Федеральное государственное автономное образовательное учреждение  
высшего образования

**«Национальный исследовательский университет»**

**«Высшая школа экономики»**

Факультет компьютерных наук

ООП «Прикладная математика и информатика»

**Отчёт о прохождении учебной практики**

|  |  |
| --- | --- |
| **Студент:** | Соловьев Алексей Владимирович |
| **Группа:** | БПМИ-155 |

|  |  |
| --- | --- |
| **Организация: НИУВШЭ, Москва, Кочновский проезд д. 3, Кафедра технологий моделирования сложных систем** |  |

Руководитель: кафедра технологий моделирования сложных систем, старший преподаватель, Янович Юрий Александрович

Москва, 2016

Оглавление

[Введение. 2](#_Toc461054487)

[Задание 1. 3](#_Toc461054488)

[Задание 2. 4](#_Toc461054489)

[Задание 3. 6](#_Toc461054490)

[Задание 4. 9](#_Toc461054491)

[Заключение. 9](#_Toc461054492)

# Введение.

Основные цели:

1. Знакомство с задачей классификации в рамках машинного обучения;  
2. Знакомство с крупнейшим открытым хранилищем реальных задач-примеров для классификации --- репозиторием UCI;  
3. Знакомство с популярными библиотеками для классификации, и математическими методами, лежащими в их основе;  
4. Проведение "соревнования" выбранны методов на избранных задачах.  
  
Примечание. Практика основана на (почти) одноименной статье 2014 года в одном из наиболее престижных журналов по машинному обучению --- JMLR.

Решаемые задачи:

Задание 1.

1.

1.1 зайти на UCI-Repository и выбрать не менее 5 задач;

1.2 занести эти задачи в google-таблицы;

1.3 разбить данные задач на 4 части (train-input, train\_output, test\_input, test\_output) и залить на Dropbox в формате .csv;

2. Вопросы по статье "Do We Need Hundreds of Classifiers to Solve Real World

Classification Problems?"

2.1 Критерии выбора задач из статьи

2.2 Критерии выбора методов

2.3 CV или 70-30?

2.4 Как обрабатывались ситуации падения алгоритма на каких-то конкретных задачах при агрегировании результатов?

3\*. Сделать автоматическое скачивание выборок из UCI.

Задание 2.

1. Обработать свои данные (устранить буквы/строки, знаки вопросов, пропуски в выборках);
2. Выбрать три метода классификации и протестировать их на всех задачах из google-таблицы;

Задание 3. Нарисовать кривые Долан-Мора по всем методам, которые тестировали задачи.

Задание 4. Выбрать задачу. Улучшить для неё точность на каком-нибудь алгоритме.

# Задание 1.

Часть 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Задача | объектов | признаков | Классов | Доля объектов наибольшего класса, % |
| Blood Transfusion | 748 | 4 | 2 | 76,2 |
| Mammographic  Mass | 961 | 5 | 2 | 53,7 |
| Pima Indians Diabetes | 768 | 8 | 2 | 65,1 |
| Fertility | 100 | 9 | 2 | 88 |
| Horse Colic | 368 | 25 | 2 | 63,7 |

Часть 2:

1. Использовался UCI репозиторий. Из 165 имеющихся наборов данных (на март 2013) 57 отбросили (25 слишком больших наборов, 27 не подходили по формату (“common UCI format”) и 5 из-за других причин). Также использовались 4 набора данных не из UCI. Некоторые наборы из UCI были рассмотрены как несколько задач классификации, так как имели несколько колонок класс. Таким образом, получилось 121 набор данных.

2. Использовалось 179 классификаторов, реализованных на C/C++, Matlab, R и Weka. 84 Weka классификаторов из 93 используемых OpenML (9 не смогли включить, так как они принимали только номинальные (не численные) входы, а все входы уже были в числовых значениях). Были использованы 36 классификаторов, реализованных в R, 48 в R с использованием caret package, 6 в C, 5 в Matlab. Были разработаны собственные версии классификаторов на С.

3. Сначала случайным образом генерируется набор тренировки и тестирования (50% для каждого набора). Затем, получившиеся множества используются для настройки параметров, выбирая значения параметров так, чтобы они обеспечивали наивысшую точность на тестовом наборе. Индексы моделей обучения и тестирования (разделение данных) задаются в файле conxuntos.dat для каждого набора данных (одни и те же для всех классификаторов). Затем происходит 4-fold-cross validation с использованием имеющихся данных. В итоге, результатом является среднее значение 4 наборов.

4. Всего было найдено 449 ошибок (2.1% из всех 21659 случаев). Ошибки были исключены из расчета средней точности для каждого классификатора. Для некоторых наборов данных, которые предоставляют отдельные данные для обучения и тестирования, классификатор (с настроенными значениями параметров) обучен и проверен на соответствующих наборах данных. В этом случае, результат теста вычисляется на тестовом наборе. Если ошибки не удавалось исправить, наборы данных отбрасывали.

# Задание 2.

Участники практики в совокупности выбрали 31 задачу. В некоторых из них были некорректные данные (буквы/строки, знаки вопросов, пропуски в выборках). Восемь задач было решено не включать в тестирование. Часть выборок удалось исправить.

Задачи, распределенные на 4 выборки, были скачаны с dropbbox. С помощью библиотеки os и pandas происходила загрузка данных для тестирования алгоритмов.

Тестирование задач классификаторами происходило на языке программирования Python. Был скачан инсталлятор Anaconda, содержащий в себе библиотеки (pandas, numpy, scipy, matplotlib), которые использовались для работы с алгоритмами. Для реализации методов на задачах использовалась библиотека scikit-learn. Было выбрано три алгоритма классификации: LinearDiscriminantAnalysis, KNeighborsClassifier, VotingClassifier.

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **os**

**from** **sklearn.neighbors** **import** KNeighborsClassifier

**from** **sklearn.discriminant\_analysis** **import** LinearDiscriminantAnalysis

**from** **sklearn.linear\_model** **import** LogisticRegression

**from** **sklearn.naive\_bayes** **import** GaussianNB

**from** **sklearn.ensemble** **import** RandomForestClassifier

**from** **sklearn.ensemble** **import** VotingClassifier

names = os.listdir('C:/Users/Aleksey/Documents/problems')

models = [LinearDiscriminantAnalysis(), KNeighborsClassifier()]

data = pd.DataFrame()

**for** name **in** names:

files = os.listdir('C:/Users/Aleksey/Documents/problems/' + name)

test\_input = pd.read\_csv('problems/' + name + '/' + files[0], sep=',', header=**None**)

train\_input = pd.read\_csv('problems/' + name + '/' + files[2], sep=',', header=**None**)

test\_output = pd.read\_csv('problems/' + name + '/' + files[1], sep=',', header=**None**)

train\_output = pd.read\_csv('problems/' + name + '/' + files[3], sep=',', header=**None**)

train\_output = train\_output.values.ravel()

models[0].fit(train\_input, train\_output)

models[1].fit(train\_input, train\_output)

clf1 = LogisticRegression(random\_state=1)

clf2 = RandomForestClassifier(random\_state=1)

clf3 = GaussianNB()

eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)], voting='hard')

eclf1 = eclf1.fit(train\_input, train\_output)

new\_line = {'Name':name, 'LDA':models[0].score(test\_input, test\_output) \* 100,

'KNC':models[1].score(test\_input, test\_output) \* 100,

'VC':eclf1.score(test\_input, test\_output) \* 100}

data = data.append(new\_line, ignore\_index=**True**)

data

Каждый из алгоритмов смог протестировать все корректные задачи.

**Таблица с результатами работы алгоритмов:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **LinearDiscriminantAnalysis** | **KNeighborsClassifier** | **VotingClassifier** |
| 1 | iris | 100 | 97.78 | 97.78 |
| 2 | Bank Marketing | 76.09 | 75.13 | 75.4 |
| 3 | Statlog (German Credit Data) | 79.66 | 69.66 | 78.67 |
| 4 | STUDENT ALCOHOL CONSUMPTION | 0 | 0 | 0 |
| 5 | Qualitative\_Bankruptcy | 100 | 100 | 100 |
| 6 | Wine Data Set | 93.44 | 60.65 | 93.44 |
| 7 | Teaching Assistant Evaluation | 57.77 | 37.77 | 62.23 |
| 8 | Fertility | 76.66 | 90.0 | 90.0 |
| 9 | Blood Transfusion | 77.59 | 78.01 | 77.59 |
| 10 | Mammographic Mass | 0 | 0 | 0 |
| 11 | Pima Indians Diabetes | 81.08 | 72.58 | 81.47 |
| 12 | Haberman's Survival | 71.43 | 68.13 | 71.43 |
| 13 | Balance Scale | 85.56 | 84.49 | 88.77 |
| 14 | Tic-Tac-Toe Endgame | 71.08 | 87.45 | 72.47 |
| 15 | Glass | 85.93 | 96.87 | 96.88 |
| 16 | Statlog (Heart) | 82.71 | 74.07 | 85.19 |
| 17 | Car Evaluation | 0 | 0 | 0 |
| 18 | Nursery Data Set | 0 | 0 | 0 |
| 19 | Yeast Data Set | 56.08 | 54.28 | 54.73 |
| 20 | Contraceptive Method Choice Data Se | 48.98 | 51.25 | 53.29 |
| 21 | Ecoli Data Set | 89.9 | 89.89 | 82.83 |
| 22 | Banknote authentication | 97.57 | 100 | 99.27 |
| 23 | Chronic\_Kidney\_Disease | 0 | 0 | 0 |
| 24 | Dermatology | 99.09 | 99.09 | 99.09 |
| 25 | Horse Colic | 0 | 0 | 0 |
| 26 | Congressional Voting Records | 0 | 0 | 0 |
| 27 | pen-based recognition of handwritten digits | 82.96 | 97.6 | 90.79 |
| 28 | lenses | 66.66 | 66.66 | 83.34 |
| 29 | zoo | 70.96 | 70.96 | 87.1 |
| 30 | hepatitis | 0 | 0 | 0 |
| 31 | seeds | 92.06 | 82.54 | 77.78 |

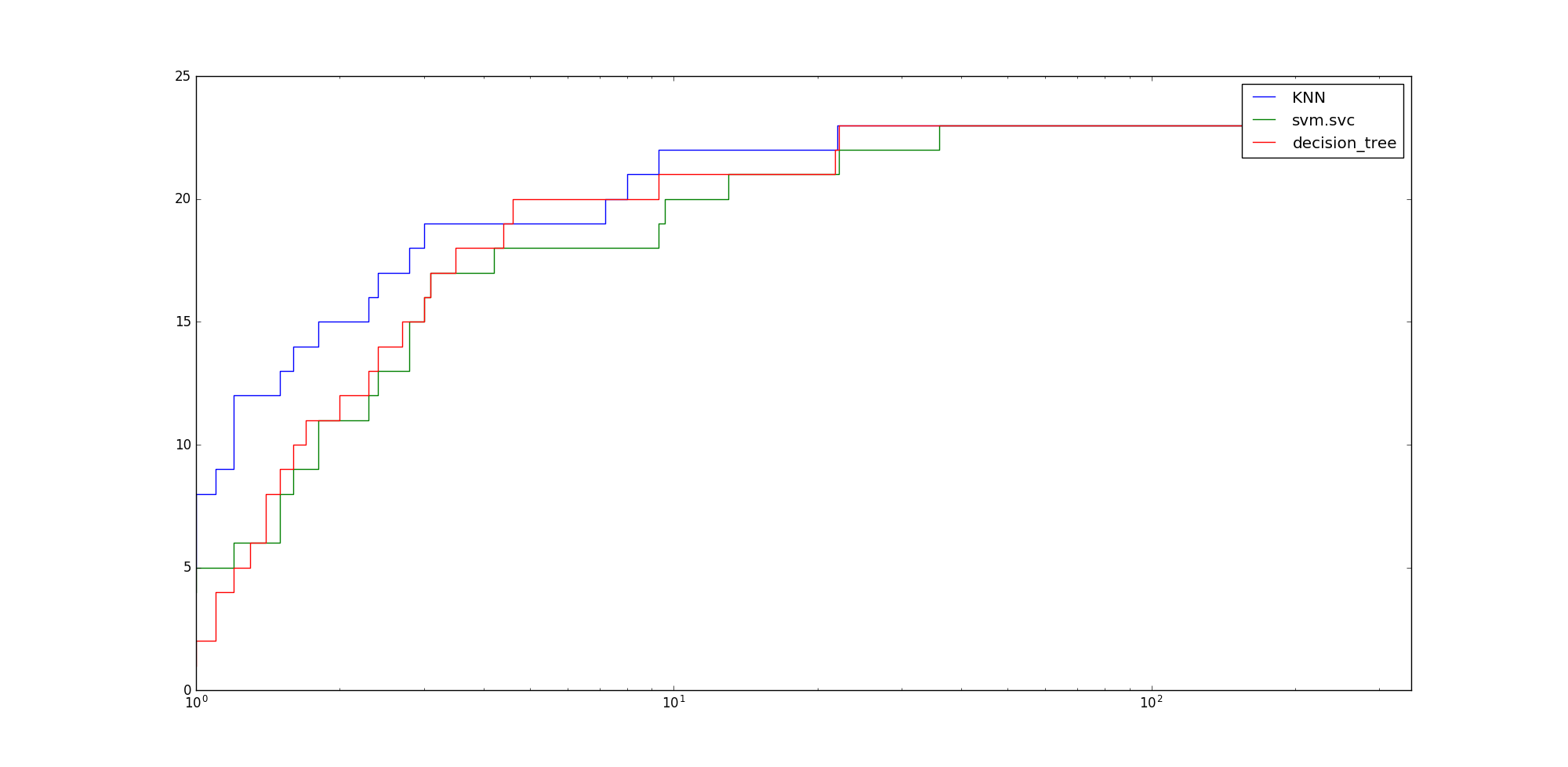
# Задание 3.

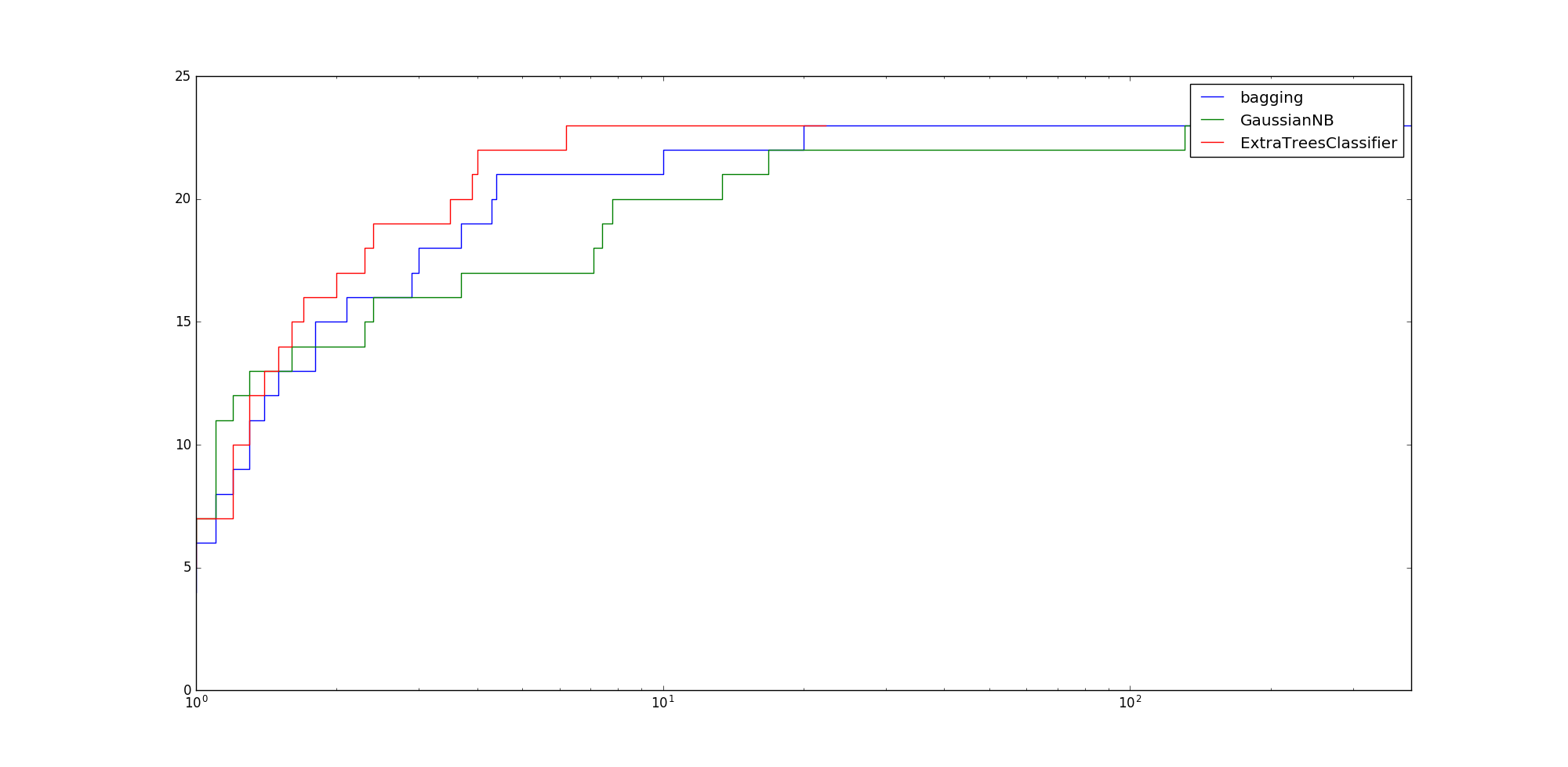
Одним из наиболее удобных инструментов для сравнения результатов работы алгоритмов на большом числе тестовых выборок является графическое представление в виде кривых Долан-Мора.

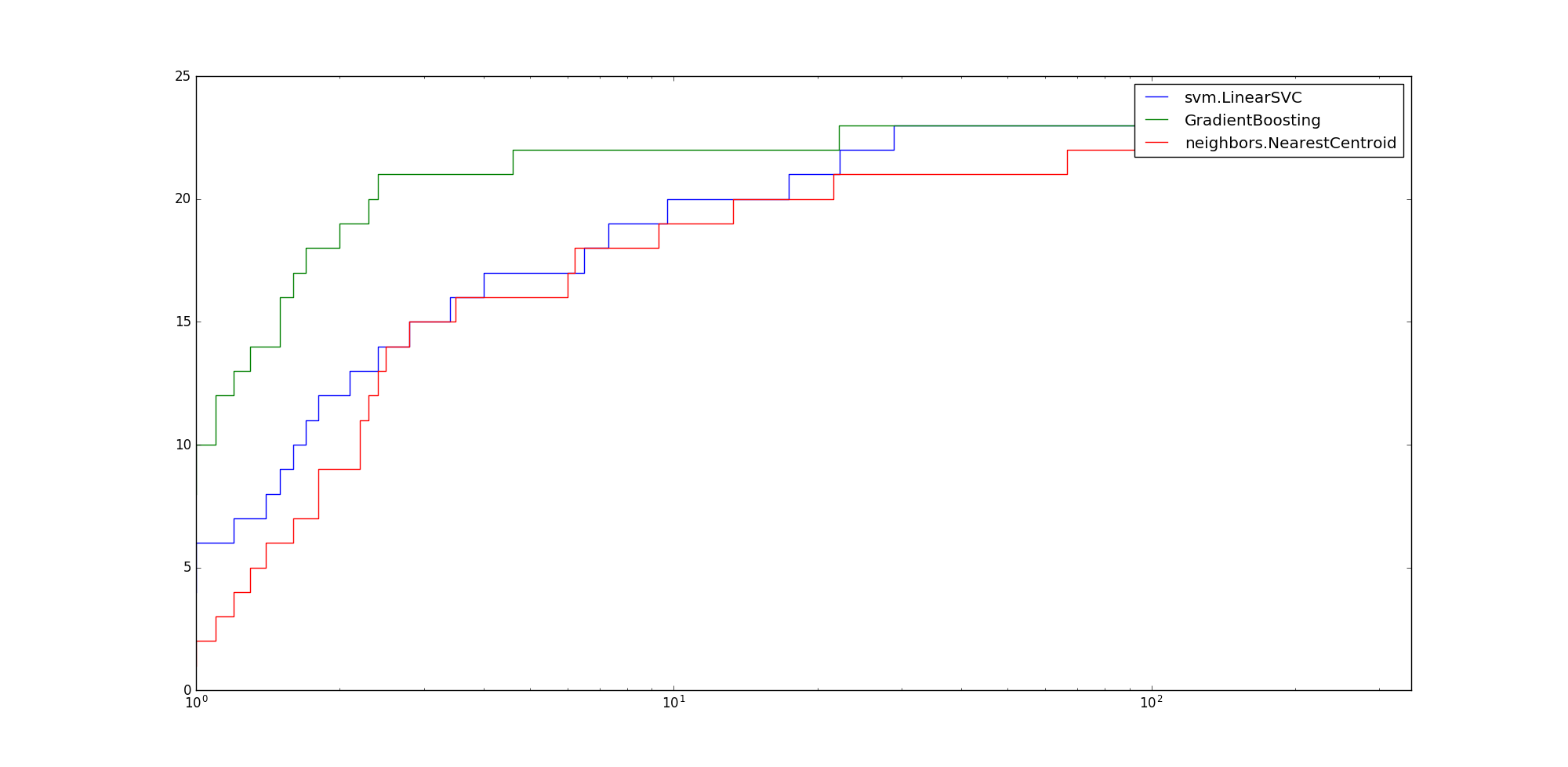
Принцип построения кривых Долан-Мора: по оси абсцисс данных графиков откладывается отношение ошибки рассматриваемого алгоритма(i-ого) к минимальной ошибке среди всех методов на конкретной задаче. По оси ординат откладывается доля задач, на которых ошибка метода не превосходит данного отношения.

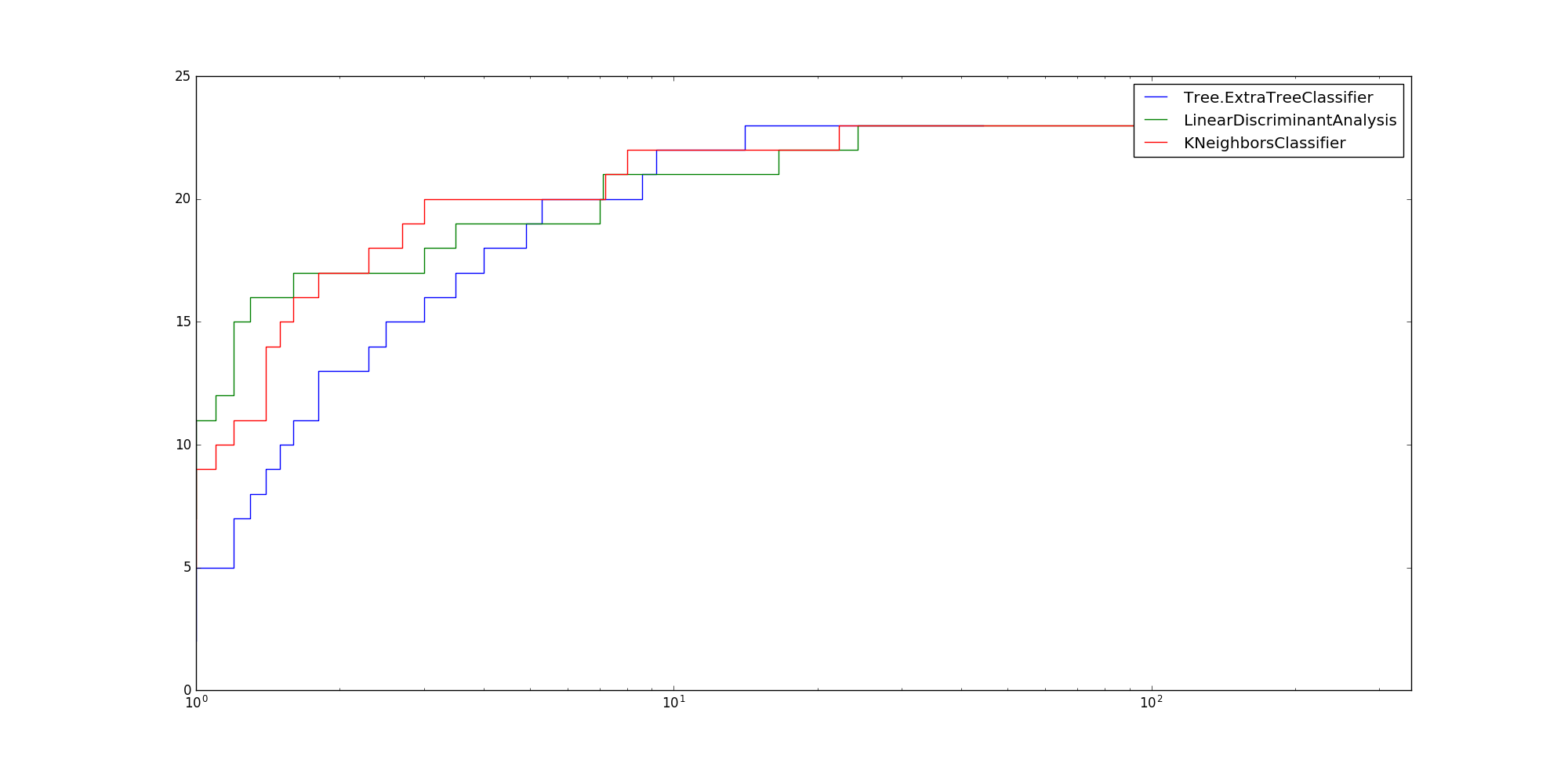
Принцип сравнения кривых: чем выше расположена кривая, тем лучше работает метод, соответствующий данной кривой.

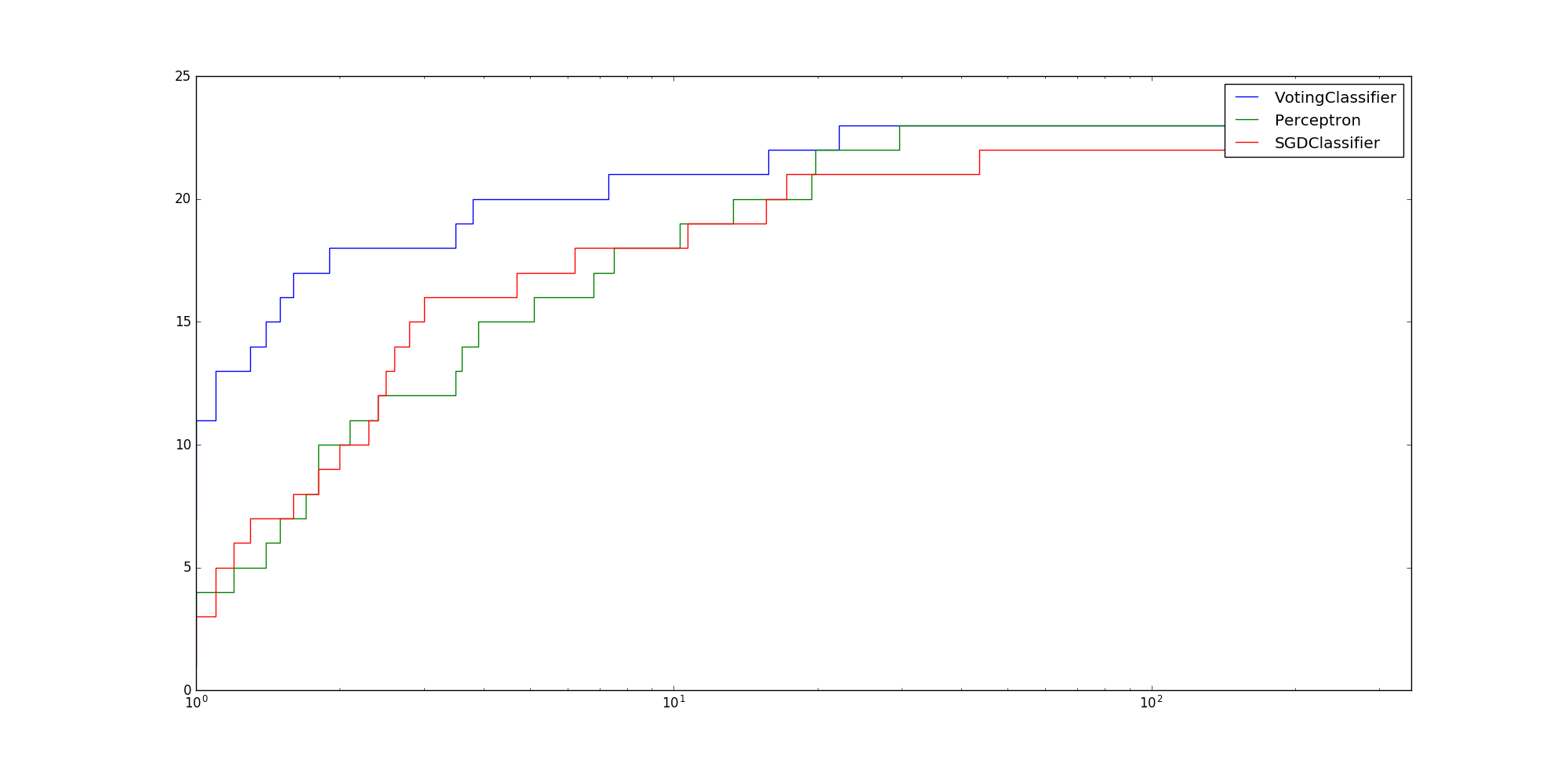
Кривые Долан-Мора для алгоритмов, выбранных участниками практики:

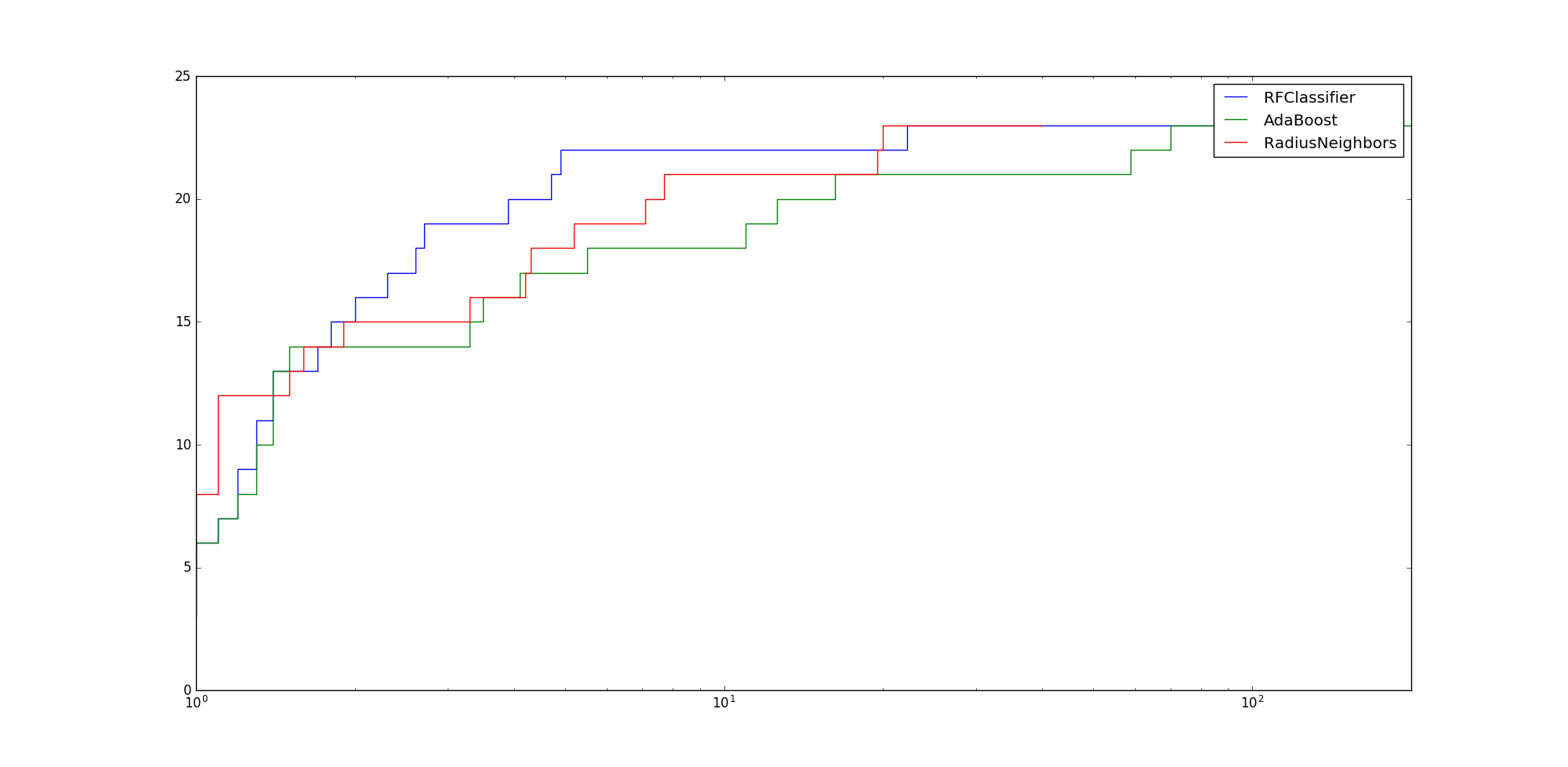












Наилучшими методами оказались VotingClassifier и GradientBoosting.

# Задание 4.

Была выбрана задача “Contraceptive”. Максимальной точность составляла приблизительно 54.42 процента. Для увеличения точности использовался метод Voting Classifier. Так как лучший результат для данной задачи помимо Voting Classifier показали методы: Random Forest Classifier, Ada Boost Classifier, Gradient Boosting Classifier, то для построения модели Voting Classifier использовались данные классификаторы. Также, для оптимизации алгоритма перед запуском были нормализованы входные данные.

**from** **sklearn.ensemble** **import** RandomForestClassifier,AdaBoostClassifier,GradientBoostingClassifier

test\_input = preprocessing.normalize(pd.read\_csv('problems/' + name + '/' + files[0], sep=',', header=**None**))

train\_put = preprocessing.normalize(pd.read\_csv('problems/' + name + '/' + files[1], sep=',', header=**None**))

**…**

clf1 = RandomForestClassifier(random\_state=1)

clf2 = AdaBoostClassifier()

clf3 = GradientBoostingClassifier()

eclf1 = VotingClassifier(estimators=[('lr', clf1), ('rf', clf2), ('gnb', clf3)], voting='hard')

eclf1 = eclf1.fit(train\_input, train\_output)

# Заключение.

Данную практику я оцениваю, как крайне положительный опыт. Я впервые ознакомился с видами задач в рамках машинного обучения. Более подробно узнал о задачах классификации и об основных методах для работы с ними. Также, я ознакомился с крупнейшим открытым хранилищем реальных задач-примеров для классификации --- репозиторием UCI. У меня появилось примерное понимание математических методов, лежащих в основе работы рассматриваемых алгоритмов. Еще я узнал, как с помощью кривых Долан-Мора можно сравнивать работу алгоритмов классификации на выборке задач и попытался улучшить работу алгоритма на задаче. По мере выполнения задания, я узнал про довольно удобный инсталлятор Anaconda, отлично документированную библиотку scikit-learn, содержащую основные методы машинного обучения на Python. Также, я немного разобрался с библиотекой os, чтобы более удобно загружать данные для тестирования задач на методах, и библиотеками urllib, request для автоматического скачивания выборок из UCI. Для использования классификаторов и построения кривых я ознакомился с библиотеками pandas, numpy, scipy и matplotlib.

Мои пожелания к будущей деятельности с позиции полученного опыта – это возможно сделать полноценный проект, в основе которого лежат текущие наработки. Также, я хотел бы более подробно узнать про другие задачи в рамках машинного обучения. Например, задачи регрессии и кластеризации.