Lazy Predict Documentation

Release 0.2.12

Shankar Rao Pandala

Sep 28, 2022

Contents:

CHAPTER 1

Installation

1.1 Stable release

To install Lazy Predict, run this command in your terminal:

```
$ pip install lazypredict
```

This is the preferred method to install Lazy Predict, as it will always install the most recent stable release.

If you don't have pip installed, this Python installation guide can guide you through the process.

1.2 From sources

The sources for Lazy Predict can be downloaded from the Github repo.

You can either clone the public repository:

```
$ git clone git://github.com/shankarpandala/lazypredict
```

Or download the tarball:

```
$ curl -OJL https://github.com/shankarpandala/lazypredict/tarball/master
```

Once you have a copy of the source, you can install it with:

```
$ python setup.py install
```

CHAPTER 2

Usage

To use Lazy Predict in a project:

```
import lazypredict
```

CHAPTER 3

Classification

Example

```
from lazypredict.Supervised import LazyClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split
data = load_breast_cancer()
X = data.data
y= data.target
X_train, X_test, y_train, y_test = train_test_split(X, y,test_size=.5,random_state␣
↪=123)
clf = LazyClassifier(verbose=0,ignore_warnings=True, custom_metric=None)
models,predictions = clf.fit(X_train, X_test, y_train, y_test)
models
```

```
| Model                          |   Accuracy |   Balanced Accuracy |   ROC AUC |␣
↪F1 Score |   Time Taken |
|:-------------------------------|-----------:|--------------------:|----------:|-----
↪------:|-------------:|
| LinearSVC                      |   0.989474 |            0.987544 |  0.987544 |   0.
↪989462 |    0.0150008 |
| SGDClassifier                  |   0.989474 |            0.987544 |  0.987544 |   0.
↪989462 |    0.0109992 |
| MLPClassifier                  |   0.985965 |            0.986904 |  0.986904 |   0.
↪985994 |    0.426     |
| Perceptron                     |   0.985965 |            0.984797 |  0.984797 |   0.
↪985965 |    0.0120046 |
| LogisticRegression             |   0.985965 |            0.98269  |  0.98269  |   0.
↪985934 |    0.0200036 |
| LogisticRegressionCV           |   0.985965 |            0.98269  |  0.98269  |   0.
↪985934 |    0.262997  |
| SVC                            |   0.982456 |            0.979942 |  0.979942 |   0.
↪982437 |    0.0140011 |
| CalibratedClassifierCV         |   0.982456 |            0.975728 |  0.975728 |   0.
↪982357 |    0.0350015 |
```

(continues on next page)

(continued from previous page)

```
| PassiveAggressiveClassifier     |   0.975439 |           0.974448 |  0.974448 |   0.
↪975464 |    0.0130005 |
| LabelPropagation                |   0.975439 |           0.974448 |  0.974448 |   0.
↪975464 |    0.0429988 |
| LabelSpreading                  |   0.975439 |           0.974448 |  0.974448 |   0.
↪975464 |    0.0310006 |
| RandomForestClassifier          |   0.97193  |           0.969594 |  0.969594 |   0.
↪97193  |    0.033     |
| GradientBoostingClassifier      |   0.97193  |           0.967486 |  0.967486 |   0.
↪971869 |    0.166998  |
| QuadraticDiscriminantAnalysis   |   0.964912 |           0.966206 |  0.966206 |   0.
↪965052 |    0.0119994 |
| HistGradientBoostingClassifier |   0.968421 |           0.964739 |  0.964739 |   0.
↪968387 |    0.682003  |
| RidgeClassifierCV               |   0.97193  |           0.963272 |  0.963272 |   0.
↪971736 |    0.0130029 |
| RidgeClassifier                 |   0.968421 |           0.960525 |  0.960525 |   0.
↪968242 |    0.0119977 |
| AdaBoostClassifier              |   0.961404 |           0.959245 |  0.959245 |   0.
↪961444 |    0.204998  |
| ExtraTreesClassifier            |   0.961404 |           0.957138 |  0.957138 |   0.
↪961362 |    0.0270066 |
| KNeighborsClassifier            |   0.961404 |           0.95503  |  0.95503  |   0.
↪961276 |    0.0560005 |
| BaggingClassifier               |   0.947368 |           0.954577 |  0.954577 |   0.
↪947882 |    0.0559971 |
| BernoulliNB                     |   0.950877 |           0.951003 |  0.951003 |   0.
↪951072 |    0.0169988 |
| LinearDiscriminantAnalysis      |   0.961404 |           0.950816 |  0.950816 |   0.
↪961089 |    0.0199995 |
| GaussianNB                      |   0.954386 |           0.949536 |  0.949536 |   0.
↪954337 |    0.0139935 |
| NuSVC                           |   0.954386 |           0.943215 |  0.943215 |   0.
↪954014 |    0.019989  |
| DecisionTreeClassifier          |   0.936842 |           0.933693 |  0.933693 |   0.
↪936971 |    0.0170023 |
| NearestCentroid                 |   0.947368 |           0.933506 |  0.933506 |   0.
↪946801 |    0.0160074 |
| ExtraTreeClassifier             |   0.922807 |           0.912168 |  0.912168 |   0.
↪922462 |    0.0109999 |
| CheckingClassifier              |   0.361404 |           0.5      |  0.5      |   0.
↪191879 |    0.0170043 |
| DummyClassifier                 |   0.512281 |           0.489598 |  0.489598 |   0.
↪518924 |    0.0119965 |
```

CHAPTER 4

Regression

Example

```
from lazypredict.Supervised import LazyRegressor
from sklearn import datasets
from sklearn.utils import shuffle
import numpy as np
boston = datasets.load_boston()
X, y = shuffle(boston.data, boston.target, random_state=13)
X = X.astype(np.float32)
offset = int(X.shape[0] * 0.9)
X_train, y_train = X[:offset], y[:offset]
X_test, y_test = X[offset:], y[offset:]
reg = LazyRegressor(verbose=0,ignore_warnings=False, custom_metric=None )
models,predictions = reg.fit(X_train, X_test, y_train, y_test)


| Model                         |   R-Squared |     RMSE |   Time Taken |
|:------------------------------|------------:|---------:|-------------:|
| SVR                           |    0.877199 |  2.62054 |    0.0330021 |
| RandomForestRegressor         |    0.874429 |  2.64993 |    0.0659981 |
| ExtraTreesRegressor           |    0.867566 |  2.72138 |    0.0570002 |
| AdaBoostRegressor             |    0.865851 |  2.73895 |    0.144999  |
| NuSVR                         |    0.863712 |  2.7607  |    0.0340044 |
| GradientBoostingRegressor     |    0.858693 |  2.81107 |    0.13      |
| KNeighborsRegressor           |    0.826307 |  3.1166  |    0.0179954 |
| HistGradientBoostingRegressor |    0.810479 |  3.25551 |    0.820995  |
| BaggingRegressor              |    0.800056 |  3.34383 |    0.0579946 |
| MLPRegressor                  |    0.750536 |  3.73503 |    0.725997  |
| HuberRegressor                |    0.736973 |  3.83522 |    0.0370018 |
| LinearSVR                     |    0.71914  |  3.9631  |    0.0179989 |
| RidgeCV                       |    0.718402 |  3.9683  |    0.018003  |
| BayesianRidge                 |    0.718102 |  3.97041 |    0.0159984 |
| Ridge                         |    0.71765  |  3.9736  |    0.0149941 |
| LinearRegression              |    0.71753  |  3.97444 |    0.0190051 |
```

(continues on next page)

(continued from previous page)

TransformedTargetRegressor	0.71753	3.97444	0.012001
LassoCV	0.717337	3.9758	0.0960066
ElasticNetCV	0.717104	3.97744	0.0860076
LassoLarsCV	0.717045	3.97786	0.0490005
LassoLarsIC	0.716636	3.98073	0.0210001
LarsCV	0.715031	3.99199	0.0450008
Lars	0.715031	3.99199	0.0269964
SGDRegressor	0.714362	3.99667	0.0210009
RANSACRegressor	0.707849	4.04198	0.111998
ElasticNet	0.690408	4.16088	0.0190012
Lasso	0.662141	4.34668	0.0180018
OrthogonalMatchingPursuitCV	0.591632	4.77877	0.0180008
ExtraTreeRegressor	0.583314	4.82719	0.0129974
PassiveAggressiveRegressor	0.556668	4.97914	0.0150032
GaussianProcessRegressor	0.428298	5.65425	0.0580051
OrthogonalMatchingPursuit	0.379295	5.89159	0.0180039
DecisionTreeRegressor	0.318767	6.17217	0.0230272
DummyRegressor	-0.0215752	7.55832	0.0140116
LassoLars	-0.0215752	7.55832	0.0180008
KernelRidge	-8.24669	22.7396	0.0309792

Warning: Regression and Classification are replaced with LazyRegressor and LazyClassifier. Regression and Classification classes will be removed in next release

CHAPTER 5

Indices and tables

- genindex
- modindex
- search