Отчет

Вступление

В 2014 году Мануэлем Фернандез-Дельгадо была написана статья «Do we Need Hundreds of Classifiers to Solve Real World Classification Problems?», в которой сравнивалась эффективность работы различных классификаторов на большом множестве задач.

Сейчас перед нами поставлена цель: повторить данный опыт с меньшим количеством задач и классификаторов.

Задание №1: Вопросы по статье

1. Критерий выбора задач.

Из 165 задач, находящихся в репозитории UCI были исключены 57 по следующим причинам: 25 со слишком большим количеством признаков и/или объектов, 27 не соответствовали общему формату UCI, 5 по различным причинам (один вход, классы без объектов, классы с одним объектом и т.п.). Добавлены 4 задачи из реальной жизни. Некоторые задачи делились на несколько задач, например, задача о *Кардиотокографии,* где входные данные могли принадлежать 3 или 10 классам.
Таким образом, всего получилась 121 задача.

2. Критерий выбора методов.

Было выбрано 179 классификаторов, происходящих из 17 «семейств»., то есть происхождения, например, некоторые классификаторы пришли из статистики, другие – из символьного искусственного интеллекта и data mining, и т.д. Требовалось сравнить все классификаторы и выявить те, которые работали лучше всего на всех разных задачах, поэтому в особенности необходимо было выбирать классификаторы из разных семейств, поскольку большинство из уже имеющихся сравнений охватывает лишь сравнения классификаторов одного семейства.

3. Методы валидации.

Для настройки параметров тестовая и обучающая выборки генерировались рандомно, но таким образом, чтобы каждый класс имел поровну тестовых и обучающих объектов. После чего подбирался параметр, дающий наилучшие результаты на этой тестовой выборке. Затем следовало 4-fold cross validation, и обучение и тестирование производились исходя из выборок, полученных в результате кроссвалидации.

4. Обработка невычислений.

Невычисления и падения программ исключались из рассчета средней ошибки.

Задание №2: Выбор задач и методов

1. Выбранные задачи

Мною выбраны 4 задачи классификации, в основном из реальной жизни: классификация зерен пшеницы; выявление того, может ли человек носить контактные линзы; классификация животных по их признакам; распознавание цифр.

Далее представлена информация о выбранных наборах данных, взятая с сайта https://archive.ics.uci.edu/ml/datasets.html

Seeds Data Set

Data Set Information:

The examined group comprised kernels belonging to three different varieties of wheat: Kama, Rosa and Canadian, 70 elements each, randomly selected for

the experiment. High quality visualization of the internal kernel structure was detected using a soft X-ray technique. It is non-destructive and considerably cheaper than other more sophisticated imaging techniques like scanning microscopy or laser technology. The images were recorded on 13x18 cm X-ray KODAK plates. Studies were conducted using combine harvested wheat grain originating from experimental fields, explored at the Institute of Agrophysics of the Polish Academy of Sciences in Lublin.

The data set can be used for the tasks of classification and cluster analysis.

Attribute Information:

To construct the data, seven geometric parameters of wheat kernels were measured:

1. area A,

2. perimeter P,

3. compactness C = 4*pi*A/P^2,

4. length of kernel,

5. width of kernel,

6. asymmetry coefficient

7. length of kernel groove.

All of these parameters were real-valued continuous

Lenses Data Set:

The examples are complete and noise free. The examples highly simplified the problem. The attributes do not fully describe all the factors affecting the decision as to which type, if any, to fit.

Notes:

--This database is complete (all possible combinations of attribute-value pairs are represented).

--Each instance is complete and correct.

--9 rules cover the training set.

Attribute Information:

-- 3 Classes

1 : the patient should be fitted with hard contact lenses,

2 : the patient should be fitted with soft contact lenses,

3 : the patient should not be fitted with contact lenses.

1. age of the patient: (1) young, (2) pre-presbyopic, (3) presbyopic

2. spectacle prescription: (1) myope, (2) hypermetrope

3. astigmatic: (1) no, (2) yes

4. tear production rate: (1) reduced, (2) normal

Zoo Data Set:

A simple database containing 17 Boolean-valued attributes. The "type" attribute appears to be the class attribute. Here is a breakdown of which animals are in which type:

Class# -- Set of animals:

====== ==========

1 -- (41) aardvark, antelope, bear, boar, buffalo, calf, cavy, cheetah, deer, dolphin, elephant, fruitbat, giraffe, girl, goat, gorilla, hamster, hare, leopard, lion, lynx, mink, mole, mongoose, opossum, oryx, platypus, polecat, pony, porpoise, puma, pussycat, raccoon, reindeer, seal, sealion, squirrel, vampire, vole, wallaby,wolf

2 -- (20) chicken, crow, dove, duck, flamingo, gull, hawk, kiwi, lark, ostrich, parakeet, penguin, pheasant, rhea, skimmer, skua, sparrow, swan, vulture, wren

3 -- (5) pitviper, seasnake, slowworm, tortoise, tuatara

4 -- (13) bass, carp, catfish, chub, dogfish, haddock, herring, pike, piranha, seahorse, sole, stingray, tuna

5 -- (4) frog, frog, newt, toad

6 -- (8) flea, gnat, honeybee, housefly, ladybird, moth, termite, wasp

7 -- (10) clam, crab, crayfish, lobster, octopus, scorpion, seawasp, slug, starfish, worm

Pen-Based Recognition of Handwritten Digits Data Set:

Data Set Information:

We create a digit database by collecting 250 samples from 44 writers. The samples written by 30 writers are used for training, cross-validation and writer dependent testing, and the digits written by the other 14 are used for writer independent testing. This database is also available in the UNIPEN format.

These writers are asked to write 250 digits in random order inside boxes of 500 by 500 tablet pixel resolution.
In our study, we use only (x, y) coordinate information. The stylus pressure level values are ignored. First we apply normalization to make our representation invariant to translations and scale distortions. The raw data that we capture from the tablet consist of integer values between 0 and 500 (tablet input box resolution). The new coordinates are such that the coordinate which has the maximum range varies between 0 and 100. Usually x stays in this range, since most characters are taller than they are wide.

In order to train and test our classifiers, we need to represent digits as constant length feature vectors. A commonly used technique leading to good results is resampling the (x_t, y_t) points. Temporal resampling (points regularly spaced in time) or spatial resampling (points regularly spaced in arc length) can be used here. Raw point data are already regularly spaced in time but the distance between them is variable. Our resampling algorithm uses simple linear interpolation between pairs of points. The resampled digits are represented as a sequence of T points (x_t, y_t)_{t=1}^T, regularly spaced in arc length, as opposed to the input sequence, which is regularly spaced in time.

So, the input vector size is 2*T, two times the number of points resampled. We considered spatial resampling to T=8,12,16 points in our experiments and found that T=8 gave the best trade-off between accuracy and complexity.

Attribute Information:

All input attributes are integers in the range 0..100.

The last attribute is the class code 0..9

2. Выбранные методы

1) Random Forest classifier, из библиотеки Scikit-learn для Python — алгоритм «Случайного леса».

2) SGDClassifier, так же из Scikit-learn для Python — линейный классификатор (SVM, метод опорных векторов), обучающийся с помощью метода стохастического градиентного спуска.

3) Perceptron из Scikit-learn для Python. Линейный классификатор, разработанный Фрэнком Розенблаттом.

2.1. Выдержки кода использования методов

1) Random Forest Classifier.

```
from sklearn.ensemble import RandomForestClassifier
from numpy import genfromtxt

path = "/home/ed/Dropbox/HSE_2016/problems/"
path_answer = "/home/ed/Документы/Classifier Results/"
name = input()
path += name
path += '/'

def main():
    train = genfromtxt(path + "train_input.csv", delimiter=',')
    target = genfromtxt(path + "train_output.csv", delimiter=',')
    test = genfromtxt(path + "test_input.csv", delimiter=',')
    label = genfromtxt(path + "test_output.csv", delimiter=',')

    sgd = RandomForestClassifier()
    sgd.fit(train, target)
    result = sgd.score(test, label)
    print(result)
    number = input()
    f = open(path_answer + "RF.csv", "a")
    f.write(number + "," + str(result) + "\n")
    f.close()

if __name__=="__main__":
    main()
```

(Сперва вводится имя папки на Dropbox, содержащей обучающую выборку, далее часть имени, общая для файлов, находящихся в ней, остальное очевидно: составление обучающей и тестирующей выборок, генерация дерева, вычисление процента точности, запись результата, а так же вводится и записывается ID задачи)

2) Linear classifier with SGD

```
from sklearn.linear_model import SGDClassifier
from numpy import genfromtxt

path = "/home/ed/Dropbox/HSE_2016/problems/"
path_answer = "/home/ed/Документы/Classifier Results/"
name = input()
path += name
path += '/'
name = input()

def main():
    train = genfromtxt(path + "train_input.csv", delimiter=',')
    target = genfromtxt(path + "train_output.csv", delimiter=',')
    test = genfromtxt(path + "test_input.csv", delimiter=',')
    label = genfromtxt(path + "test_output.csv", delimiter=',')
```

```
    sgd = SGDClassifier()
    sgd.fit(train, target)
    result = sgd.score(test, label)
    print(result)
    number = input()
    f = open(path_answer + "SGD.csv", "a")
    f.write(number + "," + str(result) + "\n")
    f.close()

if __name__=="__main__":
    main()
```

(Аналогично, вводится название папки и затем часть имени файлов, общая для всех файлов в этой папке. Вводится так же ID задачи, результаты сохраняются в специальный файл для дальнейшей обработки)

3) Perceptron

```
from sklearn.linear_model import Perceptron
from numpy import genfromtxt

path = "/home/ed/Dropbox/HSE_2016/problems/"
path_answer = "/home/ed/Документы/Classifier Results/"
name = input()
path += name
path += '/'
name = input()

def main():
    train = genfromtxt(path + name + "_train_input.csv", delimiter=',')
    target = genfromtxt(path + name + "_train_output.csv", delimiter=',')
    test = genfromtxt(path + name + "_test_input.csv", delimiter=',')
    label = genfromtxt(path + name + "_test_output.csv", delimiter=',')

    per = Perceptron()
    per.fit(train, target)
    result = per.score(test, label)
    print(result)
    number = input()
    f = open(path_answer + "perceptron.csv", "a")
    f.write(number + "," + str(result) + "\n")
    f.close()

if __name__=="__main__":
    main()
```

(все аналогично)

Задание №3: Сравнение эффективности всех методов

Ниже представлены таблицы, содержащие результаты работы всех алгоритмов на всех методах. В первом столбце перечислены ID задач, а в первой строке названия классификаторов, а в ячейках указана точность, полученная при тестировании, в процентах.

Таблица 1

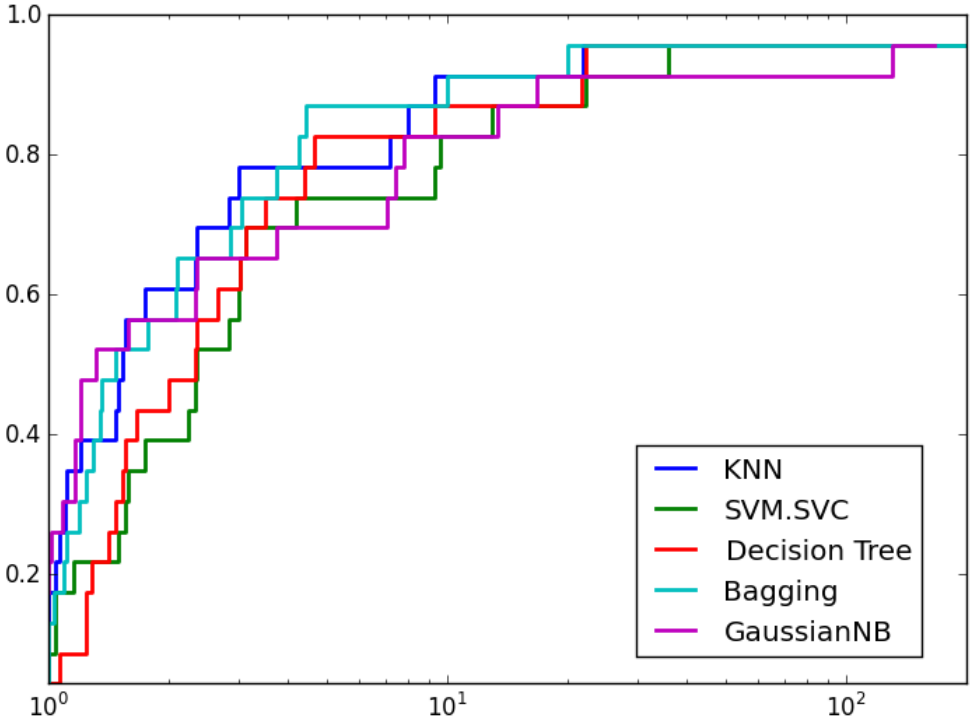
Id	knn	svm.svc	decision_tree	bagging	GaussianNB	ExtraTreesCla	svm.LinearS	GradientBoostir	neighbors.NearestCe
1	97.8	97.78	97.78	98.0	100	97.77	97.77	97.78	93.33
2	75.13	0.0	74.43	76.0	75.67	75.11	66.32	75.66	68.18
3	69.7	69.33	72.0	71.0	74	71.66	68.33	80.33	65.33
4	0	0.0	0.0	0.0	0	0	0	0	0
5	100	100.0	100.0	100.0	98.66	100	100	100	98.67
6	60.66	36.07	86.89	85.0	95.08	95.08	63.93	91.80	70.49
7	37.8	57.78	66.67	68.0	57.77	66.66	44.44	73.33	35.56
8	90	90.0	76.67	86.0	93.33	76.66	90.0	90.0	83.33
9	78	76.76	70.54	74.0	75.93	73.44	31.95	78.84	67.22
10	0	0.0	0.0	0.0	0	0	0	0	0
11	72.59	67.57	71.43	77.0	77.60	72.97	66.79	80.31	59.85
12	68.13	72.53	65.93	64.0	71.43	65.93	26.37	69.23	70.33
13	0	0.0	0.0	81.0	89.30	81.28	87.16	85.56	70.05
14	87.46	83.28	92.33	95.0	70.73	89.2	69.69	98.26	62.37
15	96.88	93.75	96.88	98.0	84.38	92.19	42.19	95.31	87.5
16	74.07	66.67	75.31	80.0	82.72	82.72	85.19	76.54	65.43
17	0	0.0	0.0	0.0	0	0	0	0	0
18	0	0.0	0.0	0.0	0	0	0	0	0
19	0	0.0	0.0	53.0	0	0	0	57.66	47.75
20	51.25	52.38	43.31	50.0	46.71	46.48	45.12	54.42	37.64
21	89.9	78.79	78.79	66.0	78.78	86.86	90.9	88.89	79.8
22	100	100	97.82	99.0	86.89	100	99.03	99.76	74.76
23	0	0.0	0.0	0.0	0	0	0	0	0
24	99.1	97.27	97.27	96.0	93.63	98.18	96.36	99.09	99.09
25	0	0.0	0.0	0.0	0	0	0	0	0
26	0	0.0	0.0	0.0	0	0	0	0	0
27	97.6	13.69	92.51	95.0	82.24	96.19	84.47	96.11	77.76
28	66.7	66.67	83.33	60.0	83.33	100	66.66	66.67	83.33
29	70.97	70.97	80.65	90.0	90.32	77.41	83.87	80.65	77.42
30	0	0.0	0.0	0.0	0	0	0	0	0
31	81.97	81.97	70.49	73.0	76.19	74.60	93.65	70.78	77.78

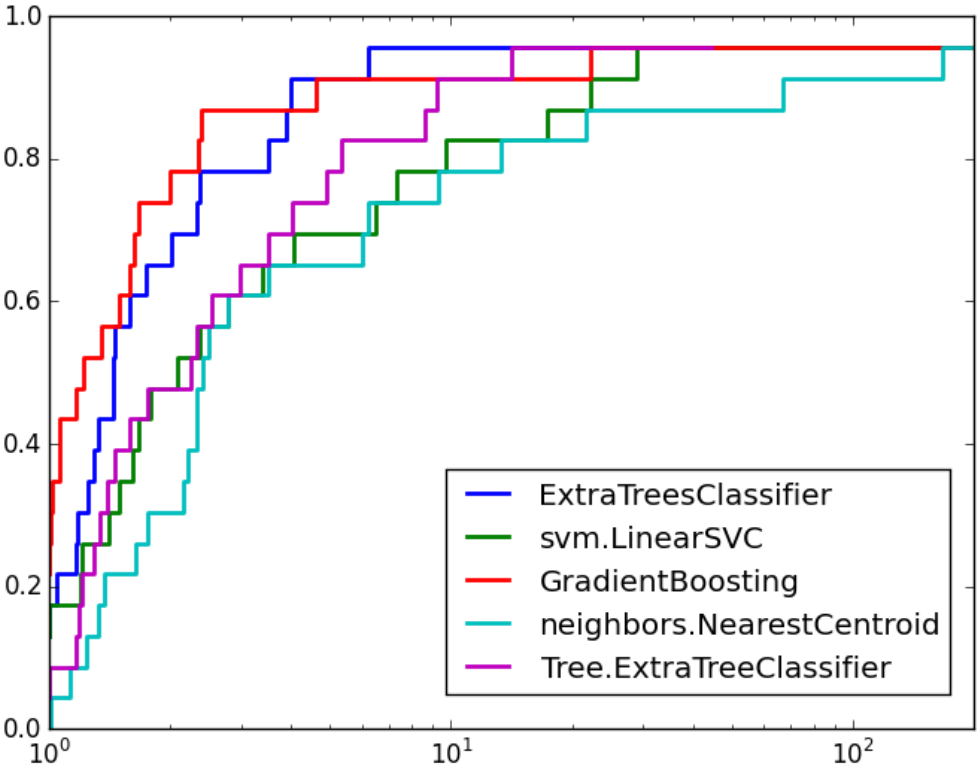
Таблица 2

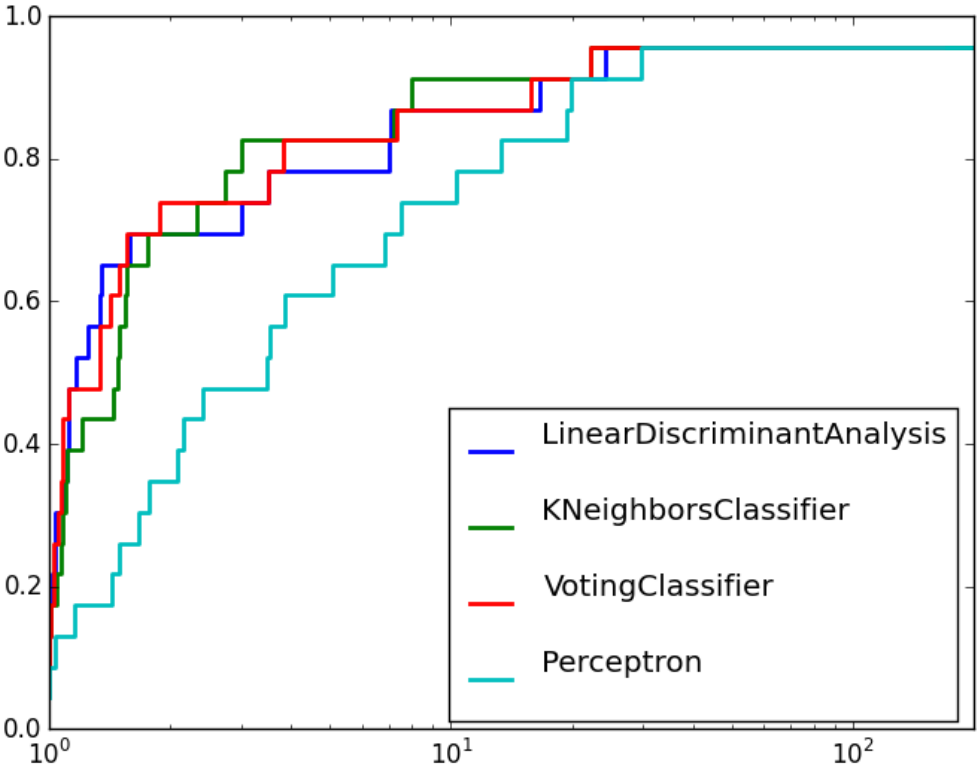
Id	Tree.ExtraTree	LinearDiscrimi	KNeighborsClassifie	VotingClassifier	Perceptron	SGDClassi	RFClass	AdaBoost	RadiusNeighb
1	95.56	100	97.78	97.78	100	75.56	97.78	93.0	98.0
2	71.73	76.09	75.13	75.4	75.23	44.7	75.34	76.0	0.0
3	65.33	79.66	69.66	78.67	31.33	69	73.67	76.0	71.0
4	0	0	0	0	0	0	0	0.0	0.0
5	100	100	100	100	98.67	100	100	100.0	100.0
6	73.77	93.44	60.65	93.44	49.18	47.54	86.89	83.0	62.0
7	68.89	57.77	37.77	62.23	35.56	35.56	62.22	59.0	59.0
8	76.67	76.66	90.0	90.0	90	83.33	86.67	90.0	93.0
9	72.2	77.59	78.01	77.59	22.82	77.59	77.18	80.0	79.0
10	0	0	0	0	0	0	0	0.0	0.0
11	72.97	81.08	72.58	81.47	67.18	66.4	76.83	80.0	73.0
12	65.93	71.43	68.13	71.43	73.63	71.43	68.13	67.0	73.0
13	72.73	85.56	84.49	88.77	77	85.56	81.82	85.0	84.0
14	83.97	71.08	87.45	72.47	65.5	70.03	95.47	78.0	66.0
15	82.81	85.93	96.87	96.88	40.63	68.75	92.19	78.0	98.0
16	76.54	82.71	74.07	85.19	69.14	59.26	79.01	78.0	72.0
17	0	0	0	0	0	0	0	0.0	0.0
18	0	0	0	0	0	0	0	0.0	0.0
19	43.24	56.08	54.28	54.73	51.13	47.3	54.5	42.0	33.0
20	45.12	48.98	51.25	53.29	34.69	45.8	50.79	54.0	52.0
21	72.73	89.9	89.89	82.83	67.68	76.77	84.85	50.0	61.0
22	99.51	97.57	100	99.27	98.06	95.63	99.51	100.0	100.0
23	0	0	0	0	0	0	0	0.0	0.0
24	87.27	99.09	99.09	99.09	95.45	98.19	99.09	47.0	64.0
25	0	0	0	0	0	0	0	0.0	0.0
26	0	0	0	0	0	0	0	0.0	0.0
27	90.34	82.96	97.6	90.79	83.62	85.07	95.6	61.0	83.0
28	100	66.66	66.66	83.34	66.67	66.67	83.3	80.0	100.0
29	77.42	70.96	70.96	87.1	83.87	70.97	77.42	60.0	50.0
30	0	0	0	0	0	0	0	0.0	0.0
31	85.71	92.06	82.54	77.78	52.38	69.84	69.84	79.0	79.0

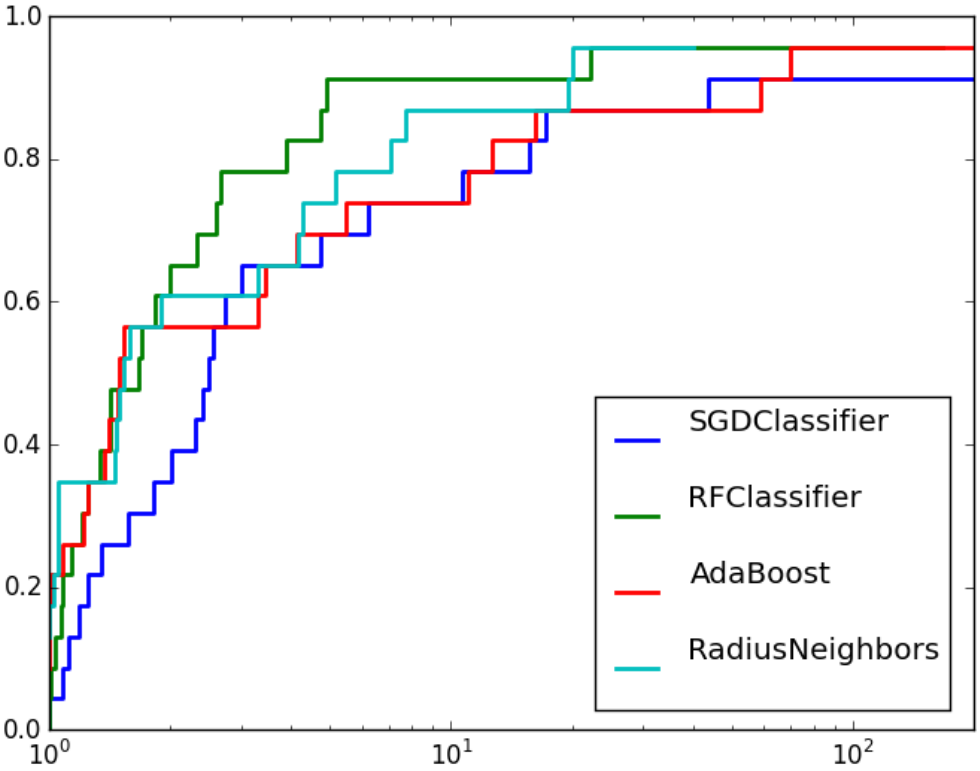
(нули в ячейках таблицы означает, что задачу было решено исключить из списка)

Так же по результатам тестирования были построены кривые Долана-Мора, наглядно демонстрирующие относительную эффективность классификаторов.


1.0
0.8
0.6
0.4
0.2
10^0
10^1
10^2
KNN
SVM.SVC
Decision Tree
Bagging
GaussianNB


1.0
0.8
0.6
0.4
0.2
0.0
10^0
10^1
10^2
ExtraTreesClassifier
svm.LinearSVC
GradientBoosting
neighbors.NearestCentroid
Tree.ExtraTreeClassifier


1.0
0.8
0.6
0.4
0.2
0.0
10^0
10^1
10^2
LinearDiscriminantAnalysis
KNeighborsClassifier
VotingClassifier
Perceptron


1.0
0.8
0.6
0.4
0.2
0.0
10^0
10^1
10^2
SGDClassifier
RFClassifier
AdaBoost
RadiusNeighbors

Задание №4: Улучшение результата на одной задаче

Для улучшения результата я решил выбрать одну из своих задач - Pen-Based Recognition of Handwritten Digits Data Set (ID 27) и классификатор – RandomForestClassifier.

Приблизительный результат его работы до подбора параметров – 95.6% точности.
Для подбора параметров я использовал 25-блочную кросс валидацию.

Код, подбирающий параметры:

```
from sklearn.ensemble import RandomForestClassifier
from numpy import genfromtxt
from sklearn import cross_validation

path = "C:/Pendigits/pendigits_"
train = genfromtxt(path + "train_input.csv", delimiter=',')
target = genfromtxt(path + "train_output.csv", delimiter=',')
test = genfromtxt(path + "test_input.csv", delimiter=',')
label = genfromtxt(path + "test_output.csv", delimiter=',')

max_score = 0
best_parameter = 0
for i in range(10, 200, 10):
    rf = RandomForestClassifier(n_estimators=i)
    rf.fit(train, target)
    score = cross_validation.cross_val_score(rf, train, y=target, cv=25).mean()
    if score > max_score + 0.00009:
        max_score = score
        best_parameter = i
print('n_estimators=', best_parameter)

gini = RandomForestClassifier(n_estimators=best_parameter, criterion='gini')
gini.fit(train, target)
score_gini = cross_validation.cross_val_score(gini, train, y=target, cv=25).mean()

entropy = RandomForestClassifier(n_estimators=best_parameter, criterion='entropy')
entropy.fit(train, target)
score_entropy = cross_validation.cross_val_score(entropy, train, y=target, cv=25).mean()

criterion = ""
if score_gini > score_entropy:
    criterion = "gini"
else:
    criterion = "entropy"

print("criterion=", criterion)

RF = RandomForestClassifier(n_estimators=best_parameter, criterion=criterion)
```

```
RF.fit(train, target)
print(RF.score(test, label))
```

Если результат улучшался менее, чем на 0,00009, то изменения параметра игнорировалось (чтобы избежать добавления лишних деревьев с маленьким результатом). Так же шаг делался размера 10 для уменьшения времени работы программы.

Лучшие значения параметров: n_estimators=150, criterion=”entropy”.

В итоге, результат улучшился примерно до 96,48%.

Выводы:

Благодаря данной практике мне удалось узнать основы машинного обучения, познакомиться с различными методами МО, а также научиться применять их на практике. Я познакомился с различной литературой по данной теме и узнал много нового, что однозначно поможет мне и далее осваивать машинное обучение.