Lazy Predict – Best Suitable Model.

What is Lazy Predict?

- It is one of the best python libraries that helps you to semi-automate your Machine Learning Task.
- It builds a lot of basic models without much code and helps understand which models work better without any parameter tuning.

Solving Breast Cancer Problem(Classification) Statement using Lazy Predict

Dataset Description: Here we are using Breast Cancer Dataset in which we have to predict whether a person suffering from cancer or not.

import the required library:

In [1]:
```
### importing lazypredict library
import lazypredict

### importing LazyClassifier for classification problem
from lazypredict.Supervised import LazyClassifier

### importing LazyClassifier for classification problem because here we are solvi
from lazypredict.Supervised import LazyClassifier

### importing breast Cancer Dataset from sklearn
from sklearn.datasets import load_breast_cancer

### spliting dataset into training and testing part
from sklearn.model_selection import train_test_split

# Hide warnings
import warnings
warnings.filterwarnings("ignore")
```

```
D:\Anaconda3-2020.11-Windows-x86_64\lib\site-packages\sklearn\utils\deprecatio
n.py:143: FutureWarning: The sklearn.utils.testing module is  deprecated in ver
sion 0.22 and will be removed in version 0.24. The corresponding classes / func
tions should instead be imported from sklearn.utils. Anything that cannot be im
ported from sklearn.utils is now part of the private API.
  warnings.warn(message, FutureWarning)
```

Load the dataset:

In [2]:
```
### storing dataset in data variable
data = load_breast_cancer()
```

separate out dependent and independent features:

In [3]:
```
### separating dataset into dependent and independent features
X = data.data
y = data.target
```

split the dataset into the training and testing part:

In [4]:
```
### splitting dataset into training and testing part(50% training and 50% testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.5, random_st
```

create an object of LazyClassifier class:

In [5]:
```
clf = LazyClassifier(verbose=0, ignore_warnings=True, custom_metric = None)
```

Fit our training and testing data to the LazyClassifier object:

In [6]:
```
### fitting data in LazyClassifier
models,predictions = clf.fit(X_train, X_test, y_train, y_test)

### lets check which model did better on Breast Cancer Dataset
print(models)
```

```
100%|██████████| 29/29 [00:00<00:00, 32.49it/s]
```

Model	Accuracy	Balanced Accuracy	ROC AUC	F1 Score
LinearSVC	0.99	0.99	0.99	0.99
Perceptron	0.99	0.98	0.98	0.99
LogisticRegression	0.99	0.98	0.98	0.99
SVC	0.98	0.98	0.98	0.98
XGBClassifier	0.98	0.98	0.98	0.98
LabelPropagation	0.98	0.97	0.97	0.98
LabelSpreading	0.98	0.97	0.97	0.98
BaggingClassifier	0.97	0.97	0.97	0.97
PassiveAggressiveClassifier	0.98	0.97	0.97	0.98
SGDClassifier	0.98	0.97	0.97	0.98
RandomForestClassifier	0.97	0.97	0.97	0.97
CalibratedClassifierCV	0.98	0.97	0.97	0.98
QuadraticDiscriminantAnalysis	0.96	0.97	0.97	0.97
ExtraTreesClassifier	0.97	0.96	0.96	0.97
RidgeClassifierCV	0.97	0.96	0.96	0.97
LGBMClassifier	0.96	0.96	0.96	0.96
RidgeClassifier	0.97	0.96	0.96	0.97
AdaBoostClassifier	0.96	0.96	0.96	0.96
KNeighborsClassifier	0.96	0.96	0.96	0.96
BernoulliNB	0.95	0.95	0.95	0.95
LinearDiscriminantAnalysis	0.96	0.95	0.95	0.96
GaussianNB	0.95	0.95	0.95	0.95
NuSVC	0.95	0.94	0.94	0.95
ExtraTreeClassifier	0.94	0.93	0.93	0.94
NearestCentroid	0.95	0.93	0.93	0.95
DecisionTreeClassifier	0.93	0.93	0.93	0.93
DummyClassifier	0.53	0.50	0.50	0.53

Model	Time Taken
LinearSVC	0.01
Perceptron	0.01
LogisticRegression	0.02
SVC	0.01
XGBClassifier	0.07
LabelPropagation	0.02
LabelSpreading	0.02
BaggingClassifier	0.04
PassiveAggressiveClassifier	0.01
SGDClassifier	0.01
RandomForestClassifier	0.16
CalibratedClassifierCV	0.03
QuadraticDiscriminantAnalysis	0.01
ExtraTreesClassifier	0.12
RidgeClassifierCV	0.01

```
LGBMClassifier                     0.07
RidgeClassifier                    0.01
AdaBoostClassifier                 0.12
KNeighborsClassifier               0.02
BernoulliNB                        0.01
LinearDiscriminantAnalysis         0.01
GaussianNB                         0.01
NuSVC                              0.02
ExtraTreeClassifier                0.01
NearestCentroid                    0.01
DecisionTreeClassifier             0.01
DummyClassifier                    0.01
```

Solving Boston House Price Prediction(Regression) Problem using Lazy Predict :

Let's import all required library:

In [7]:
```
### Importing LazyRegressor
from lazypredict.Supervised import LazyRegressor

### Importing dataset available in sklearn
from sklearn import datasets
from sklearn.utils import shuffle
import numpy as np
```

Load the dataset:

In [8]:
```
### storing the Boston dataset in variable
boston = datasets.load_boston()
```

separate out dependent and independent features:

In [9]:
```
### loading and shuffling the dataset
X, y = shuffle(boston.data, boston.target, random_state=13)
offset = int(X.shape[0] * 0.9)
```

split the dataset into the training and testing part:

In [10]:
```
### splitting dataset into training and testing part.
X_train, y_train = X[:offset], y[:offset]
X_test, y_test = X[offset:], y[offset:]
```

create an object of LazyRegressor class:

In [11]:
```
### fitting data in LazyRegressor because here we are solving Regression use case
reg = LazyRegressor(verbose=0, ignore_warnings=False, custom_metric=None)
```

fit our training and testing data to the LazyClassifier object:

In [12]:
```
### fitting data in LazyClassifier
models, predictions = reg.fit(X_train, X_test, y_train, y_test)

### lets check which model did better on Breast Cancer Dataset
print(models)
```

```
100%|██████████| 42/42 [00:04<00:00,  8.75it/s]
```

Model	Adjusted R-Squared	R-Squared	RMSE	Time Taken
SVR	0.83	0.88	2.62	0.04
BaggingRegressor	0.83	0.88	2.63	0.04
NuSVR	0.82	0.86	2.76	0.03
RandomForestRegressor	0.81	0.86	2.79	0.65
XGBRegressor	0.81	0.86	2.79	0.08
GradientBoostingRegressor	0.81	0.86	2.84	0.11
ExtraTreesRegressor	0.79	0.84	2.98	0.20
HistGradientBoostingRegressor	0.77	0.83	3.06	1.03
AdaBoostRegressor	0.77	0.83	3.06	0.10
PoissonRegressor	0.77	0.83	3.11	0.03
LGBMRegressor	0.77	0.83	3.11	0.07
KNeighborsRegressor	0.77	0.83	3.12	0.04
DecisionTreeRegressor	0.65	0.74	3.79	0.01
MLPRegressor	0.65	0.74	3.80	1.50
HuberRegressor	0.64	0.74	3.84	0.06
GammaRegressor	0.64	0.73	3.88	0.01
LinearSVR	0.62	0.72	3.96	0.03
RidgeCV	0.62	0.72	3.97	0.02
BayesianRidge	0.62	0.72	3.97	0.01
Ridge	0.62	0.72	3.97	0.02
TransformedTargetRegressor	0.62	0.72	3.97	0.02
LinearRegression	0.62	0.72	3.97	0.02
ElasticNetCV	0.62	0.72	3.98	0.06
LassoCV	0.62	0.72	3.98	0.13
LassoLarsIC	0.62	0.72	3.98	0.02
LassoLarsCV	0.62	0.72	3.98	0.05
Lars	0.61	0.72	3.99	0.02
LarsCV	0.61	0.71	4.02	0.05
SGDRegressor	0.60	0.70	4.07	0.02
TweedieRegressor	0.59	0.70	4.12	0.01
GeneralizedLinearRegressor	0.59	0.70	4.12	0.01
ElasticNet	0.58	0.69	4.16	0.01
Lasso	0.54	0.66	4.35	0.02
RANSACRegressor	0.53	0.65	4.41	0.12
OrthogonalMatchingPursuitCV	0.45	0.59	4.78	0.03
PassiveAggressiveRegressor	0.37	0.54	5.09	0.02
GaussianProcessRegressor	0.23	0.43	5.65	0.03
OrthogonalMatchingPursuit	0.16	0.38	5.89	0.02
ExtraTreeRegressor	0.08	0.32	6.17	0.01
DummyRegressor	-0.38	-0.02	7.56	0.01
LassoLars	-0.38	-0.02	7.56	0.02
KernelRidge	-11.50	-8.25	22.74	0.03