Sec 1 HW 8

February 29, 2024

1 0.) Import and Clean data

[1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
```

[2]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import BaggingClassifier
from sklearn.datasets import make_classification
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.tree import plot_tree
from sklearn.metrics import confusion_matrix
import seaborn as sns
```

[3]:
```
#drive.mount('/content/gdrive/', force_remount = True)
```

[6]:
```
df = pd.read_csv('/Users/adrianonggowarsito/Desktop/bank-additional-full.csv')
```

[7]:
```
df.head()
```

[7]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	…	campaign	pdays	previous	poutcome	emp.var.rate
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1
2	37	services	married	high.school	no	yes	no	telephone	may	mon	…	1	999	0	nonexistent	1.1
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	…	1	999	0	nonexistent	1.1
4	56	services	married	high.school	no	no	yes	telephone	may	mon	…	1	999	0	nonexistent	1.1

```
   cons.price.idx  cons.conf.idx  euribor3m  nr.employed   y
0          93.994          -36.4      4.857       5191.0  no
1          93.994          -36.4      4.857       5191.0  no
2          93.994          -36.4      4.857       5191.0  no
3          93.994          -36.4      4.857       5191.0  no
4          93.994          -36.4      4.857       5191.0  no

[5 rows x 21 columns]
```

[8]:
```
df = df.drop(["default",␣
 ↪"pdays",         "previous",         "poutcome",         "emp.var.
 ↪rate",         "cons.price.idx",         "cons.conf.
 ↪idx",         "euribor3m",         "nr.employed"], axis = 1)
df = pd.get_dummies(df, columns = ["loan",␣
 ↪"job","marital","housing","contact","day_of_week", "campaign", "month",␣
 ↪"education"],drop_first = True)
```

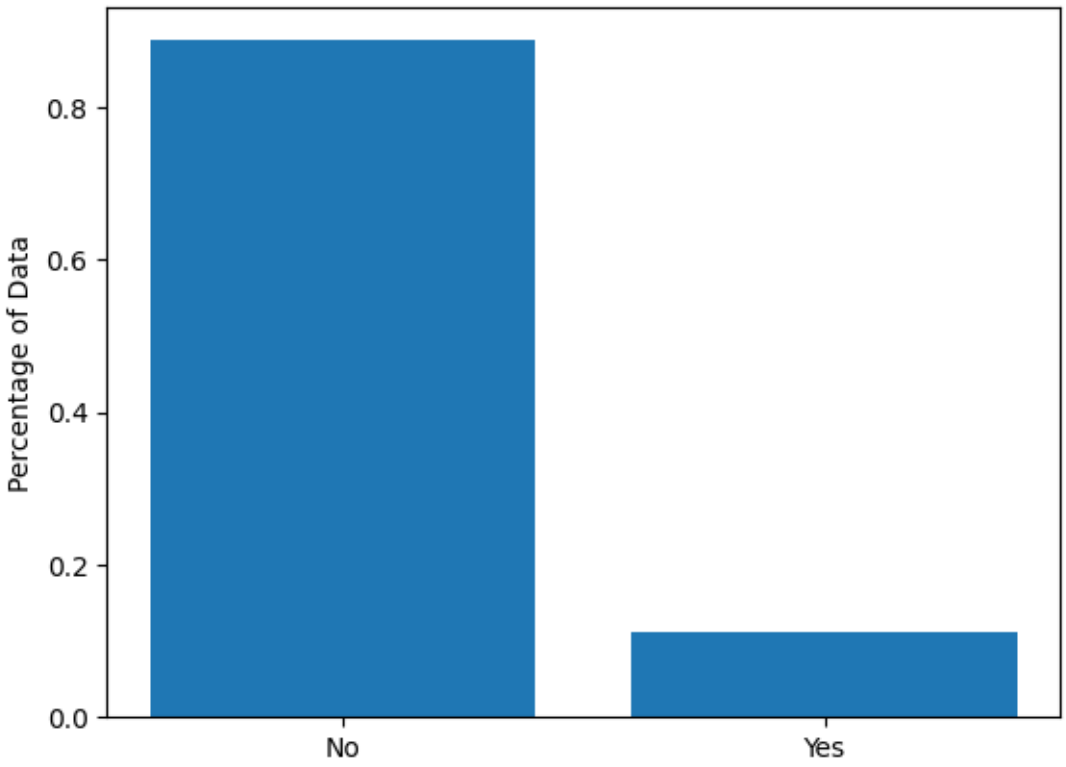
[9]:
```
df.head()
```

[9]:
```
   age  duration   y  loan_unknown  loan_yes  job_blue-collar  \
0   56       261  no             0         0                0
1   57       149  no             0         0                0
2   37       226  no             0         0                0
3   40       151  no             0         0                0
4   56       307  no             0         1                0

   job_entrepreneur  job_housemaid  job_management  job_retired  …  \
0                 0              1               0            0  …
1                 0              0               0            0  …
2                 0              0               0            0  …
3                 0              0               0            0  …
4                 0              0               0            0  …

   month_nov  month_oct  month_sep  education_basic.6y  education_basic.9y  \
0          0          0          0                   0                   0
1          0          0          0                   0                   0
2          0          0          0                   0                   0
3          0          0          0                   1                   0
4          0          0          0                   0                   0

   education_high.school  education_illiterate  education_professional.course  \
0                      0                     0                              0
1                      1                     0                              0
2                      1                     0                              0
3                      0                     0                              0
4                      1                     0                              0
```

```
   education_university.degree  education_unknown
0                            0                  0
1                            0                  0
2                            0                  0
3                            0                  0
4                            0                  0

[5 rows x 83 columns]
```

[10]:
```
y = pd.get_dummies(df["y"], drop_first = True)
X = df.drop(["y"], axis = 1)
```

[11]:
```
obs = len(y)
plt.bar(["No","Yes"],[len(y[y.yes==0])/obs,len(y[y.yes==1])/obs])
plt.ylabel("Percentage of Data")
plt.show()
```



[13]:
```
# Train Test Split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,␣
 ↪random_state=42)

scaler = StandardScaler().fit(X_train)
```

```
X_scaled = scaler.transform(X_train)
X_test = scaler.transform(X_test)
```

2 1.) Based on the visualization above, use your expert opinion to transform the data based on what we learned this quarter

[14]:
```
################
###TRANSFORM###
################
from imblearn.over_sampling import SMOTE
oversample = SMOTE()
X_scaled,y_train = oversample.fit_resample(X_scaled,y_train)
```

3 2.) Build and visualize a decision tree of Max Depth 3. Show the confusion matrix.

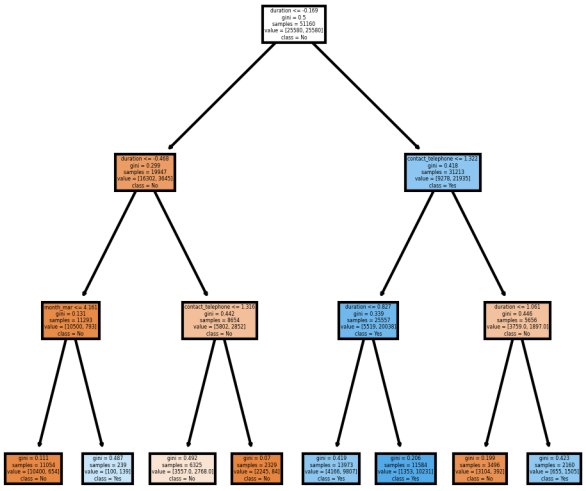
[15]:
```
dtree_main = DecisionTreeClassifier(max_depth = 3)
dtree_main.fit(X_scaled, y_train)
```

[15]:
```
DecisionTreeClassifier(max_depth=3)
```

[16]:
```
fig, axes = plt.subplots(nrows = 1,ncols = 1,figsize = (4,4), dpi=300)
plot_tree(dtree_main, filled = True, feature_names = X.columns,␣
 ↪class_names=["No","Yes"])

#fig.savefig('imagename.png')
```

[16]:
```
[Text(0.5, 0.875, 'duration <= -0.169\ngini = 0.5\nsamples = 51160\nvalue =
[25580, 25580]\nclass = No'),
 Text(0.25, 0.625, 'duration <= -0.468\ngini = 0.299\nsamples = 19947\nvalue =
[16302, 3645]\nclass = No'),
 Text(0.125, 0.375, 'month_mar <= 4.161\ngini = 0.131\nsamples = 11293\nvalue =
[10500, 793]\nclass = No'),
 Text(0.0625, 0.125, 'gini = 0.111\nsamples = 11054\nvalue = [10400, 654]\nclass
= No'),
 Text(0.1875, 0.125, 'gini = 0.487\nsamples = 239\nvalue = [100, 139]\nclass =
Yes'),
 Text(0.375, 0.375, 'contact_telephone <= 1.316\ngini = 0.442\nsamples =
8654\nvalue = [5802, 2852]\nclass = No'),
 Text(0.3125, 0.125, 'gini = 0.492\nsamples = 6325\nvalue = [3557.0,
2768.0]\nclass = No'),
 Text(0.4375, 0.125, 'gini = 0.07\nsamples = 2329\nvalue = [2245, 84]\nclass =
No'),
 Text(0.75, 0.625, 'contact_telephone <= 1.322\ngini = 0.418\nsamples =
```

```
31213\nvalue = [9278, 21935]\nclass = Yes'),
 Text(0.625, 0.375, 'duration <= 0.827\ngini = 0.339\nsamples = 25557\nvalue =
[5519, 20038]\nclass = Yes'),
 Text(0.5625, 0.125, 'gini = 0.419\nsamples = 13973\nvalue = [4166, 9807]\nclass
= Yes'),
 Text(0.6875, 0.125, 'gini = 0.206\nsamples = 11584\nvalue = [1353,
10231]\nclass = Yes'),
 Text(0.875, 0.375, 'duration <= 1.061\ngini = 0.446\nsamples = 5656\nvalue =
[3759.0, 1897.0]\nclass = No'),
 Text(0.8125, 0.125, 'gini = 0.199\nsamples = 3496\nvalue = [3104, 392]\nclass =
No'),
 Text(0.9375, 0.125, 'gini = 0.423\nsamples = 2160\nvalue = [655, 1505]\nclass =
Yes')]
```

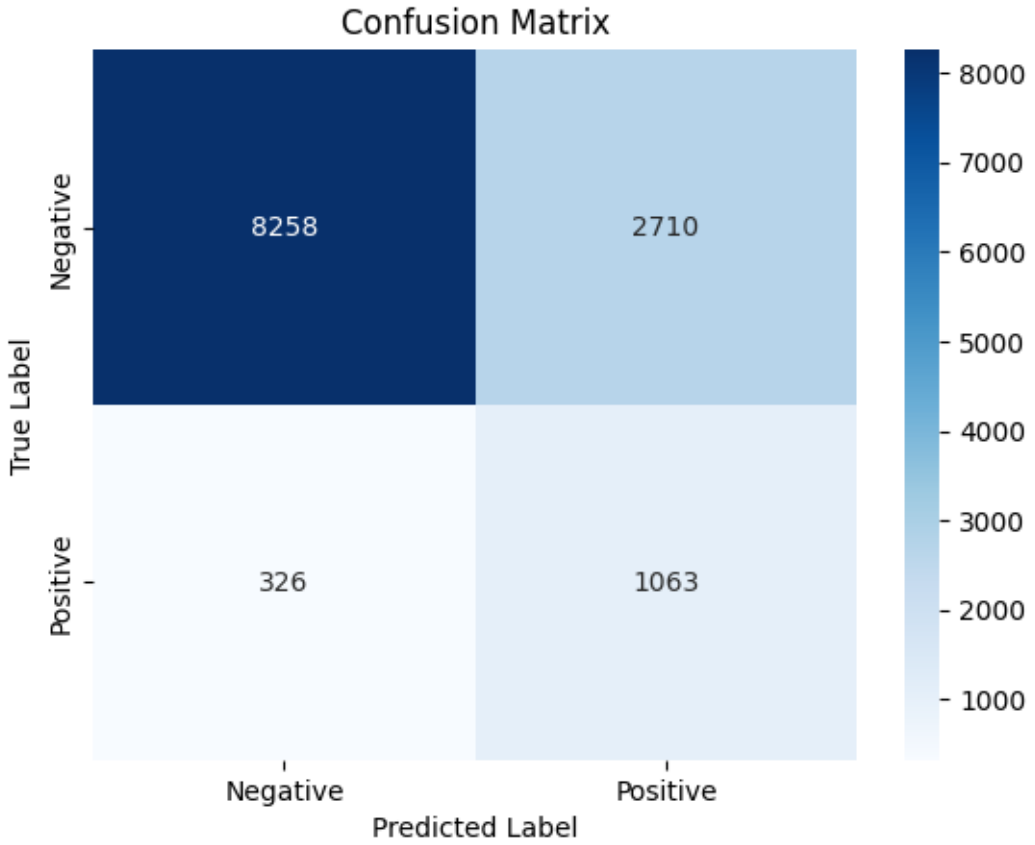


4 1b.) Confusion matrix on out of sample data. Visualize and store as variable

[17]:
```
y_pred = dtree_main.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[18]:
```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
```

```
sns.heatmap(cm_raw, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



5 3.) Use bagging on your descision tree

[19]:
```
dtree = DecisionTreeClassifier(max_depth = 3)
```

[20]:
```
bagging = BaggingClassifier(estimator=dtree,
                  n_estimators = 100,
                  max_samples = 0.5,
                  max_features = 1.)
```

[21]:
```
bagging.fit(X_scaled,y_train)
```

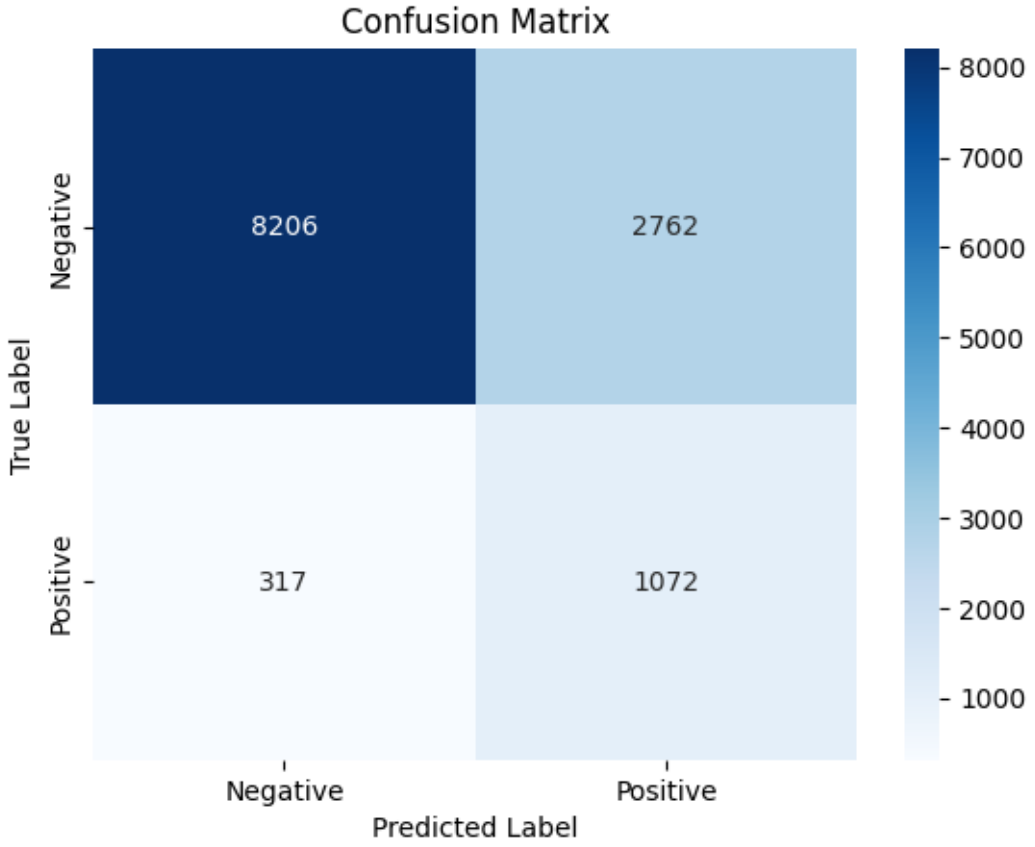
```
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
```

```
packages/sklearn/ensemble/_bagging.py:782: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[21]:
```
BaggingClassifier(estimator=DecisionTreeClassifier(max_depth=3),
                  max_samples=0.5, n_estimators=100)
```

[22]:
```
y_pred = bagging.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[23]:
```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_raw, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



6 4.) Boost your tree

[24]:
```
from sklearn.ensemble import AdaBoostClassifier
```

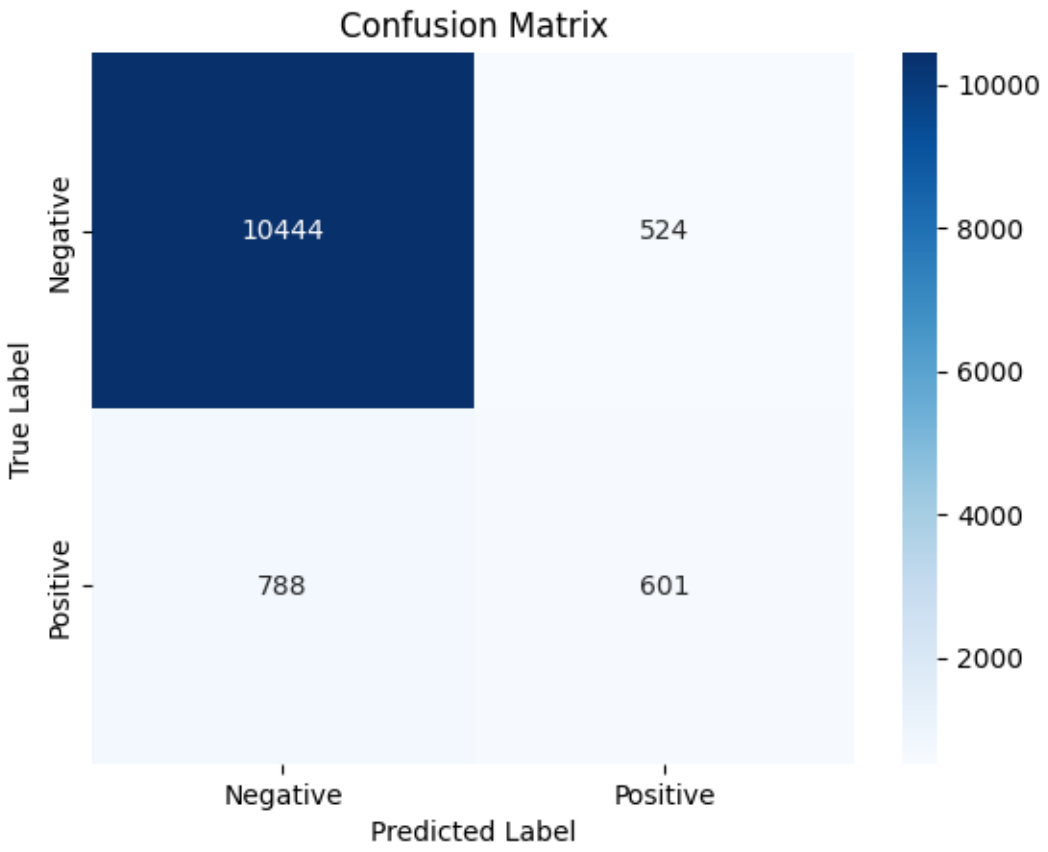
[25]:
```
boost = AdaBoostClassifier(estimator=dtree,
                           n_estimators = 100)
boost.fit(X_scaled,y_train)
```

```
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/utils/validation.py:1229: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/ensemble/_weight_boosting.py:519: FutureWarning: The SAMME.R
algorithm (the default) is deprecated and will be removed in 1.6. Use the SAMME
algorithm to circumvent this warning.
  warnings.warn(
```

[25]:
```
AdaBoostClassifier(estimator=DecisionTreeClassifier(max_depth=3),
                   n_estimators=100)
```

[26]:
```
y_pred = boost.predict(X_test)
y_true = y_test
cm_raw = confusion_matrix(y_true, y_pred)
```

[27]:
```
class_labels = ['Negative', 'Positive']

# Plot the confusion matrix as a heatmap
sns.heatmap(cm_raw, annot=True, fmt='d', cmap='Blues',␣
 ↪xticklabels=class_labels, yticklabels=class_labels)
plt.title('Confusion Matrix')
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.show()
```



7 5.) Create a superlearner with at least 4 base learner models. Use a logistic reg for your metalearner. Interpret your coefficients and save your CM.

[28]:
```
pip install mlens
```

```
Collecting mlens
  Downloading mlens-0.2.3-py2.py3-none-any.whl (227 kB)
                              227.7/227.7

kB 5.3 MB/s eta 0:00:0000:01
Requirement already satisfied: numpy>=1.11 in
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-packages (from mlens)
(1.26.4)
Requirement already satisfied: scipy>=0.17 in
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-packages (from mlens)
(1.12.0)
Installing collected packages: mlens
Successfully installed mlens-0.2.3
```

```
Note: you may need to restart the kernel to use updated packages.
```

[29]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.neighbors import KNeighborsClassifier
from mlens.ensemble import SuperLearner
```

```
[MLENS] backend: threading
```

```
---------------------------------------------------------------------------
ImportError                               Traceback (most recent call last)
Cell In[29], line 4
      2 from sklearn.ensemble import RandomForestClassifier
      3 from sklearn.neighbors import KNeighborsClassifier
----> 4 from mlens.ensemble import SuperLearner

File ~/anaconda3/lib/python3.10/site-packages/mlens/ensemble/__init__.py:12
      1 """ML-Ensemble
      2
      3 :author: Sebastian Flennerhag
   (…)
      9 can be used in conjunction with any other standard estimator.
     10 """
---> 12 from .super_learner import SuperLearner
     13 from .blend import BlendEnsemble
     14 from .subsemble import Subsemble

File ~/anaconda3/lib/python3.10/site-packages/mlens/ensemble/super_learner.py:1
      1 """ML-ENSEMBLE
      2
      3 :author: Sebastian Flennerhag
   (…)
      7 Super Learner class. Fully integrable with Scikit-learn.
      8 """
     10 from __future__ import division
---> 12 from .base import BaseEnsemble
     13 from ..index import FoldIndex, FullIndex
     16 class SuperLearner(BaseEnsemble):

File ~/anaconda3/lib/python3.10/site-packages/mlens/ensemble/base.py:20
     17 import warnings
     19 from .. import config
---> 20 from ..parallel import Layer, ParallelProcessing, make_group
     21 from ..parallel.base import BaseStacker
     22 from ..externals.sklearn.validation import check_random_state

File ~/anaconda3/lib/python3.10/site-packages/mlens/parallel/__init__.py:15
      1 """ML-ENSEMBLE
```

```
      2
      3 :author: Sebastian Flennerhag
   (…)
     12 as handles for multiple instances and wrappers for standard parallel jo␣
 ↪calls.
     13 """
     14 from .backend import ParallelProcessing, ParallelEvaluation, Job,␣
 ↪dump_array
---> 15 from .learner import Learner, EvalLearner, Transformer, EvalTransformer
     16 from .layer import Layer
     17 from .handles import Group, make_group, Pipeline

File ~/anaconda3/lib/python3.10/site-packages/mlens/parallel/learner.py:24
     19 from ._base_functions import (
     20     slice_array, set_output_columns, assign_predictions,␣
 ↪score_predictions,
     21     replace, save, load, prune_files, check_params)
     22 from .base import OutputMixin, ProbaMixin, IndexMixin, BaseEstimator
---> 24 from ..metrics import Data
     25 from ..utils import safe_print, print_time, format_name,␣
 ↪assert_valid_pipeline
     26 from ..utils.exceptions import (NotFittedError, FitFailedWarning,
     27                                 ParallelProcessingError,␣
 ↪NotInitializedError)

File ~/anaconda3/lib/python3.10/site-packages/mlens/metrics/__init__.py:10
      1 """ML-ENSEMBLE
      2
      3 :author: Sebastian Flennerhag
   (…)
      7 Metric utilities and functions.
      8 """
---> 10 from ..externals.sklearn.scorer import make_scorer
     11 from .metrics import rmse, mape, wape
     12 from .utils import assemble_table, assemble_data, Data

File ~/anaconda3/lib/python3.10/site-packages/mlens/externals/sklearn/scorer.py
 ↪33
     31 from .. import six
     32 from .base import is_regressor
---> 33 from .type_of_target import type_of_target
     36 class _BaseScorer(six.with_metaclass(ABCMeta, object)):
     37     def __init__(self, score_func, sign, kwargs):

File ~/anaconda3/lib/python3.10/site-packages/mlens/externals/sklearn/
 ↪type_of_target.py:10
      6 # Author: Arnaud Joly, Joel Nothman, Hamzeh Alsalhi
      7 # License: BSD 3 clause
```

```
      9 from __future__ import division
---> 10 from collections import Sequence
     13 from scipy.sparse import issparse
     14 from scipy.sparse.base import spmatrix

ImportError: cannot import name 'Sequence' from 'collections' (/Users/
 ↪adrianonggowarsito/anaconda3/lib/python3.10/collections/__init__.py)
```

[30]:
```
base_predictions =[list(dtree_main.predict(X_scaled)),
list(boost.predict(X_scaled)),
list(bagging.predict(X_scaled))]
```

[31]:
```
n = len(base_predictions[0])
n
```

[31]:
```
51160
```

[32]:
```
base_predictions = np.array(base_predictions).transpose()
```

[33]:
```
super_learner = LogisticRegression()
```

[34]:
```
super_learner.fit(base_predictions,y_train)
```

```
/Users/adrianonggowarsito/anaconda3/lib/python3.10/site-
packages/sklearn/utils/validation.py:1229: DataConversionWarning: A column-
vector y was passed when a 1d array was expected. Please change the shape of y
to (n_samples, ), for example using ravel().
  y = column_or_1d(y, warn=True)
```

[34]:
```
LogisticRegression()
```

[35]:
```
super_learner.coef_
```

[35]:
```
array([[0.75159341, 5.35570314, 0.82853626]])
```

Boost method has the most weight = more robust

8 6.)