Stackin___For_Classification_with_Python

May 18, 2022

1 Stacking For Classification with Python

Estimated time needed: **45** minutes

1.1 Objectives

After completing this lab you will be able to:

- **Understand** what Stacking is and how it works
- **Understand** that Random Forests have less Correlation between predictors in their ensemble, improving accuracy
- **Apply** Stacking
- **Understand** Hyperparameters selection in Stacking

Stacking takes several classification models called base learners and uses their output as the input for the meta-classifier. Consider the figure below the base learners $h\_1(x)$, $h\_2(x)$, $h\_3(x)$, and $h\_4(x)$ has the output $\hat{y} * 1$, $\hat{y} * 2$, $\hat{y} * 3$, $\hat{y} * 4$. These are used as an input to the meta classifier $H(\hat{y} * 1, \hat{y} * 2, \hat{y} * 3, \hat{y} * 4)$, makes the final prediction $\hat{y} = H(\hat{y} * 1, \hat{y} * 2, \hat{y} * 3, \hat{y} * 4)$.

Fig. 1 Stacking takes several classification models called base learners and uses their output as the input for the meta-classifier.

We can train all the models using all the data but this causes over-fitting. To get a better idea of how the algorithm works we use K-fold Cross-validation. We use K-1 folds to train the base classifiers and the last fold to train the meta classifier. We repeat the process using different combinations of each fold. This is shown in Fig 2 where the color-coded square represents the different runs and folds. Each row represents a different run of K fold cross-validation, each column is one of K folds where K=3. For each column, we use the same color columns to train the classifiers and the different color is used to train the meta classifier.

Fig. 2 K-fold Cross-validation to train Stacking classifier.

Table of contents

```
<ol>
    <li><a>Apply Staking Using Wine Data </a></li>
    <li><a href="https://practice/?utm_medium=Exinfluencer&utm_source=Exinfluencer&utm_content=
    <li><a>Cancer Data Example</a></li>
</ol>
```

First, let’s install and import the required libraries:

[1]:
```
# All Libraries required for this lab are listed below. The libraries,␣
↪pre-installed on Skills Network Labs are commented.
# !mamba install -qy pandas==1.3.3 numpy==1.21.2 ipywidgets==7.4.2 scipy==7.4.2␣
↪tqdm==4.62.3 matplotlib==3.5.0 seaborn==0.9.0

# You will need scikit-learn>=0.22.0 as StackingClassifier does not exist in␣
↪version <0.22.0
!mamba install -c conda-forge -qy scikit-learn=0.22.0

# Note: If your environment doesn't support "!mamba install", use "!pip install"
```

```
  Package                                    Version  Build                   Channel
Size


  Install:



  + atk-1.0                                   2.36.0  h3371d22_4
conda-forge/linux-64     574kB
  + cached-property                            1.5.2  hd8ed1ab_1
conda-forge/noarch         4kB
  + cached_property                            1.5.2  pyha770c72_1
conda-forge/noarch        11kB
  + font-ttf-dejavu-sans-mono                   2.37  hab24e00_0
conda-forge/noarch       397kB
  + font-ttf-inconsolata                       3.000  h77eed37_0
conda-forge/noarch        97kB
  + font-ttf-source-code-pro                   2.038  h77eed37_0
conda-forge/noarch       701kB
  + font-ttf-ubuntu                             0.83  hab24e00_0
conda-forge/noarch         2MB
  + fonts-conda-ecosystem                          1  0
conda-forge/noarch         4kB
  + fonts-conda-forge                              1  0
conda-forge/noarch         4kB
  + gdk-pixbuf                                2.42.8  hff1cb4f_0
conda-forge/linux-64     609kB
  + gtk2                                     2.24.33  h90689f9_2
conda-forge/linux-64       8MB
  + gts                                        0.7.6  h64030ff_2
conda-forge/linux-64     421kB
  + jbig                                         2.1  h7f98852_2003
conda-forge/linux-64      44kB
  + joblib                                     1.1.0  pyhd8ed1ab_0
conda-forge/noarch       215kB
```

```
  + lerc                                      3.0  h9c3ff4c_0
conda-forge/linux-64     222kB
  + libdeflate                               1.10  h7f98852_0
conda-forge/linux-64      79kB
  + libgd                                   2.3.3  h283352f_2
conda-forge/linux-64     278kB
  + libllvm13                              13.0.1  hf817b99_2
conda-forge/linux-64      35MB
  + librsvg                                2.52.5  h0a9e6e8_2
conda-forge/linux-64       6MB
  + libwebp-base                            1.2.2  h7f98852_1
conda-forge/linux-64     844kB
  + scikit-learn                             0.22  py37hcdab131_1
conda-forge/linux-64       7MB

  Change:

  - cairo                                  1.16.0  h6cf1ce9_1008
installed
  + cairo                                  1.16.0  ha12eb4b_1010
conda-forge/linux-64       2MB
  - krb5                                   1.19.3  h3790be6_0
installed
  + krb5                                   1.19.3  h08a2579_0
conda-forge/linux-64       2MB
  - leptonica                              1.78.0  h42ed529_2
installed
  + leptonica                              1.78.0  ha0c4403_3
conda-forge/linux-64       3MB
  - libevent                               2.1.10  h9b69904_4
installed
  + libevent                               2.1.10  h28343ad_4
conda-forge/linux-64       1MB
  - libnghttp2                             1.47.0  h727a467_0
installed
  + libnghttp2                             1.47.0  he49606f_0
conda-forge/linux-64     844kB
  - libssh2                                1.10.0  ha56f1ee_2
installed
  + libssh2                                1.10.0  ha35d2d1_2
conda-forge/linux-64     238kB
  - libxml2                                2.9.12  h72842e0_0
installed
  + libxml2                                2.9.12  h885dcf4_1
conda-forge/linux-64     778kB
  - python                                 3.7.12  hb7a2778_100_cpython
```

```
installed
  + python                                  3.7.12  hf930737_100_cpython
conda-forge/linux-64      60MB
  - qt                                      5.12.9  hda022c4_4
installed
  + qt                                      5.12.9  h1304e3e_6
conda-forge/linux-64     103MB
  - tesseract                                4.1.1  h9862bf9_2
installed
  + tesseract                                4.1.1  h58164bb_3
conda-forge/linux-64     325MB

  Upgrade:



  - cryptography                            36.0.2  py37h38fbfac_1
installed
  + cryptography                            37.0.1  py37h9ce1e76_0
pkgs/main/linux-64         1MB
  - ffmpeg                                   4.1.3  h167e202_0
installed
  + ffmpeg                                   4.3.2  h37c90e5_3
conda-forge/linux-64      10MB
  - graphviz                                2.40.1  h0511662_2
installed
  + graphviz                                 3.0.0  h5abf519_1
conda-forge/linux-64       3MB
  - grpcio                                  1.45.0  py37he500948_0
installed
  + grpcio                                  1.46.1  py37h0327239_0
conda-forge/linux-64       3MB
  - gst-plugins-base                        1.18.5  hf529b03_3
installed
  + gst-plugins-base                        1.20.1  hcf0ee16_1
conda-forge/linux-64       3MB
  - gstreamer                               1.18.5  h9f60fe5_3
installed
  + gstreamer                               1.20.2  hd4edc92_1
conda-forge/linux-64       2MB
  - h5py                                    2.10.0  nompi_py37h513d04c_102
installed
  + h5py                                     3.6.0  nompi_py37hd308b1e_100
conda-forge/linux-64       1MB
  - harfbuzz                                 2.9.1  h83ec7ef_1
installed
  + harfbuzz                                 3.4.0  hb4a5f5f_0
conda-forge/linux-64       2MB
```

```
  - hdf5                                      1.10.5  nompi_h5b725eb_1114
installed
  + hdf5                                      1.12.1  nompi_h4df4325_104
conda-forge/linux-64       4MB
  - icu                                         68.2  h9c3ff4c_0
installed
  + icu                                         69.1  h9c3ff4c_0
conda-forge/linux-64      14MB
  - libarchive                                 3.5.1  h3f442fb_1
installed
  + libarchive                                 3.5.2  hed592e5_1
conda-forge/linux-64       2MB
  - libclang                                  11.1.0  default_ha53f305_1
installed
  + libclang                                  13.0.1  default_hc23dcda_0
conda-forge/linux-64      12MB
  - libcurl                                   7.82.0  h7bff187_0
installed
  + libcurl                                   7.83.1  h2283fc2_0
conda-forge/linux-64     352kB
  - libopencv                                  3.4.9  py37_2
installed
  + libopencv                                  4.5.3  py37hb6cea29_4
conda-forge/linux-64      36MB
  - libpq                                       13.5  hd57d9b9_1
installed
  + libpq                                       14.3  he2d8382_0
conda-forge/linux-64       3MB
  - libtiff                                    4.1.0  hc3755c2_3
installed
  + libtiff                                    4.3.0  h542a066_3
conda-forge/linux-64     653kB
  - libwebp                                    1.0.2  h56121f0_5
installed
  + libwebp                                    1.2.2  h3452ae3_0
conda-forge/linux-64      87kB
  - mysql-common                              8.0.25  ha770c72_0
installed
  + mysql-common                              8.0.29  h26416b9_1
conda-forge/linux-64       2MB
  - mysql-libs                                8.0.25  h935591d_0
installed
  + mysql-libs                                8.0.29  hbc51c84_1
conda-forge/linux-64       2MB
  - opencv                                     3.4.9  py37_2
installed
  + opencv                                     4.5.3  py37h89c1867_4
conda-forge/linux-64      22kB
```

```
  - openh264                           1.8.0  hdbcaa40_1000
installed
  + openh264                           2.1.1  h780b84a_0
conda-forge/linux-64       2MB
  - openssl                           1.1.1n  h166bdaf_0
installed
  + openssl                            3.0.3  h166bdaf_0
conda-forge/linux-64       3MB
  - pango                             1.42.4  h69149e4_5
installed
  + pango                             1.50.5  h4dcc4a0_1
conda-forge/linux-64     466kB
  - py-opencv                          3.4.9  py37h5ca1d4c_2
installed
  + py-opencv                          4.5.3  py37h6531663_4
conda-forge/linux-64       1MB
  - x264                      1!152.20180806  h14c3975_0
installed
  + x264                          1!161.3030  h7f98852_1
conda-forge/linux-64       3MB
  - zstd                               1.4.9  ha95c52a_0
installed
  + zstd                               1.5.2  h8a70e8d_1
conda-forge/linux-64     462kB

  Downgrade:



  - libprotobuf                       3.20.0  h6239696_0
installed
  + libprotobuf                       3.18.1  h780b84a_0
conda-forge/linux-64       3MB
  - protobuf                          3.20.0  py37hd23a5d3_4
installed
  + protobuf                          3.18.1  py37hcd2ae1e_0
conda-forge/linux-64     353kB

  Summary:

  Install: 21 packages
  Change: 10 packages
  Upgrade: 26 packages
  Downgrade: 2 packages

  Total download: 673MB
```

```
Preparing transaction: …working… done
Verifying transaction: …working… done
Executing transaction: …working…

done
```

[19]:
```
import pandas as pd
# import pylab as plt
import numpy as np
import scipy.optimize as opt
from sklearn import preprocessing
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import metrics
from sklearn.model_selection import GridSearchCV
import seaborn as sns
from sklearn import preprocessing
from sklearn.ensemble import StackingClassifier
from sklearn.svm import SVC
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import LogisticRegression
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
/tmp/ipykernel_1275/82920220.py in <module>
     10 import seaborn as sns
     11 from sklearn import preprocessing
---> 12 from sklearn.ensemble import StackingClassifier
     13 from sklearn.svm import SVC
     14 from sklearn.neighbors import KNeighborsClassifier

~/conda/envs/python/lib/python3.7/site-packages/sklearn/ensemble/__init__.py in
 ↪<module>
      5
      6 from ._base import BaseEnsemble
----> 7 from ._forest import RandomForestClassifier
      8 from ._forest import RandomForestRegressor
      9 from ._forest import RandomTreesEmbedding

~/conda/envs/python/lib/python3.7/site-packages/sklearn/ensemble/_forest.py in␣
 ↪<module>
     54 from ..metrics import r2_score
     55 from ..preprocessing import OneHotEncoder
```

```
---> 56 from ..tree import (DecisionTreeClassifier, DecisionTreeRegressor,
     57                     ExtraTreeClassifier, ExtraTreeRegressor)
     58 from ..tree._tree import DTYPE, DOUBLE

~/conda/envs/python/lib/python3.7/site-packages/sklearn/tree/__init__.py in␣
 ↪<module>
      4 """
      5
----> 6 from ._classes import BaseDecisionTree
      7 from ._classes import DecisionTreeClassifier
      8 from ._classes import DecisionTreeRegressor

~/conda/envs/python/lib/python3.7/site-packages/sklearn/tree/_classes.py in␣
 ↪<module>
     38 from ..utils.validation import check_is_fitted
     39
---> 40 from ._criterion import Criterion
     41 from ._splitter import Splitter
     42 from ._tree import DepthFirstTreeBuilder

~/conda/envs/python/lib/python3.7/site-packages/sklearn/tree/_criterion.
 ↪cpython-37m-x86_64-linux-gnu.so in init sklearn.tree._criterion()

AttributeError: type object 'sklearn.tree._criterion.array' has no attribute␣
 ↪'__reduce_cython__'
```

Ignore error warnings

[3]:
```
import warnings
warnings.filterwarnings('ignore')
```

This function will calculate the accuracy of the training and testing data given a model.

[4]:
```
def get_accuracy(X_train, X_test, y_train, y_test, model):
    return  {"test Accuracy":metrics.accuracy_score(y_test, model.
 ↪predict(X_test)),"trian Accuracy": metrics.accuracy_score(y_train, model.
 ↪predict(X_train))}
```

Apply Staking Using Wine Data

The class is an essential factor in determining the quality of the wine; this dataset uses chemical analysis of wines grown in the same region in Italy from three different cultivars. Your task is to determine the class of the wine using the features from the chemical analysis. The features or attributes include

For more info here ,let's load the dataset:

[5]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
↪cloud/IBM-ML241EN-SkillsNetwork/labs/datasets/wine.data",names= ['Class',␣
↪'Alcohol', 'Malic acid', 'Ash',
         'Alcalinity of ash' ,'Magnesium', 'Total phenols',
         'Flavanoids', 'Nonflavanoid phenols', 'Proanthocyanins',      'Color␣
↪intensity', 'Hue', 'OD280/OD315 of diluted wines',
         'Proline'])

df.head()
```

[5]:
```
   Class  Alcohol  Malic acid   Ash  Alcalinity of ash  Magnesium  \
0      1    14.23        1.71  2.43               15.6        127
1      1    13.20        1.78  2.14               11.2        100
2      1    13.16        2.36  2.67               18.6        101
3      1    14.37        1.95  2.50               16.8        113
4      1    13.24        2.59  2.87               21.0        118

   Total phenols  Flavanoids  Nonflavanoid phenols  Proanthocyanins  \
0           2.80        3.06                  0.28             2.29
1           2.65        2.76                  0.26             1.28
2           2.80        3.24                  0.30             2.81
3           3.85        3.49                  0.24             2.18
4           2.80        2.69                  0.39             1.82

   Color intensity   Hue  OD280/OD315 of diluted wines  Proline
0             5.64  1.04                          3.92     1065
1             4.38  1.05                          3.40     1050
2             5.68  1.03                          3.17     1185
3             7.80  0.86                          3.45     1480
4             4.32  1.04                          2.93      735
```

We see all the dataset is comprised of numerical values using the method dtypes

[6]:
```
df.dtypes
```

[6]:
```
Class                             int64
Alcohol                         float64
Malic acid                      float64
Ash                             float64
Alcalinity of ash               float64
Magnesium                         int64
Total phenols                   float64
Flavanoids                      float64
Nonflavanoid phenols            float64
Proanthocyanins                 float64
Color intensity                 float64
Hue                             float64
```

```
OD280/OD315 of diluted wines    float64
Proline                           int64
dtype: object
```

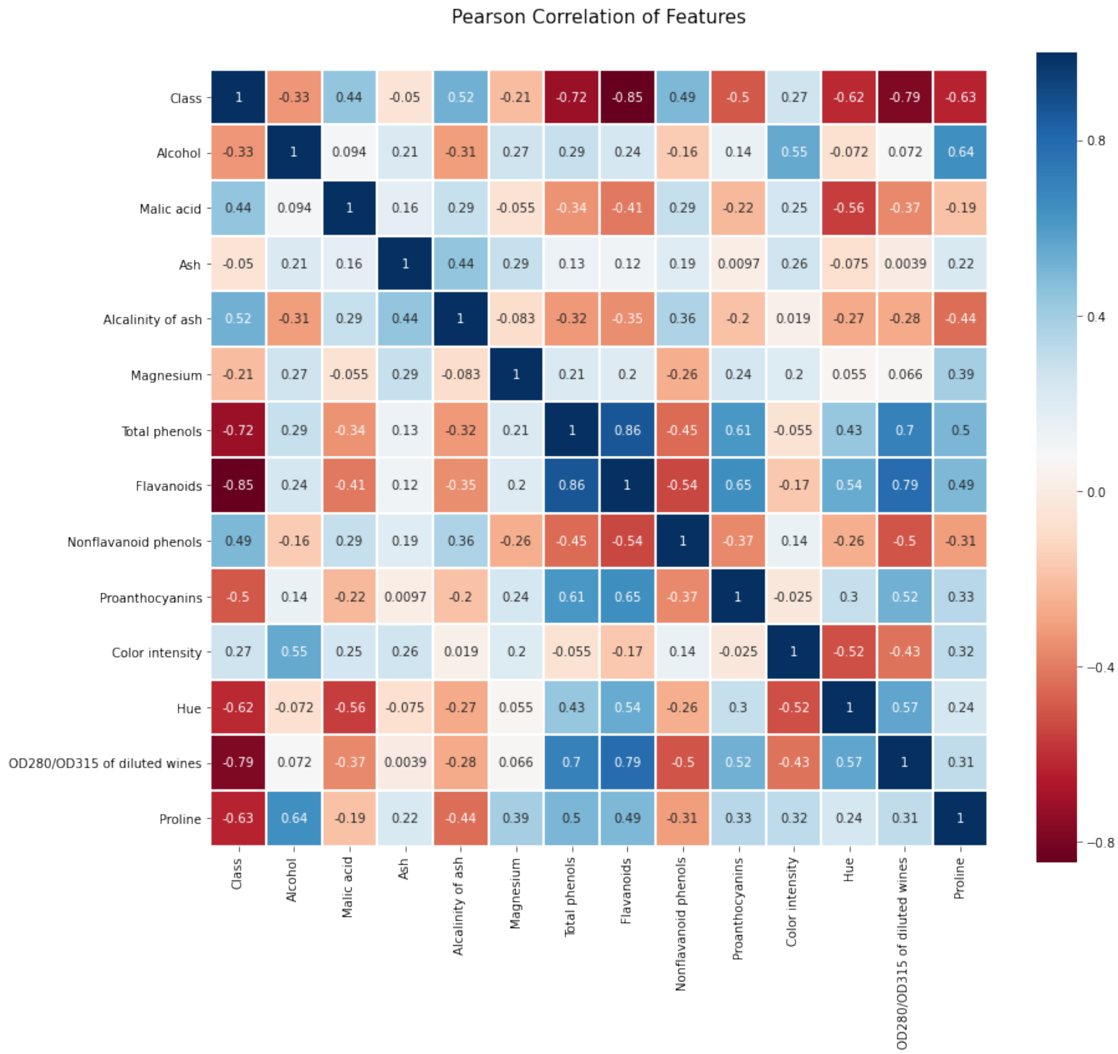
the column class has the class of the wine, we can use the method unique() to obtain the classes:

[7]:
```
df['Class'].unique()
```

[7]:
```
array([1, 2, 3])
```

We can examine the correlation between each feature and the class variable. By examining the first row or column we see the features are correlated with the class variable.

[8]:
```
colormap = plt.cm.RdBu
plt.figure(figsize=(14,12))
plt.title('Pearson Correlation of Features', y=1.05, size=15)
sns.heatmap(df.astype(float).corr(),linewidths=0.1,vmax=1.0,
            square=True, cmap=colormap, linecolor='white', annot=True)
plt.show()
```



We can also examine the Pairplot between pairs of features and the histogram; color-coded to each class. We see the separation between different classes:

[9]:
```
# May need to specify bandwidth (bw) in order to plot, else can ignore.
sns.pairplot(df, hue="Class") #, diag_kws={'bw': 0.2})
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
/tmp/ipykernel_1275/2512947022.py in <module>
      1 # May need to specify bandwidth (bw) in order to plot, else can ignore.
----> 2 sns.pairplot(df, hue="Class") #, diag_kws={'bw': 0.2})

~/conda/envs/python/lib/python3.7/site-packages/seaborn/axisgrid.py in
  ↪pairplot(data, hue, hue_order, palette, vars, x_vars, y_vars, kind, diag_kind
  ↪markers, height, aspect, dropna, plot_kws, diag_kws, grid_kws, size)
```

```
   2109             diag_kws.setdefault("shade", True)
   2110             diag_kws["legend"] = False
-> 2111             grid.map_diag(kdeplot, **diag_kws)
   2112 
   2113     # Maybe plot on the off-diagonals

~/conda/envs/python/lib/python3.7/site-packages/seaborn/axisgrid.py in␣
 ↪map_diag(self, func, **kwargs)
   1397                     color = fixed_color
   1398 
-> 1399                 func(data_k, label=label_k, color=color, **kwargs)
   1400 
   1401             self._clean_axis(ax)

~/conda/envs/python/lib/python3.7/site-packages/seaborn/distributions.py in␣
 ↪kdeplot(data, data2, shade, vertical, kernel, bw, gridsize, cut, clip, legend␣
 ↪cumulative, shade_lowest, cbar, cbar_ax, cbar_kws, ax, **kwargs)
    689         ax = _univariate_kdeplot(data, shade, vertical, kernel, bw,
    690                                  gridsize, cut, clip, legend, ax,
--> 691                                  cumulative=cumulative, **kwargs)
 
    692 
    693     return ax

~/conda/envs/python/lib/python3.7/site-packages/seaborn/distributions.py in␣
 ↪_univariate_kdeplot(data, shade, vertical, kernel, bw, gridsize, cut, clip,␣
 ↪legend, ax, cumulative, **kwargs)
    281         x, y = _statsmodels_univariate_kde(data, kernel, bw,
    282                                            gridsize, cut, clip,
--> 283                                            cumulative=cumulative)
 
    284     else:
    285         # Fall back to scipy if missing statsmodels

~/conda/envs/python/lib/python3.7/site-packages/seaborn/distributions.py in␣
 ↪_statsmodels_univariate_kde(data, kernel, bw, gridsize, cut, clip, cumulative
    353     fft = kernel == "gau"
    354     kde = smnp.KDEUnivariate(data)
--> 355     kde.fit(kernel, bw, fft, gridsize=gridsize, cut=cut, clip=clip)
    356     if cumulative:
    357         grid, y = kde.support, kde.cdf

~/conda/envs/python/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py␣
 ↪in fit(self, kernel, bw, fft, weights, gridsize, adjust, cut, clip)
    173                 gridsize=gridsize,
    174                 clip=clip,
--> 175                 cut=cut,
    176             )
    177         else:
```

```
~/conda/envs/python/lib/python3.7/site-packages/statsmodels/nonparametric/kde.py␣
↪in kdensityfft(x, kernel, bw, weights, gridsize, adjust, clip, cut, retgrid)
    555     elif isinstance(bw, str):
    556         # if bw is None, select optimal bandwidth for kernel
--> 557         bw = bandwidths.select_bandwidth(x, bw, kern)
    558         # will cross-val fit this pattern?
    559     else:

~/conda/envs/python/lib/python3.7/site-packages/statsmodels/nonparametric/
↪bandwidths.py in select_bandwidth(x, bw, kernel)
    180                 "Either provide the bandwidth during initialization or use␣
↪" \
    181                 "an alternative method."
--> 182         raise RuntimeError(err)
    183     else:
    184         return bandwidth

RuntimeError: Selected KDE bandwidth is 0. Cannot estimate density. Either␣
↪provide the bandwidth during initialization or use an alternative method.
```

1.1.1 Data Pre-Processing and Selection

Let's examine the feature list

[11]:
```
features=list(df)
features[1:]
```

[11]:
```
['Alcohol',
 'Malic acid',
 'Ash',
 'Alcalinity of ash',
 'Magnesium',
 'Total phenols',
```

```
'Flavanoids',
'Nonflavanoid phenols',
'Proanthocyanins',
'Color intensity',
'Hue',
'OD280/OD315 of diluted wines',
'Proline']
```

We assign the class variables to y and feature variables to X

[12]:
```
y,X=df[features[0]] ,df[features[1:]]
```

We can standardize the data

[13]:
```
scaler = preprocessing.StandardScaler().fit(X)
X= scaler.transform(X)
```

We can check if the data is standardized by checking the mean and standard deviation, which are approximately zero:

[14]:
```
X.mean(axis=0)
```

[14]:
```
array([-8.38280756e-16, -1.19754394e-16, -8.37033314e-16, -3.99181312e-17,
       -3.99181312e-17,  0.00000000e+00, -3.99181312e-16,  3.59263181e-16,
       -1.19754394e-16,  2.49488320e-17,  1.99590656e-16,  3.19345050e-16,
       -1.59672525e-16])
```

[15]:
```
X.std(axis=0)
```

[15]:
```
array([1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1., 1.])
```

In Random Forest, we would use these data subsets to train each node of a tree.

1.1.2 Train/Test dataset

We split our dataset into train and test set:

[16]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3,␣
↪random_state=1)
print ('Train set', X_train.shape,  y_train.shape)
print ('Test set', X_test.shape,  y_test.shape)
```

```
Train set (124, 13) (124,)
Test set (54, 13) (54,)
```

Stacking consists of creating a Stacking Classifier object, but first, you require a dictionary of estimators (individual model objects or base learners). The key of the dictionary is a name that is up to you, we use the usual acronym for the model. The value is the model object in this case SVC for Support Vector Classifier, dt for Decision Tree Classifier and KNN for K Neighbors Classifier.

[17]:
```
estimators =␣
↪[('SVM',SVC(random_state=42)),('KNN',KNeighborsClassifier()),('dt',DecisionTreeClassifier()
estimators
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1275/196601360.py in <module>
----> 1 estimators =␣
 ↪[('SVM',SVC(random_state=42)),('KNN',KNeighborsClassifier()),('dt',DecisionTreeClassifier(
      2 estimators

NameError: name 'SVC' is not defined
```

To train the final model we create a Stacking Classifier, this combines the base estimators using the meta estimator. The meta-classifier is determined by the parameter final_estimator in this case we use Logistic Regression, we also input the base classifiers using the estimators parameter and fit the model.

[20]:
```
clf = StackingClassifier( estimators=estimators, final_estimator=␣
↪LogisticRegression())
clf.fit(X_train, y_train)
clf
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1275/2258820003.py in <module>
----> 1 clf = StackingClassifier( estimators=estimators, final_estimator=␣
 ↪LogisticRegression())
      2 clf.fit(X_train, y_train)
      3 clf

NameError: name 'StackingClassifier' is not defined
```

We can make a prediction

[ ]:
```
yhat=clf.predict(X_test)
yhat
```

We can obtain the training and testing accuracy, we see the model performs well.

[ ]:
```
get_accuracy(X_train, X_test, y_train, y_test, clf)
```

Note: Like most complex models Stacking is prone to overfitting

Practice

Create a Stacking Classifier object as before but exchange the Decision Tree Classifier with the SVM classifier. Calculate the accuracy on the training and testing data.

[ ]:

Click here for the solution

```
estimators = [('SVM',SVC(random_state=42)),('KNN',KNeighborsClassifier()),('lr',LogisticRegres
clf = StackingClassifier( estimators=estimators, final_estimator= DecisionTreeClassifier())
clf.fit(X_train, y_train)


get_accuracy(X_train, X_test, y_train, y_test, clf)
```

GridSearchCV and Stacking Classifiers

Imagine that you are a medical researcher compiling data for a study. You have collected data about a set of patients, all of whom suffered from the same illness. During their course of treatment, each patient responded to one of 5 medications, Drug A, Drug B, Drug c, Drug x and y.

Part of your job is to build a model to find out which drug might be appropriate for a future patient with the same illness. The features of this dataset are Age, Sex, Blood Pressure, and the Cholesterol of the patients, and the target is the drug that each patient responded to.

It is a sample of multiclass classifier, and you can use the training part of the dataset to build a decision tree, and then use it to predict the class of a unknown patient, or to prescribe a drug to a new patient. You will use GridSearchCV and Stacking Classifiers to find the best results.

[21]:
```
df = pd.read_csv("https://cf-courses-data.s3.us.cloud-object-storage.appdomain.
↪cloud/IBMDeveloperSkillsNetwork-ML0101EN-SkillsNetwork/labs/Module%203/data/
↪drug200.csv", delimiter=",")
df.head()
```

[21]:

	Age	Sex	BP	Cholesterol	Na_to_K	Drug
0	23	F	HIGH	HIGH	25.355	drugY
1	47	M	LOW	HIGH	13.093	drugC
2	47	M	LOW	HIGH	10.114	drugC
3	28	F	NORMAL	HIGH	7.798	drugX
4	61	F	LOW	HIGH	18.043	drugY

Let's create the X and y for our dataset:

[22]:
```
X = df[['Age', 'Sex', 'BP', 'Cholesterol', 'Na_to_K']].values
X[0:5]
```

[22]:
```
array([[23, 'F', 'HIGH', 'HIGH', 25.355],
       [47, 'M', 'LOW', 'HIGH', 13.093],
       [47, 'M', 'LOW', 'HIGH', 10.114],
       [28, 'F', 'NORMAL', 'HIGH', 7.798],
       [61, 'F', 'LOW', 'HIGH', 18.043]], dtype=object)
```

[23]:
```
y = df["Drug"]
y[0:5]
```

[23]:
```
0    drugY
1    drugC
2    drugC
3    drugX
4    drugY
Name: Drug, dtype: object
```

Now lets use a LabelEncoder to turn categorical features into numerical:

[24]:
```python
from sklearn import preprocessing
le_sex = preprocessing.LabelEncoder()
le_sex.fit(['F','M'])
X[:,1] = le_sex.transform(X[:,1])


le_BP = preprocessing.LabelEncoder()
le_BP.fit([ 'LOW', 'NORMAL', 'HIGH'])
X[:,2] = le_BP.transform(X[:,2])


le_Chol = preprocessing.LabelEncoder()
le_Chol.fit([ 'NORMAL', 'HIGH'])
X[:,3] = le_Chol.transform(X[:,3])

X[0:5]
```

[24]:
```
array([[23, 0, 0, 0, 25.355],
       [47, 1, 1, 0, 13.093],
       [47, 1, 1, 0, 10.114],
       [28, 0, 2, 0, 7.798],
       [61, 0, 1, 0, 18.043]], dtype=object)
```

[25]:
```python
scaler = preprocessing.StandardScaler().fit(X)
X= scaler.transform(X)
```

Split the data into training and testing data with a 80/20 split

[26]:
```python
X_train, X_test, y_train, y_test = train_test_split( X, y, test_size=0.3,␣
↪random_state=4)
print ('Train set:', X_train.shape,  y_train.shape)
print ('Test set:', X_test.shape,  y_test.shape)
```

```
Train set: (140, 5) (140,)
Test set: (60, 5) (60,)
```

We have our dictionary of estimators, the individual model objects or base learners.

```
[27]: estimators =␣
       ↪[('SVM',SVC(random_state=42)),('knn',KNeighborsClassifier()),('dt',DecisionTreeClassifier()
      estimators
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
/tmp/ipykernel_1275/1530583448.py in <module>
----> 1 estimators =␣
 ↪[('SVM',SVC(random_state=42)),('knn',KNeighborsClassifier()),('dt',DecisionTreeClassifier(
      2 estimators

NameError: name 'SVC' is not defined
```

We create a Stacking Classifier:

```
[ ]: clf = StackingClassifier( estimators=estimators, final_estimator=␣
      ↪LogisticRegression())
     clf
```

In order to alter the base models in the dictionary of hyperparameter values, we add the key value of each model followed by the parameter of the model we would like to vary.

```
[ ]: param_grid = {'dt__max_depth': [n for n in range(10)],'dt__random_state':
      ↪[0],'SVM__C':[0.01,0.1,1],'SVM__kernel':['linear', 'poly',␣
      ↪'rbf'],'knn__n_neighbors':[1,4,8,9] }
```

We use GridSearchCV to search over specified parameter values of the model.

```
[ ]: search = GridSearchCV(estimator=clf, param_grid=param_grid,scoring='accuracy')
     search.fit(X_train, y_train)
```

We can find the accuracy of the best model.

```
[ ]: search.best_score_
```

We can find the best parameter values:

```
[ ]: search.best_params_
```

We can find the accuracy test data:

```
[ ]: get_accuracy(X_train, X_test, y_train, y_test, search)
```

```
[ ]: # We use sklearn version 0.20.1 for all other labs, please run this command␣
      ↪after finishing the lab

     !mamba install -c conda-forge -qy scikit-learn=0.20.1
```

1.1.3 Thank you for completing this lab!

1.2 Author

Joseph Santarcangelo

1.2.1 Other Contributors

1.3 Change Log

Date (YYYY-MM-DD)	Version	Changed By	Change Description
2021-01-01	1.0	Joseph S	Created the initial version
2022-02-09	1.1	Steve Hord	QA pass

##