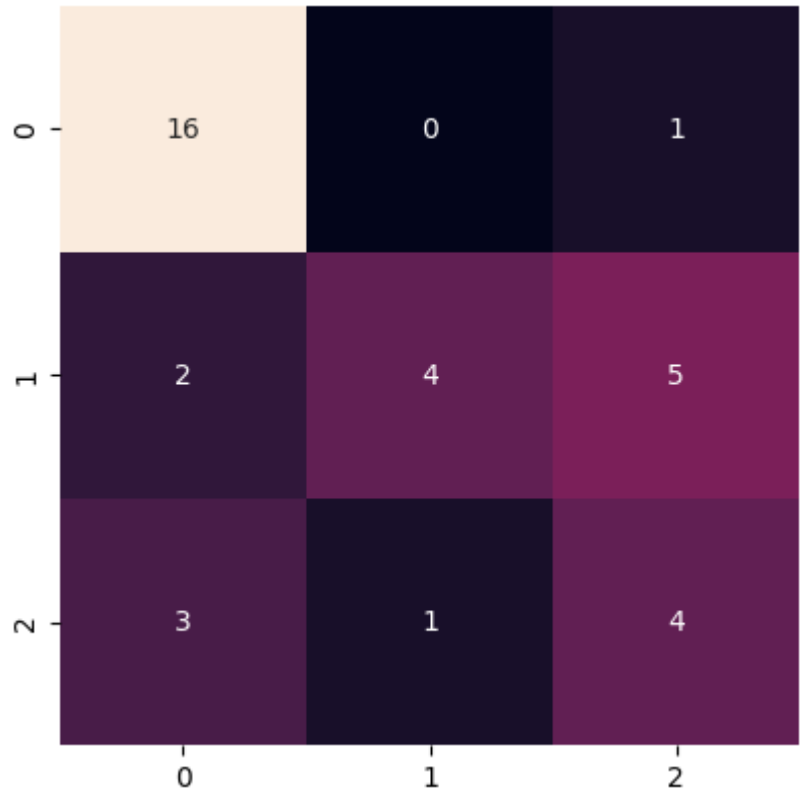
Modeling

```
[16]: # fitting
      from sklearn.linear_model import SGDClassifier
      model = SGDClassifier(random_state=3)
      model.fit(X_train, y_train)
```

```
[16]: SGDClassifier(random_state=3)
```

```
[17]: # testing
      y_pred = model.predict(X_test)
      mtr.create_confusion_matrix(y_test, y_pred)
```

```
[17]: array([[16,  0,  1],
             [ 2,  4,  5],
             [ 3,  1,  4]])
```



[18]:
```
mtr.get_classification_metrics({
    "model": model,
    "X_train": np.array(X_train),
    "y_train": np.array(y_train),
    "X_test": np.array(X_test),
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[18]:
```
{'Loss': 0.7443055555555557,
 'Bias': 0.3698465277777778,
 'Variance': 0.3744590277777778,
 'MCC': 0.4802259242337604,
 'ROC_AUC': 0,
 'Accuracy': 0.6666666666666666,
 'Precision': array([0.76190476, 0.8       , 0.4       ]),
 'Recall': array([0.94117647, 0.36363636, 0.5       ]),
 'Fscore': array([0.84210526, 0.5       , 0.44444444]),
 'Support': array([17, 11,  8])}
```

Management

[19]:
```
mtr.save_model(model, 'model.pkl')
model.score(X_test, y_test)
```

[19]:
```
0.6111111111111112
```

[20]:
```
model_resumed = mtr.resume_model('model.pkl')
model_resumed.score(X_test, y_test)
```

[20]:
```
0.6111111111111112
```

3.1.3 Example with sklearn library - text dataset

[21]:
```
!if [ ! -f 'sentiment.zip' ]; then curl https://archive.ics.uci.edu/ml/machine-learning-
↪databases/00331/sentiment%20labelled%20sentences.zip --output sentiment.zip; fi
!if [ ! -d 'sentiment labelled sentences' ]; then unzip 'sentiment.zip'; fi
!ls -ltr 'sentiment labelled sentences'
```

```
total 416
-rw-r--r--  1 alessandrabilardi  staff  85285 Feb 15  2015 imdb_labelled.txt
-rw-r--r--  1 alessandrabilardi  staff   1070 May 31  2015 readme.txt
-rw-r--r--  1 alessandrabilardi  staff  61320 Jul  5  2016 yelp_labelled.txt
-rw-r--r--  1 alessandrabilardi  staff  58226 Jul  5  2016 amazon_cells_labelled.txt
```

[22]:
```
!cat 'sentiment labelled sentences/amazon_cells_labelled.txt' | head -n 5
```

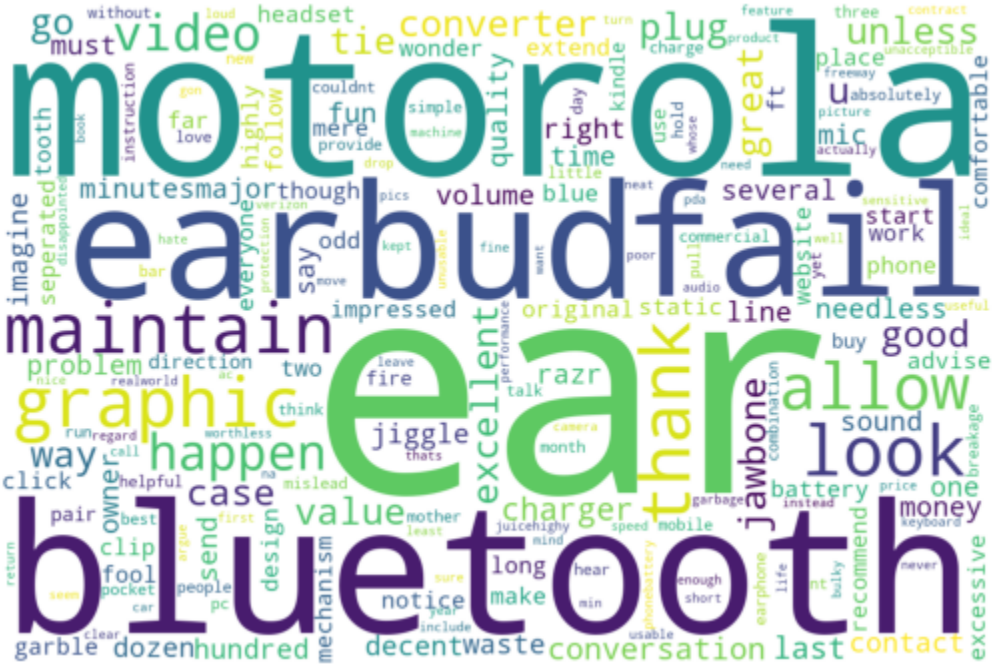
```
So there is no way for me to plug it in here in the US unless I go by a converter.      0
Good case, Excellent value.     1
Great for the jawbone.  1
Tied to charger for conversations lasting more than 45 minutes.MAJOR PROBLEMS!! 0
The mic is great.       1
cat: stdout: Broken pipe
```

[23]:
```
import pandas as pd
data = pd.read_csv(
  'sentiment labelled sentences/amazon_cells_labelled.txt',
  sep='\t',
  names=['text', 'sentiment']
)
```

Preprocessing

[24]:
```
# distribution
vocabulary = nlp.create_vocab_from_docs(data['text'])
print(nlp.get_stats_vocab(vocabulary, 2))
nlp.create_words_cloud(nlp.create_words_map(vocabulary))
```

```
(609, 1465)
```



[24]:
```
<wordcloud.wordcloud.WordCloud at 0x288b14ee0>
```

[25]:
```
# data cleaning
data['text_cleaned'] = data['text'].apply(nlp.get_doc_cleaned)
```

[26]:
```
# distribution
vocabulary = nlp.create_vocab_from_docs(data['text_cleaned'])
print(nlp.get_stats_vocab(vocabulary, 2))
nlp.create_words_cloud(nlp.create_words_map(vocabulary), is_test = True)
```

```
(602, 1452)
```

[26]:
```
<wordcloud.wordcloud.WordCloud at 0x288b14190>
```

Modeling by vectors

[27]:
```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['text_cleaned'], data['sentiment
↪'], test_size=0.2, random_state=3)
```

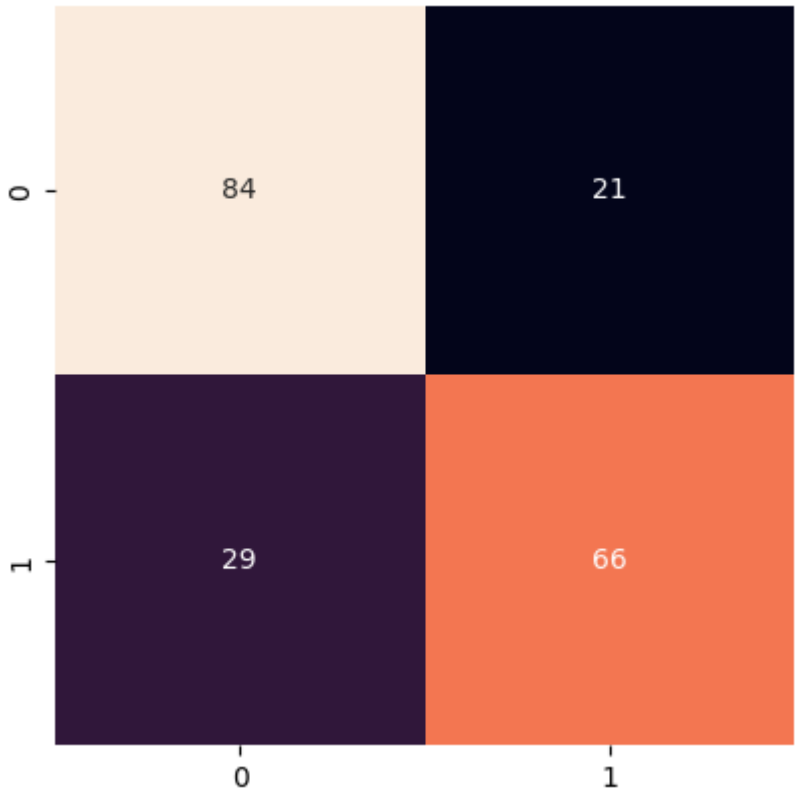
[28]:
```
# vectorization
X_vect_train = nlp.vectorize_docs(X_train, is_lemma = True)
X_vect_test = nlp.vectorize_docs(X_test, is_lemma = True, is_test = True)
```

[29]:
```
# fitting
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(random_state=3)
model.fit(X_vect_train, y_train)
```

[29]:
```
SGDClassifier(random_state=3)
```

[30]:
```
# testing
y_pred = model.predict(X_vect_test)
mtr.create_confusion_matrix(y_test, y_pred, is_test = False)
```

[30]:
```
array([[84, 21],
       [29, 66]])
```



[31]:
```
mtr.get_classification_metrics({
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[31]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.49834700984605873,
 'ROC_AUC': 0.7473684210526317,
 'Accuracy': 0.75,
 'Precision': array([0.74336283, 0.75862069]),
 'Recall': array([0.8       , 0.69473684]),
 'Fscore': array([0.7706422 , 0.72527473]),
 'Support': array([105,  95])}
```

Modeling by vectors and tokenization

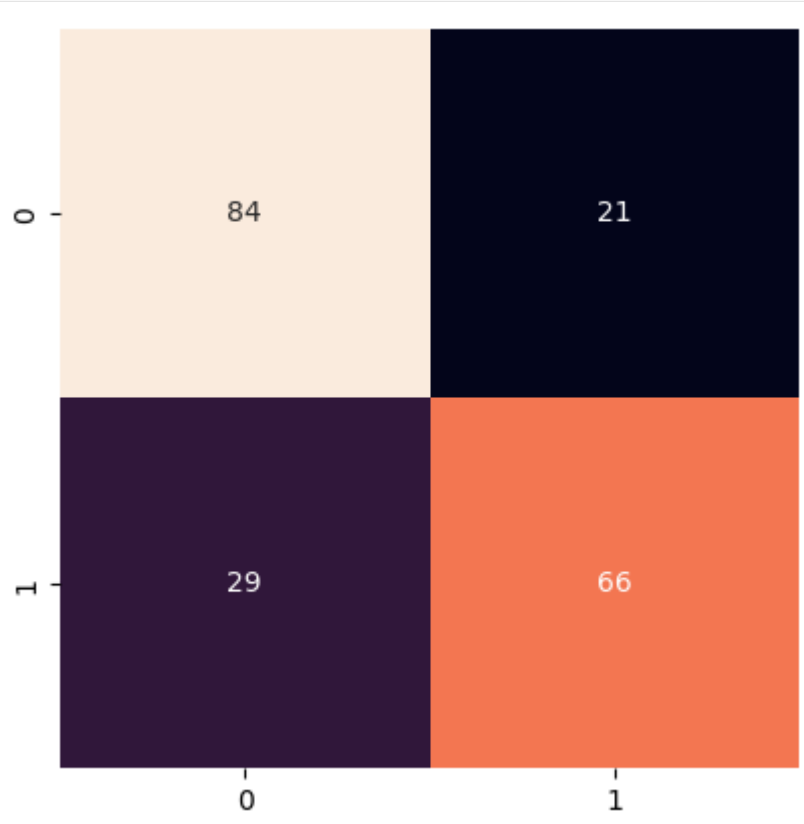
[32]:
```
# vectorization
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(analyzer = 'word', tokenizer = nlp) #, ngram_range = (1, 2)
X_vect_train = vectorizer.fit_transform(X_train)
X_vect_test = vectorizer.transform(X_test)
```

[33]:
```
# fitting
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(random_state=3)
model.fit(X_vect_train, y_train)
```

[33]:
```
SGDClassifier(random_state=3)
```

[34]:
```
# testing
y_pred = model.predict(X_vect_test)
mtr.create_confusion_matrix(y_test, y_pred, is_test = False)
```

[34]:
```
array([[84, 21],
       [29, 66]])
```



[35]:
```
mtr.get_classification_metrics({
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[35]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.49834700984605873,
 'ROC_AUC': 0.7473684210526317,
 'Accuracy': 0.75,
 'Precision': array([0.74336283, 0.75862069]),
 'Recall': array([0.8       , 0.69473684]),
 'Fscore': array([0.7706422 , 0.72527473]),
 'Support': array([105,  95])}
```

Modeling by features

[36]:
```
# data wrangling
top_100 = nlp.get_words_top(vocabulary, 100)

features = nlp.get_features_from_docs(data.text, words_top = top_100)
features_df = pd.DataFrame.from_dict(features, orient='columns')
data = pd.concat([data, features_df], axis='columns')
```

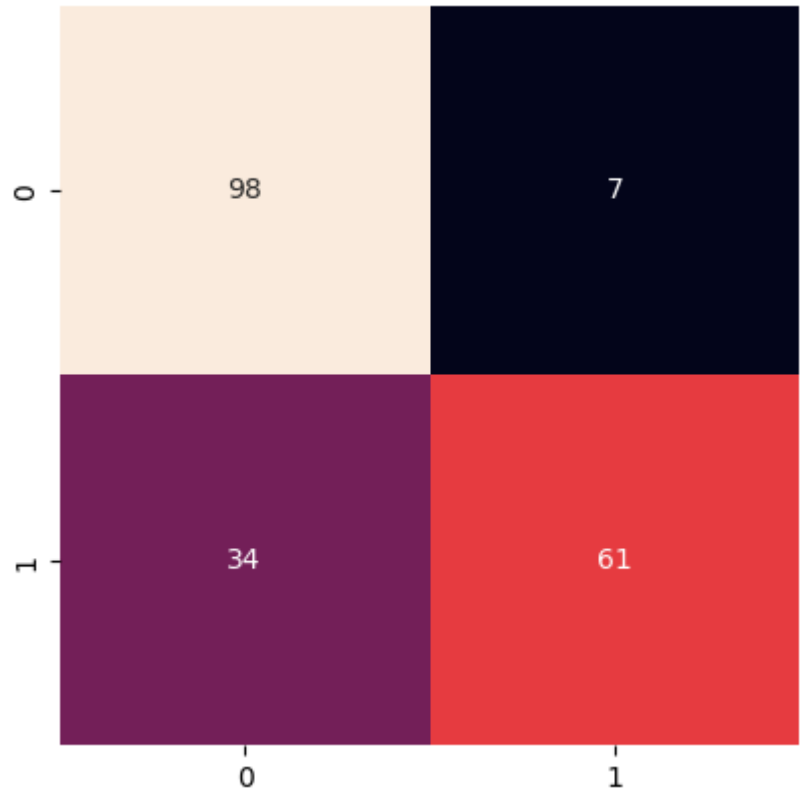
[37]:
```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data[['words_top', 'neg', 'neu', 'pos
↪', 'compound']], data['sentiment'], test_size=0.2, random_state=3)
```

[38]:
```
# fitting
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(random_state=3)
model.fit(X_train, y_train)
```

[38]:
```
SGDClassifier(random_state=3)
```

[39]:
```
# testing
y_pred = model.predict(X_test)
mtr.create_confusion_matrix(y_test, y_pred, is_test = False)
```

[39]:
```
array([[98,  7],
       [34, 61]])
```



[40]:
```
mtr.get_classification_metrics({
    "model": model,
    "X_train": np.array(X_train),
    "y_train": np.array(y_train),
    "X_test": np.array(X_test),
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[40]:
```
{'Loss': 0.23532499999999998,
 'Bias': 0.153603875,
 'Variance': 0.081721125,
 'MCC': 0.6066162196580951,
 'ROC_AUC': 0.787719298245614,
 'Accuracy': 0.795,
 'Precision': array([0.74242424, 0.89705882]),
 'Recall': array([0.93333333, 0.64210526]),
 'Fscore': array([0.82700422, 0.74846626]),
 'Support': array([105,  95])}
```

Modeling by vectors and features

[41]:
```
# data reduction
features_mix = data[['text_cleaned', 'words_top', 'neg', 'neu', 'pos', 'compound']].to_
↪dict(orient="records")
```

[42]:
```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(features_mix, data['sentiment'],␣
↪test_size=0.2, random_state=3)
```

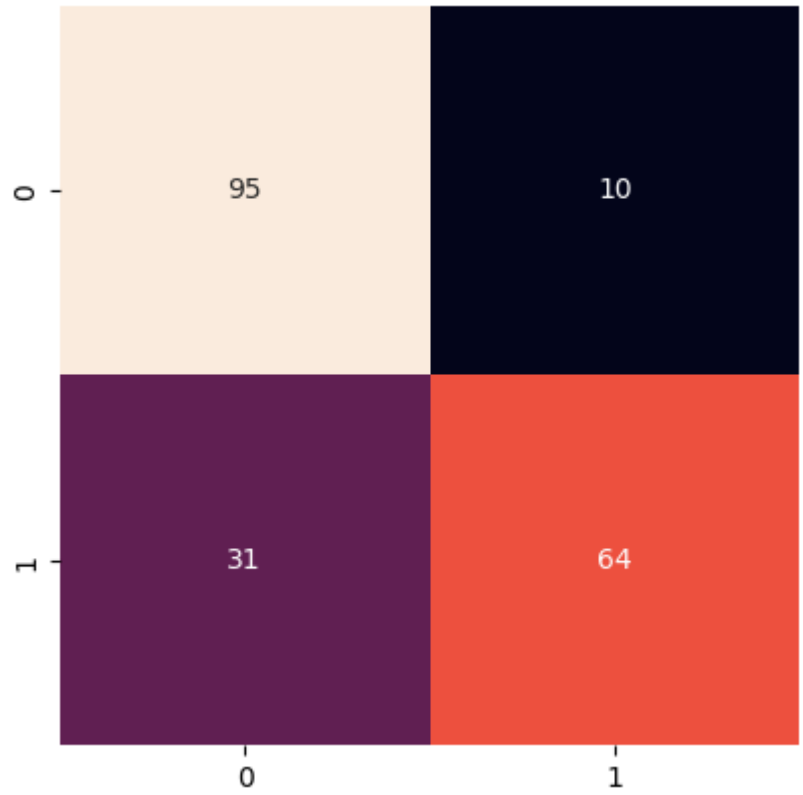
[43]:
```
# vectorization
X_vect_train = nlp.vectorize_docs(X_train)
X_vect_test = nlp.vectorize_docs(X_test, is_test = True)
```

[44]:
```
# fitting
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(random_state=3)
model.fit(X_vect_train, y_train)
```

[44]:
```
SGDClassifier(random_state=3)
```

[45]:
```
# testing
y_pred = model.predict(X_vect_test)
mtr.create_confusion_matrix(y_test, y_pred, is_test = False)
```

[45]:
```
array([[95, 10],
       [31, 64]])
```



[46]:
```
mtr.get_classification_metrics({
    "y_test": np.array(y_test),
    "y_pred": y_pred
})
```

[46]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.5982989333106916,
 'ROC_AUC': 0.7892230576441103,
 'Accuracy': 0.795,
 'Precision': array([0.75396825, 0.86486486]),
 'Recall': array([0.9047619 , 0.67368421]),
 'Fscore': array([0.82251082, 0.75739645]),
 'Support': array([105,  95])}
```

3.1.4 Example with sklearn library - pipeline

[47]:
```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data['text'], data['sentiment'],␣
↪test_size=0.2, random_state=3)
```

[48]:
```
# data preparation
vectorizer = CountVectorizer()
X_vect_train = vectorizer.fit_transform(X_train)
X_vect_test = vectorizer.transform(X_test)
# modeling
model = SGDClassifier(random_state=3)
model.fit(X_vect_train, y_train) # training
mtr.scoring(model, X_vect_test, y_test) # testing
```

[48]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.5887959168743058,
 'ROC_AUC': 0.7942355889724311,
 'Accuracy': 0.795,
 'Precision': array([0.80188679, 0.78723404]),
 'Recall': array([0.80952381, 0.77894737]),
 'Fscore': array([0.8056872 , 0.78306878]),
 'Support': array([105,  95])}
```

[49]:
```
from sklearn.pipeline import Pipeline
# data preparation
pipeline = Pipeline([
    ('count', CountVectorizer()),
    ('class', SGDClassifier(random_state=3))
])
# modeling
mtr.modeling(pipeline, X_train, y_train, X_test, y_test)
```

[49]:
```
{'Loss': 0,
 'Bias': 0,
 'Variance': 0,
 'MCC': 0.5887959168743058,
 'ROC_AUC': 0.7942355889724311,
 'Accuracy': 0.795,
 'Precision': array([0.80188679, 0.78723404]),
 'Recall': array([0.80952381, 0.77894737]),
 'Fscore': array([0.8056872 , 0.78306878]),
 'Support': array([105,  95])}
```

3.2 Datavisualization

The package only contains methods to simplify your code, so it doesn't cover all steps of machine learning process, but only those that save you lines of code.

[1]:
```
#!pip install smltk==2.2.7

# import os
# import sys
# sys.path.insert(1, os.getcwd() + '/../../../')
# print(os.getcwd())
# print(sys.path)

from smltk.datavisualization import DataVisualization
dv = DataVisualization()
```

3.2.1 Example with sklearn library - iris dataset

[2]:
```
from sklearn.datasets import load_iris
data = load_iris()
```

Datavisualization

[3]:
```
dv.get_df(data).head(5)
```

[3]:
```
   target target_name  sepal length (cm)  sepal width (cm)  petal length (cm)  \
0       0      setosa                5.1               3.5                1.4
1       0      setosa                4.9               3.0                1.4
2       0      setosa                4.7               3.2                1.3
3       0      setosa                4.6               3.1                1.5
4       0      setosa                5.0               3.6                1.4

   petal width (cm)
0               0.2
1               0.2
2               0.2
3               0.2
4               0.2
```

Preprocessing

[4]:
```
# splitting
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(data.data, data.target, test_size=0.
↪2, random_state=5)
```

Modeling

[5]:
```
# fitting
from sklearn.linear_model import SGDClassifier
model = SGDClassifier(random_state=3)
_ = model.fit(X_train, y_train)
```

[6]:
```
# testing
y_pred = model.predict(X_test)
```

Datavisualization

[7]:
```
dv.get_inference_df(data, X_test, y_test, y_pred).head(5)
```

[7]:
```
   prediction  target target_name  sepal length (cm)  sepal width (cm)  \
0           1       1  versicolor                5.8               2.7
1           2       2   virginica                6.1               2.6
2           2       2   virginica                5.8               2.8
3           0       0      setosa                4.4               3.2
4           2       2   virginica                7.2               3.6

   petal length (cm)  petal width (cm)
0                3.9               1.2
1                5.6               1.4
2                5.1               2.4
3                1.3               0.2
4                6.1               2.5
```

3.2.2 Example with torch library - image dataset

[8]:
```
import requests
from PIL import Image
url = 'https://www.projectinvictus.it/wp-content/uploads/2022/08/junk-food-scaled.jpg'
im = Image.open(requests.get(url, stream=True).raw)
```

Preprocessing

[9]:
```
# data preparation
import torchvision.transforms as transforms
transform = transforms.Compose([ transforms.Resize(800), transforms.ToTensor() ])
img = transform(im).unsqueeze(0)
```

Modeling

[10]:
```
# load pre-trained model instead to train it
import torch
model = torch.hub.load('facebookresearch/detr', 'detr_resnet50', pretrained=True)
model.eval();
```

```
Using cache found in /Users/alessandrabilardi/.cache/torch/hub/facebookresearch_detr_main
/Users/alessandrabilardi/.env/lib/python3.10/site-packages/torchvision/models/_utils.py:
↪208: UserWarning: The parameter 'pretrained' is deprecated since 0.13 and may be␣
↪removed in the future, please use 'weights' instead.
  warnings.warn(
/Users/alessandrabilardi/.env/lib/python3.10/site-packages/torchvision/models/_utils.py:
↪223: UserWarning: Arguments other than a weight enum or `None` for 'weights' are␣
↪deprecated since 0.13 and may be removed in the future. The current behavior is␣
↪equivalent to passing `weights=ResNet50_Weights.IMAGENET1K_V1`. You can also use␣
↪`weights=ResNet50_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
```

[11]:
```
# fitting
prediction = model(img)
probability, boxes = dv.get_inference_objects(im, prediction, 0.7)
```

Datavisualization

[12]:
```
# pretty view of prediction
dv.get_inference_objects_df(probability, boxes)
```

[12]:

	class	probability	xmin	ymin	xmax	\
0	bowl	0.91	121.079979	743.922058	698.561523	
0	bowl	0.92	696.310791	649.603210	1255.190918	
0	donut	1.00	2140.089355	469.912140	2559.624512	
0	donut	1.00	1892.986816	816.611267	2523.677246	
0	cup	0.88	2440.144287	-1.122757	2559.945801	
0	donut	0.78	1279.640137	728.656555	1816.003174	
0	bowl	0.99	718.509521	1025.902588	1311.592041	
0	cup	1.00	1913.083984	1.166088	2337.076904	
0	dining table	0.98	-0.428162	4.530998	2559.098877	
0	bowl	0.72	2444.689697	-1.408806	2559.865479	
0	cup	0.98	120.385170	741.396362	703.488281	
0	donut	0.82	1331.866211	146.230453	1958.745361	
0	bowl	0.80	1242.721802	728.814392	1856.770630	

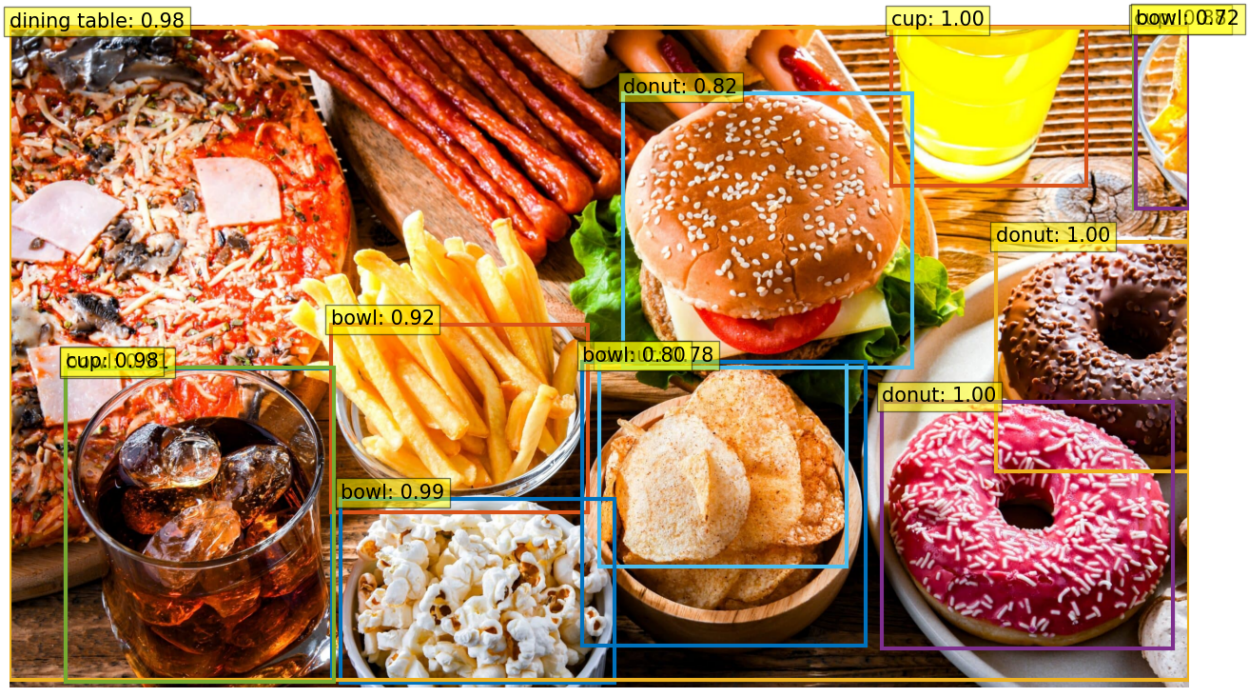
	ymax
0	1422.093384
0	1054.840454
0	966.051208
0	1351.426514
0	397.894653
0	1172.515991
0	1424.917969
0	347.949921

(continues on next page)

(continued from previous page)

```
0  1419.098145
0   396.303345
0  1423.120361
0   741.035339
0  1345.256592
```

[13]:
```
# plot image with objects detected
dv.plot_inference_objects(im, probability, boxes)
```



CHAPTER

FOUR

DEVELOPMENT

Your contribution is important but you have to follow same steps.

It will be approved changes or additions with

- methods containing title, arguments and returns descriptions
- the relative unit tests

4.1 Run tests

It is important to test your code before to create a pull request.

```
cd smltk/
virtualenv .env
source .env/bin/activate
pip3 install --upgrade -r requirements.txt
python3 -m unittest discover -v
deactivate
```

4.2 Run notebook

If you want to test your code on the usage.ipynb

```
cd smltk/
docker run --rm -p 8888:8888 -e JUPYTER_ENABLE_LAB=yes -v "$PWD":/home/jovyan/ jupyter/
↪datascience-notebook
```

CHAPTER

FIVE

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

S

INDEX

N

P

R

S

T

V

W