Московский государственный технический университет им. Н.Э. Баумана
Кафедра «Системы обработки информации и управления»

Лабораторная работа №6
по дисциплине
«Методы машинного обучения»
на тему
«Ансамбли моделей машинного обучения»

Выполнил:
студент группы ИУ5-24М
Мельников К. И.

Москва — 2019 г.

1. Ансамбли моделей машинного обучения.

Цель лабораторной работы: изучение ансамблей моделей машинного обучения.
Задание:

```
Выберите набор данных (датасет) для решения задачи классификации или регресии.
В случае необходимости проведите удаление или заполнение пропусков и кодирование к
С использованием метода train_test_split разделите выборку на обучающую и тестовую
Обучите две ансамблевые модели. Оцените качество моделей с помощью одной из подход
Произведите для каждой модели подбор значений одного гиперпараметра. В зависимости
Повторите пункт 4 для найденных оптимальных значений гиперпараметров. Сравните кач
```

2. Ход работы

```
In [1]: import numpy as np

        from PIL  import Image

        from sklearn.decomposition import PCA
        from sklearn.preprocessing import scale
        from sklearn import metrics
        from sklearn.cluster import KMeans

        from IPython.display import display

        from tqdm import tqdm_notebook as tqdm

        import matplotlib.pyplot as plt
        import matplotlib.cm as cm

        from scipy import ndimage as ndi
        from skimage.morphology import medial_axis

        from scipy import ndimage as ndi
        from skimage.morphology import medial_axis
        from skimage.morphology import skeletonize

        from scipy.spatial import Delaunay

        from sklearn.model_selection import train_test_split
```

```
In [2]: data = np.load("hiragana.npz")['arr_0']
```

```
In [5]: X = []
        Y = []
        for index,letter in enumerate(data):
            for variant in letter:
                X.append(variant)
                Y.append(index)
```

```
In [38]: def moment(array,m1,m2):
             moment = 0
             for y,ver in enumerate(array):
                 for x,hor in enumerate(ver):
                     moment += pow(x,m1)*pow(y,m2)*hor
             return moment

         def center(array):
             x = moment(array,1,0)/moment(array,0,0)
             y = moment(array,0,1)/moment(array,0,0)
             return (x,y)

         def translate(array,x,y):
             buffer = np.roll(array,-x,axis=1)
             buffer = np.roll(buffer,-y,axis=0)
             return buffer

         def centeredarray(array):
             buffer = []
             for pic in tqdm(array):
                 shape = pic.shape
                 centroid = center(pic)
                 delta_x = -shape[1]/2 + centroid[0]
                 delta_y = -shape[0]/2 + centroid[1]

                 buffer += [translate(pic,int(delta_x),int(delta_y))]
             return buffer

In [39]: test = np.array(centeredarray(X[:640]))
```

```
HBox(children=(IntProgress(value=0, max=640), HTML(value='')))
```

```
In [3]: dataL = np.load("hirag.npz")['arr_0']

In [6]: data = np.array(dataL[:1599])
        y = np.array(Y[:1599])

In [7]: datax = data.reshape(data.shape[0], data.shape[1]*data.shape[2])

In [8]: X_train, X_test, y_train, y_test = train_test_split(datax, y, test_size=0

In [9]: from sklearn.ensemble import RandomForestClassifier

In [47]: rfc = RandomForestClassifier(n_estimators=100)

In [48]: rfcf = rfc.fit(X_train, y_train)
In [12]: from sklearn.metrics import accuracy_score
         from sklearn.metrics import roc_auc_score
         from sklearn.metrics import average_precision_score
         from sklearn.preprocessing import OneHotEncoder

In [49]: predictions = rfcf.predict(X_test)

In [50]: accuracy_score(y_test, predictions)

Out[50]: 0.571969696969697

In [15]: from sklearn.ensemble import AdaBoostClassifier

In [53]: abc = AdaBoostClassifier(n_estimators=50, random_state=0)

In [54]: abcf = abc.fit(X_train, y_train)

In [55]: predictions = abcf.predict(X_test)

In [56]: accuracy_score(y_test, predictions)

Out[56]: 0.2215909090909091

In [20]: from sklearn.ensemble import GradientBoostingClassifier

In [41]: gbc = GradientBoostingClassifier(n_estimators=20, learning_rate=0.5, max_

In [42]: gbcf = gbc.fit(X_train, y_train)

In [43]: predictions = gbcf.predict(X_test)

In [44]: accuracy_score(y_test, predictions)

Out[44]: 0.4015151515151515

In [45]: from sklearn.model_selection import GridSearchCV

In [46]: parameters = {'max_depth':[2,5,10,20]}

In [51]: rfcgs = GridSearchCV(rfc, parameters, cv=5, scoring='accuracy').fit(X_tra

/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/model_selection/_sea
  DeprecationWarning)


In [52]: rfcgs.cv_results_

/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
```

```
Out[52]: {'mean_fit_time': array([0.22353926, 0.40677371, 0.66166739, 0.79426775]
          'std_fit_time': array([0.00234956, 0.01266633, 0.00536465, 0.00558671])
          'mean_score_time': array([0.01896033, 0.02015023, 0.01271248, 0.0117696
          'std_score_time': array([0.00074107, 0.0006931 , 0.00437626, 0.00099181
          'param_max_depth': masked_array(data=[2, 5, 10, 20],
                       mask=[False, False, False, False],
                 fill_value='?',
                      dtype=object),
          'params': [{'max_depth': 2},
           {'max_depth': 5},
           {'max_depth': 10},
           {'max_depth': 20}],
          'split0_test_score': array([0.45      , 0.53181818, 0.58636364, 0.59090
          'split1_test_score': array([0.46046512, 0.56744186, 0.58604651, 0.60930
          'split2_test_score': array([0.48837209, 0.5627907 , 0.55348837, 0.54418
          'split3_test_score': array([0.49056604, 0.56132075, 0.5754717 , 0.59433
          'split4_test_score': array([0.48803828, 0.55023923, 0.54066986, 0.51196
          'mean_test_score': array([0.47525677, 0.55462185, 0.56862745, 0.5704948
          'std_test_score': array([0.01696354, 0.0128778 , 0.01825765, 0.03616655
          'rank_test_score': array([4, 3, 2, 1], dtype=int32),
          'split0_train_score': array([0.53819036, 0.78143361, 0.98589894, 1.
          'split1_train_score': array([0.54439252, 0.76168224, 0.99299065, 1.
          'split2_train_score': array([0.5817757 , 0.77920561, 0.99415888, 1.
          'split3_train_score': array([0.56344587, 0.78579744, 0.99301513, 1.
          'split4_train_score': array([0.55336427, 0.774942  , 0.99767981, 1.
          'mean_train_score': array([0.55623375, 0.77661218, 0.99274868, 1.
          'std_train_score': array([0.01535199, 0.00824662, 0.00383023, 0.
```

```
In [60]: parameters = {'learning_rate':[0.05,0.1,0.2]}
```

```
In [61]: adabgs = GridSearchCV(abc, parameters, cv=5, scoring='accuracy').fit(X_t
```

```
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/model_selection/_sea
  DeprecationWarning)
```

```
In [62]: adabgs.cv_results_
```

```
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
/home/hexagramg/exp/venv/lib/python3.6/site-packages/sklearn/utils/deprecation.py
  warnings.warn(*warn_args, **warn_kwargs)
```

```
Out[62]: {'mean_fit_time': array([5.29969072, 5.24841089, 5.24106441]),
          'std_fit_time': array([0.05036853, 0.02111449, 0.02217165]),
          'mean_score_time': array([0.04807606, 0.0492311 , 0.04860482]),
          'std_score_time': array([0.00088288, 0.00092484, 0.0005285 ]),
          'param_learning_rate': masked_array(data=[0.05, 0.1, 0.2],
                       mask=[False, False, False],
                 fill_value='?',
                      dtype=object),
          'params': [{'learning_rate': 0.05},
           {'learning_rate': 0.1},
           {'learning_rate': 0.2}],
          'split0_test_score': array([0.34545455, 0.38636364, 0.35      ]),
          'split1_test_score': array([0.39534884, 0.33488372, 0.29767442]),
          'split2_test_score': array([0.33488372, 0.29302326, 0.30697674]),
          'split3_test_score': array([0.36792453, 0.3254717 , 0.26886792]),
          'split4_test_score': array([0.32057416, 0.32057416, 0.27751196]),
          'mean_test_score': array([0.35294118, 0.33239963, 0.30065359]),
          'std_test_score': array([0.02620828, 0.03078116, 0.02853955]),
          'rank_test_score': array([1, 2, 3], dtype=int32),
          'split0_train_score': array([0.41363102, 0.44653349, 0.39012926]),
          'split1_train_score': array([0.39018692, 0.38434579, 0.35163551]),
          'split2_train_score': array([0.41588785, 0.37149533, 0.41705607]),
          'split3_train_score': array([0.39813737, 0.37252619, 0.37136205]),
          'split4_train_score': array([0.4199536 , 0.38283063, 0.37470998]),
          'mean_train_score': array([0.40755935, 0.39154629, 0.38097857]),
          'std_train_score': array([0.01140545, 0.02798288, 0.02180876])}
```