Реализация алгоритмов частичного обучения на Apache Spark

Кемаев Юрий, 341 гр.

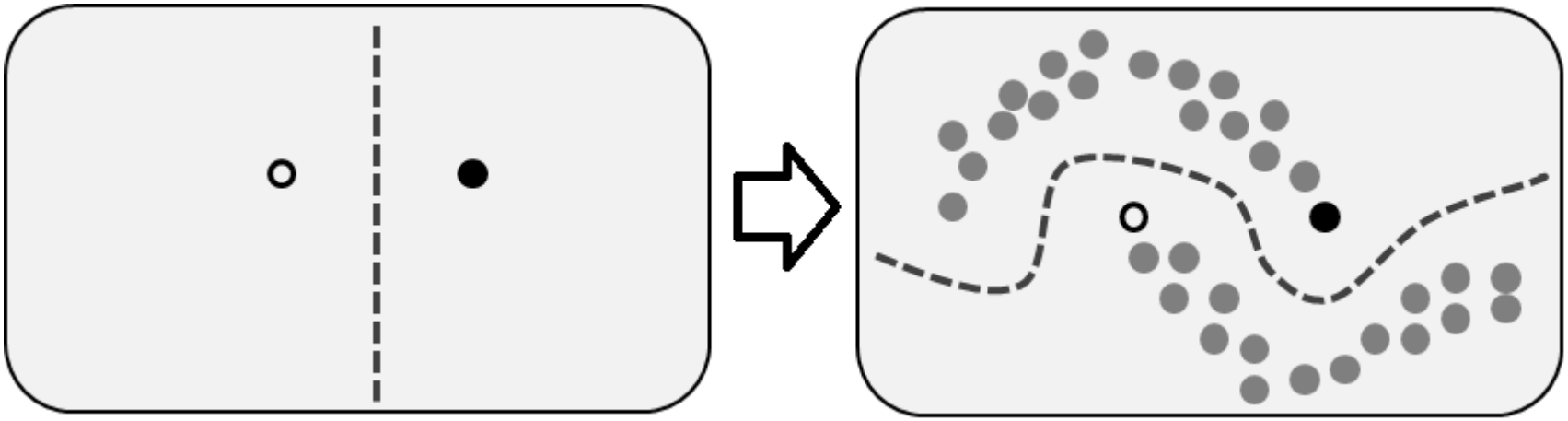
Научный руководитель: Астраханцев Н. А.

ИСП РАН, 2016г.

Частичное обучение

Определение

Частичное обучение — класс алгоритмов машинного обучения, способных тренировать эффективные модели классификации и регрессии **на малых выборках размеченных данных** при условии, что в момент обучения **доступны все неразмеченные** данные.



Apache Spark

Apache Spark — программный каркас с открытым исходным кодом для реализации распределённой обработки неструктурированных и слабоструктурированных данных, входящий в экосистему проектов Hadoop.

Ключевые особенности:

1. Отсутствие повторной загрузки данных с диска
2. Быстрое восстановление данных
3. Равномерное распределение нагрузки по узлам

Включает в себя библиотеку *MLlib*, которая предоставляет широкий выбор инструментов для задач машинного обучения.

Цель работы

Цель

Анализ и реализация некоторых алгоритмов частичного обучения на *Apache Spark*

Этапы работы:

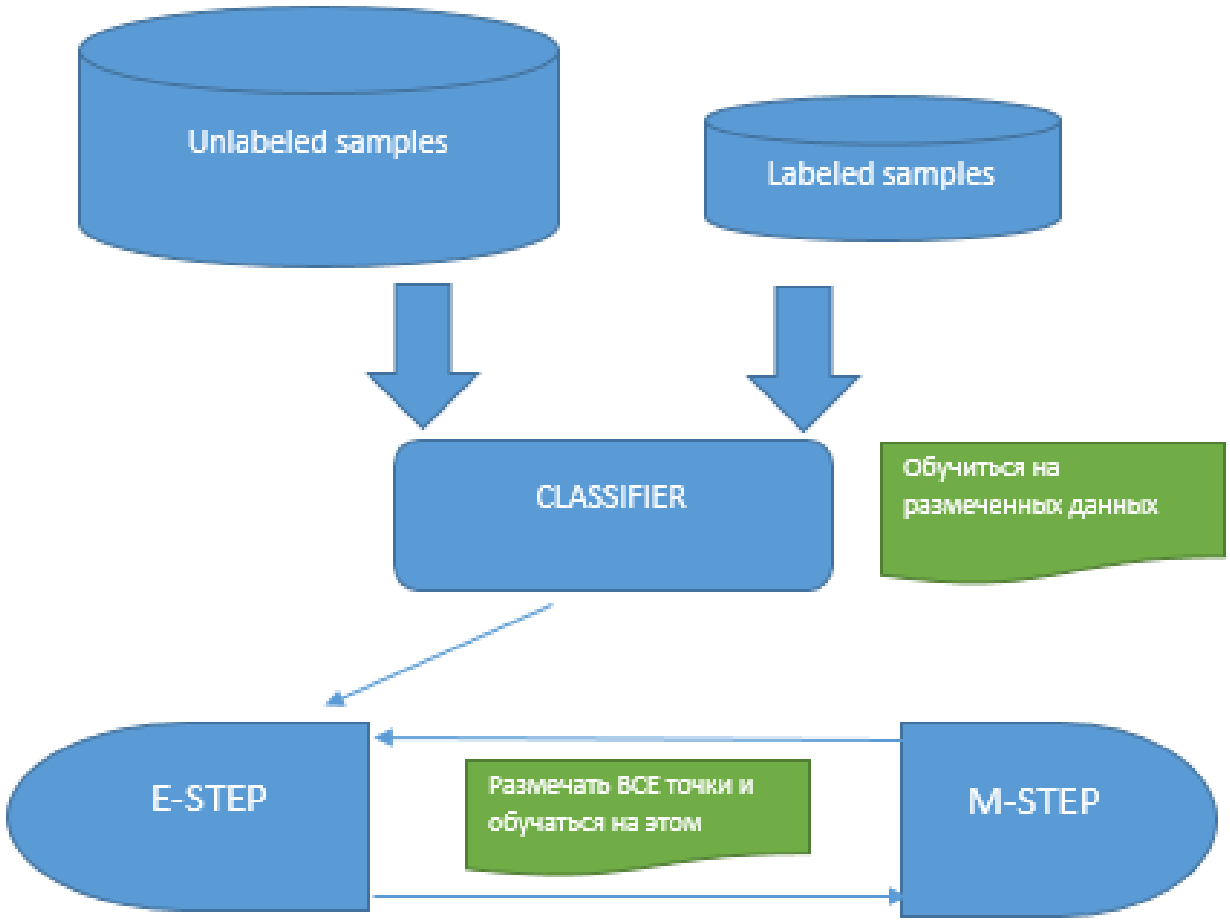
1. Анализ существующих алгоритмов частичного обучения
2. Выбор и реализация нескольких алгоритмов для задач бинарной классификации на *Apache Spark* с использованием *MLlib*
3. Выбор метрик качества и сравнительный анализ работы реализованных алгоритмов

Существующие алгоритмы частичного обучения

- **Генеративные**: Semi-supervised EM
- **Дискриминативные**: Transductive SVM, Semi-Supervised Trees, SemiBoost, Gradient Boosting with Priors and
- **Обертки над существующими алгоритмами**: Self-Training, Co-Training, Multiview Learning Manifold regularization

Алгоритмы для реализации были выбраны по причине простоты, распространенности и заявленному в статьях хорошему качеству работы.

Semi-supervised EM



Semi-supervised EM

Предположения

- Распределение данных представляет из себя смесь гауссиан
- За конечное число наблюдений этой смеси возможно полностью восстановить её параметры (**распознаваемость**)

Принцип работы

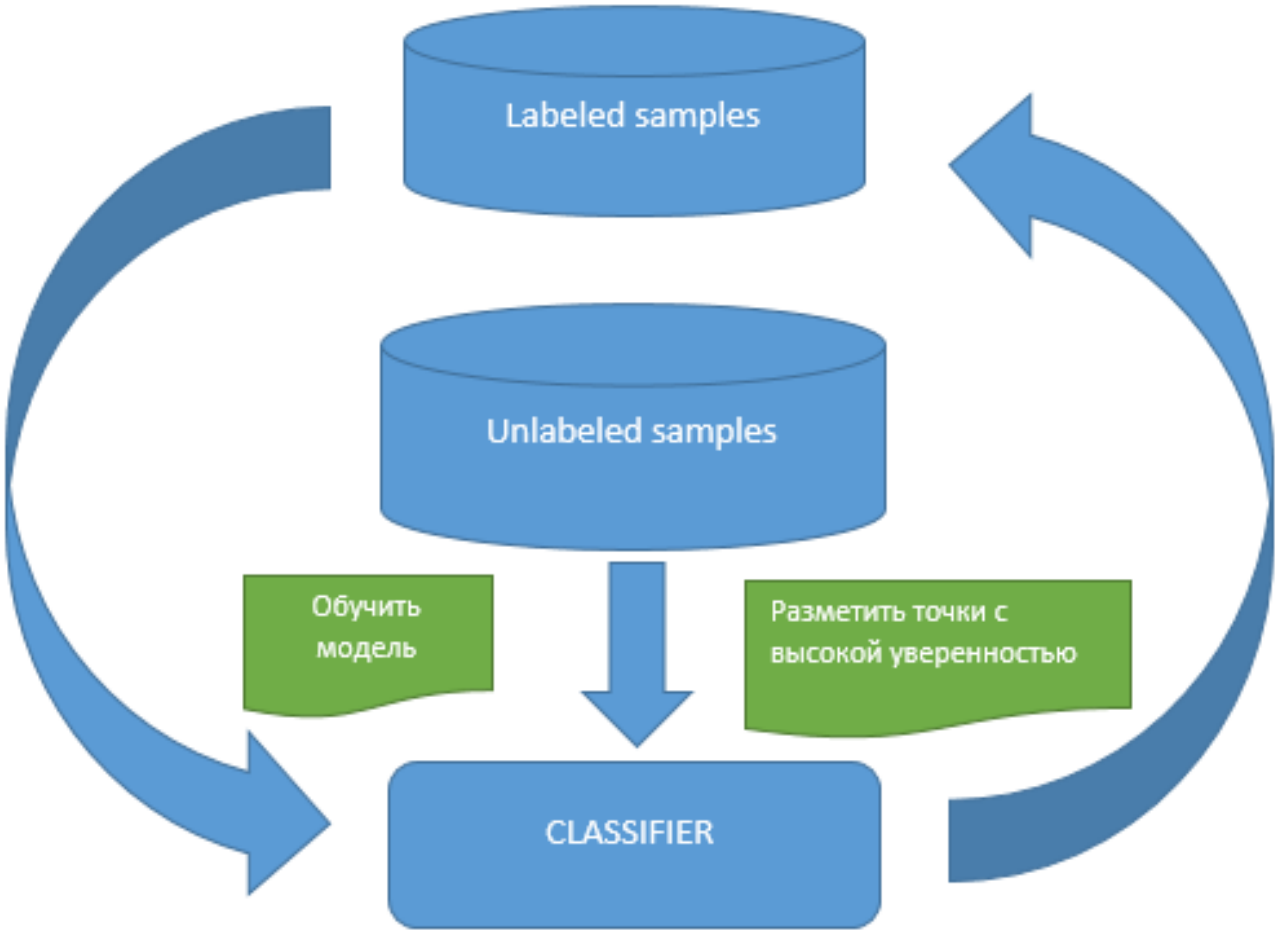
В соответствии с принципом максимального правдоподобия итеративно строятся оценки параметров распределений смеси.

\+ Можно использовать с любой вероятностной моделью

\− Сильно зависит от начальных размеченных данных (локальность)

\− Если исходные предположения не выполняются, работает плохо

SelfTraining



SelfTraining

Принцип работы

Обучаться, предсказывать и затем переобучаться, используя свои собственные наиболее надежные предсказания.

\+ Можно использовать с любым алгоритмом обучения с учителем

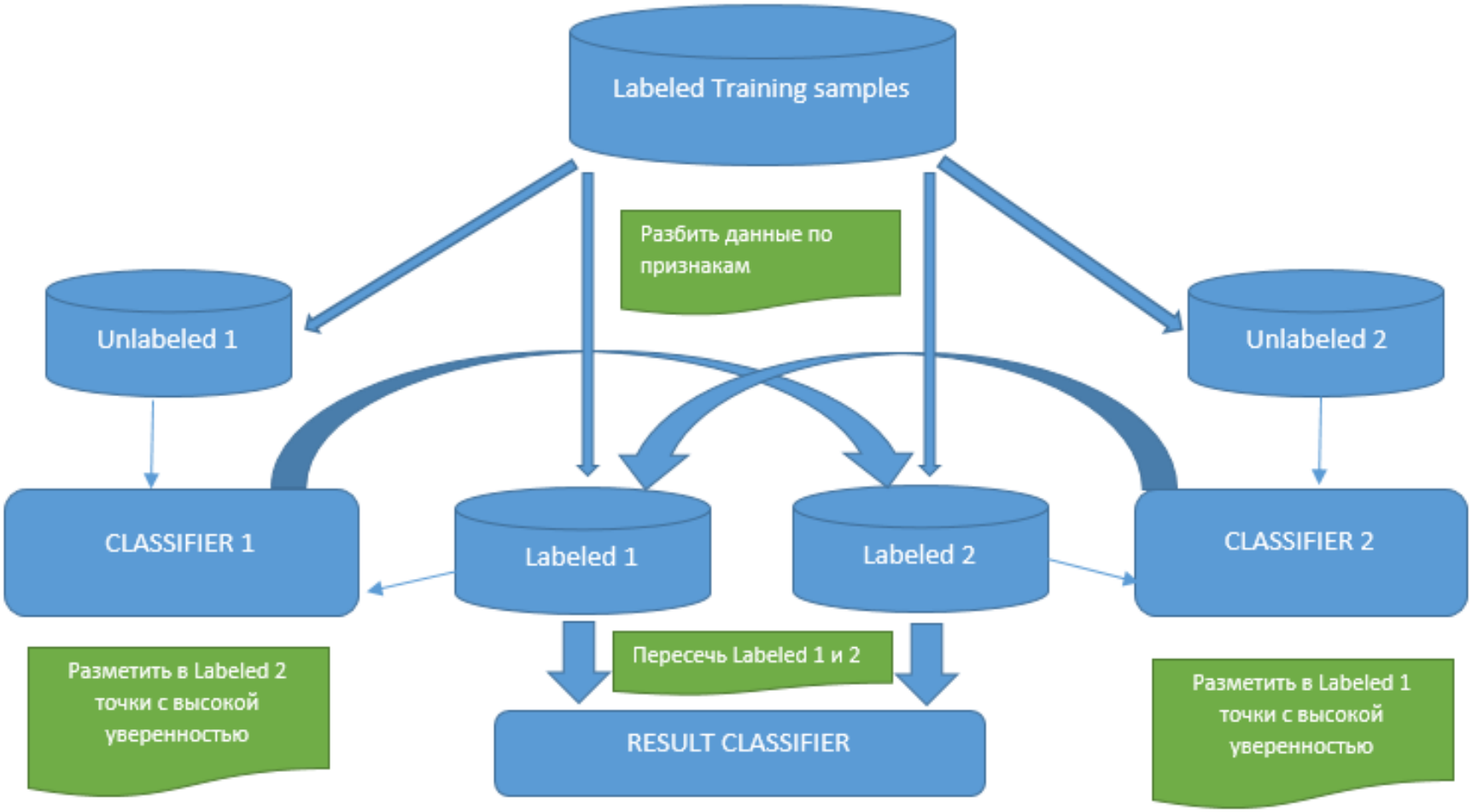
\+ Простота алгоритма и неплохие результаты на практике

\– Ошибки в размеченных данных будут многократно усилены

\– Неустойчив к выбросам в данных

\– Сложно анализировать в общем смысле

CoTraining


Labeled Training samples
Разбить данные по признакам
Unlabeled 1
Unlabeled 2
CLASSIFIER 1
Labeled 1
Labeled 2
CLASSIFIER 2
Разметить в Labeled 2 точки с высокой уверенностью
Пересечь Labeled 1 и 2
RESULT CLASSIFIER
Разметить в Labeled 1 точки с высокой уверенностью

CoTraining

Предположения

- Признаки точек можно разделить на 2 репрезентативных условно независимых по классам множества

Принцип работы

Разбить данные на 2 независимых репрезентативных множества, на каждом из них поочередно обучать алгоритм, используя предсказания друг друга.

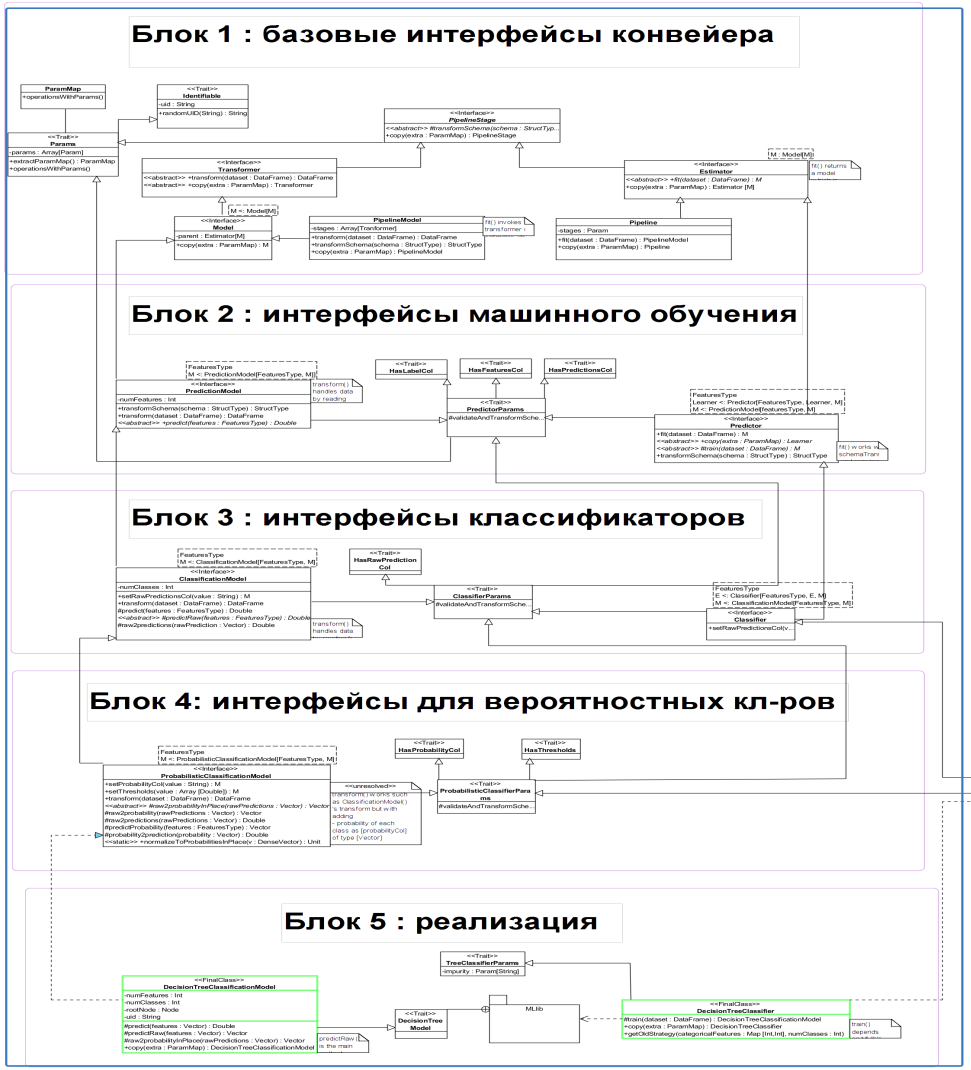
\+ Можно использовать с любыми алгоритмами обучения с учителем

\+ При выполнении предположений отличные результаты

\− Слишком сильное предположение для исходных данных

\− Нетривиальный поиск требуемого разбиения множества признаков

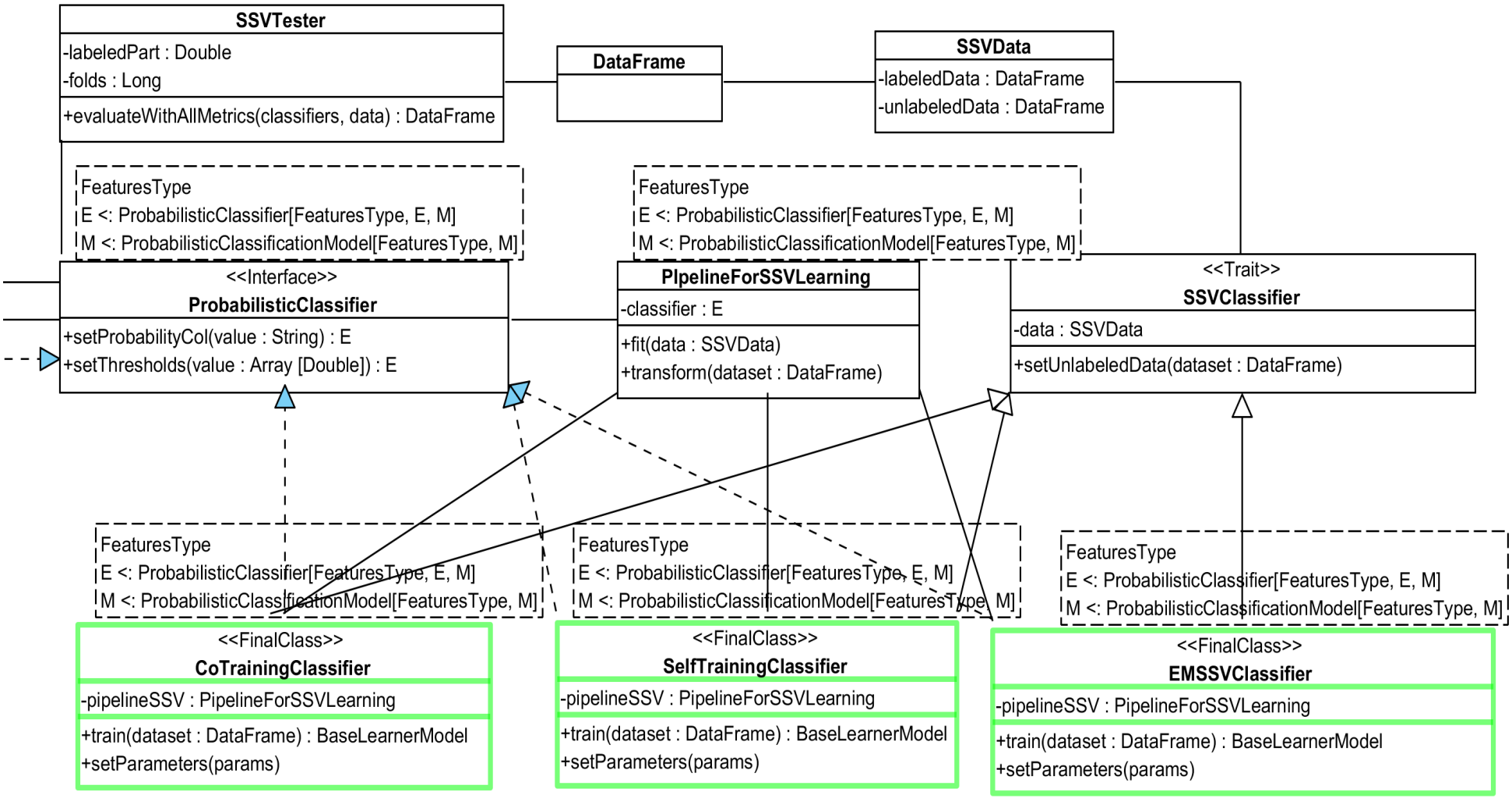
Структура классов



Структура MLlib

Semi-supervised

Детализация semi-supervised классов



Свойства реализации

Текущая реализация полностью удовлетворяет интерфейсу MLLib.
Каждый класс реализует интерфейс *Probabilistic Classifier*.

Основные черты:

- \+ Поддержка любого вероятностного базового алгоритма
- \+ Свободная настройка параметров
- \+ Возможность встроить в конвейер

Код на *github.com/hbq1/SSL_last*

Тестирование

Этапы тестирования:

1. Проанализировать зависимость качества работы от количества размеченных точек
2. Определить зависимость качества от максимального количества совершаемых итераций
3. Определить зависимость качества от границы принятия данных
4. Сравнить среднее время работы алгоритмов

Методика тестирования

Для каждого датасета проводится 10 случайных разбиений на размеченную и неразмеченную части в пропорциях $\{0.005, 0.0075, 0.01, 0.015, 0.025, 0.05, 0.1, 0.15, 0.2, 0.4\}$ от всех данных, далее отрабатывают все тестируемые алгоритмы, измеряется качество по F1-мере для каждого разбиения и результат усредняется.

- Параметры алгоритмов фиксированы
- В ходе тестирования строятся доверительные интервалы (по F1-мере)
- Измеряется среднее время работы

Данные

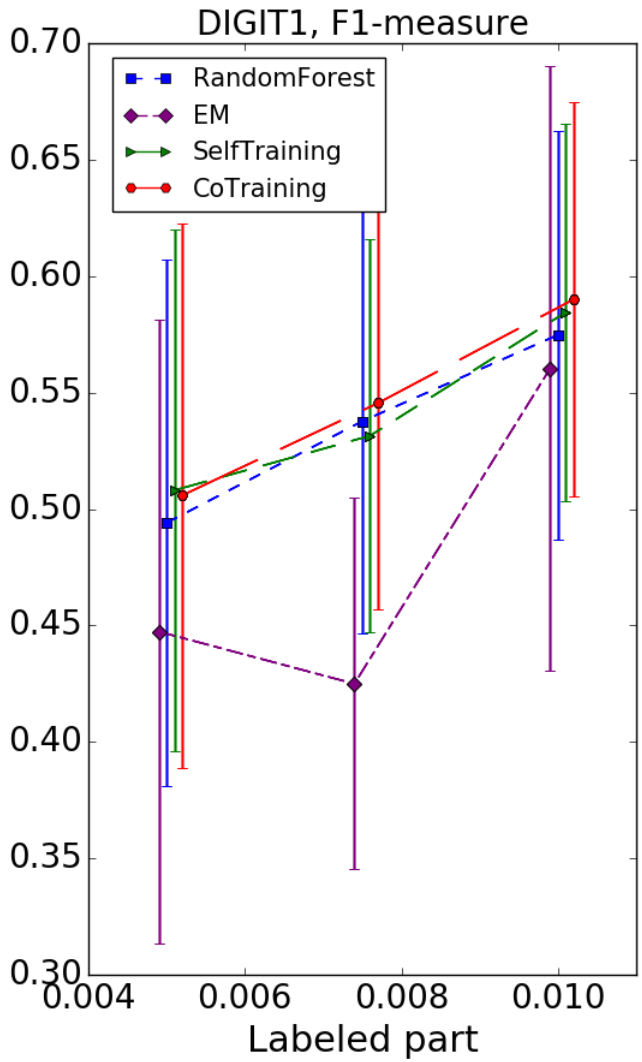
Таблица: Свойства датасетов

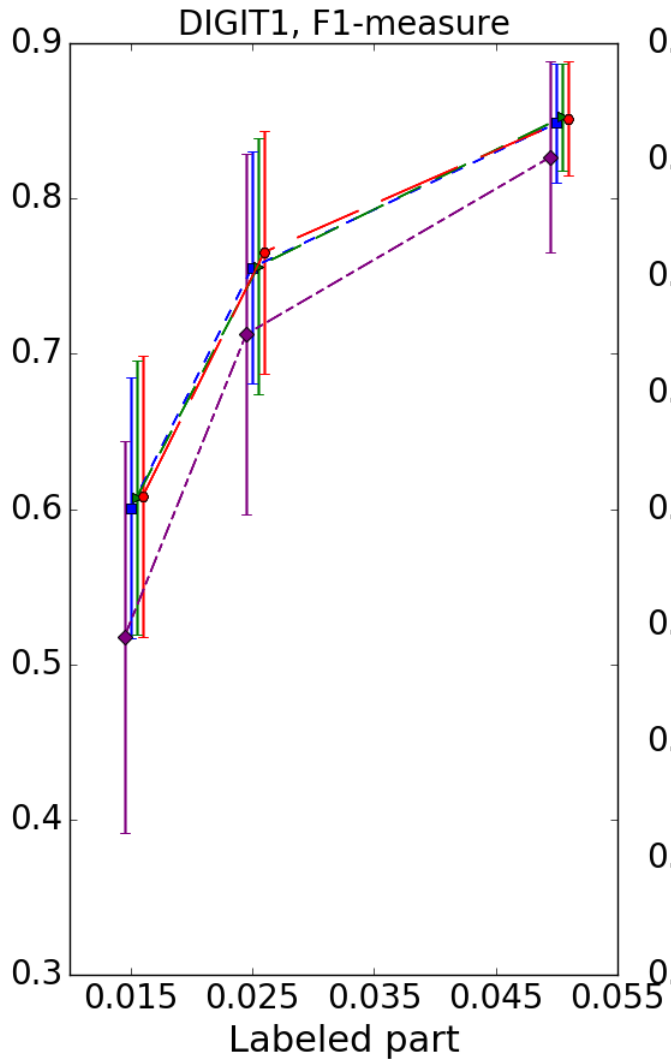
Название	точек	признаков	баланс	cluster a.	manifold a.
Digit1	1500	241	✓	×	✓
USPS	1500	241	×	✓	✓
g241c	1500	241	✓	✓	×
g241d	1500	241	×	×	×
mailSpam	17417	102	?	?	?

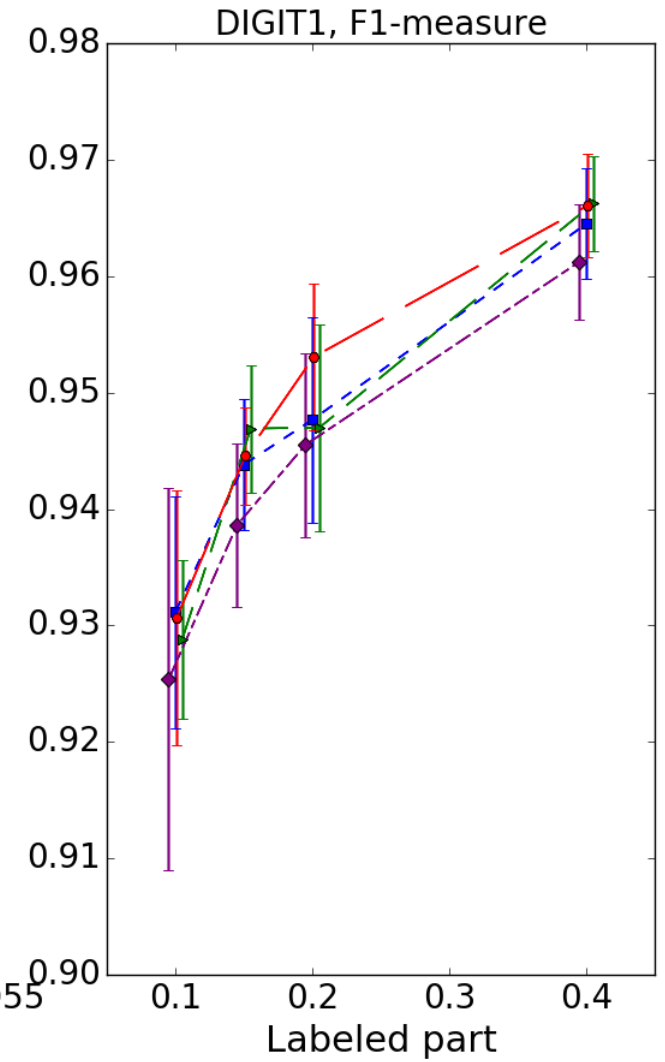
Cluster assumption — предположение о том, что данные распределены по некоторым кластерам.

Manifold assumption — предположение о том, что данные расположены на некотором подпростанстве исходного пространства признаков.

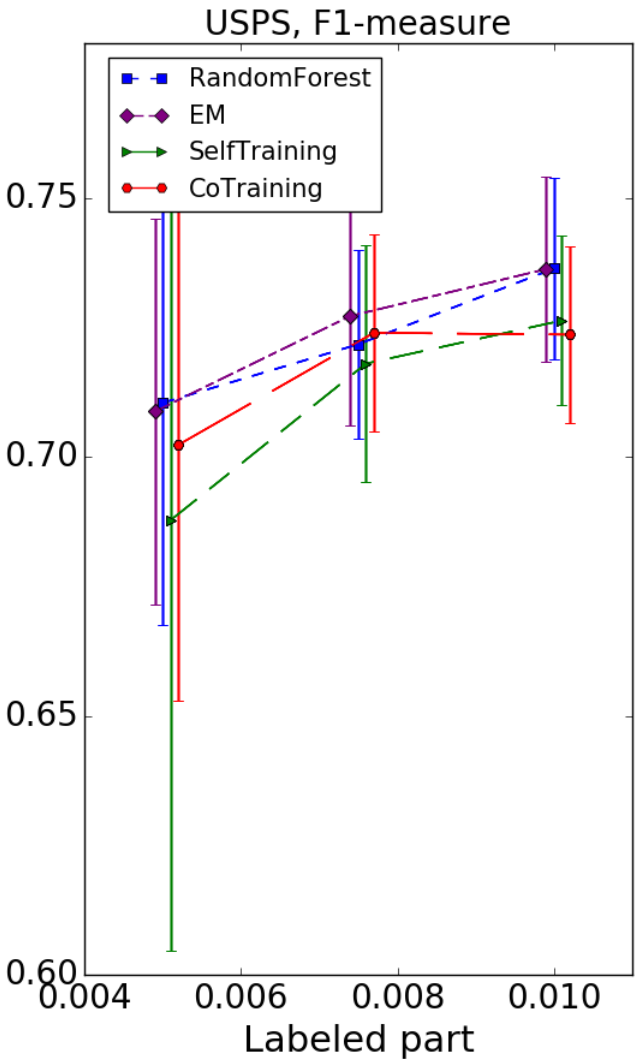
Результаты тестирования: DIGIT1

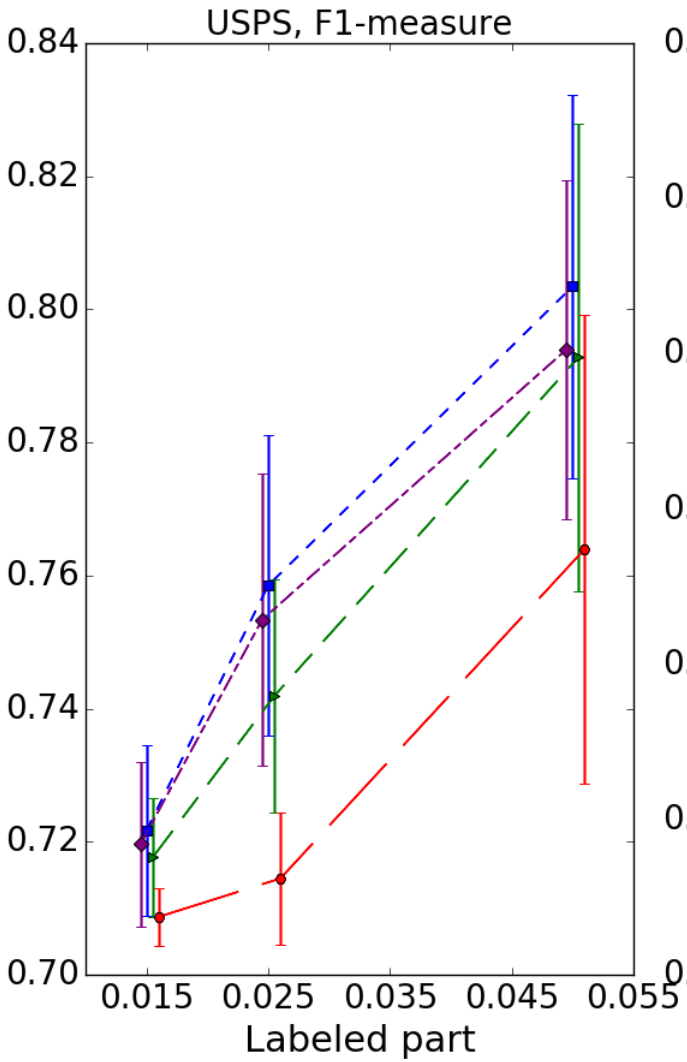


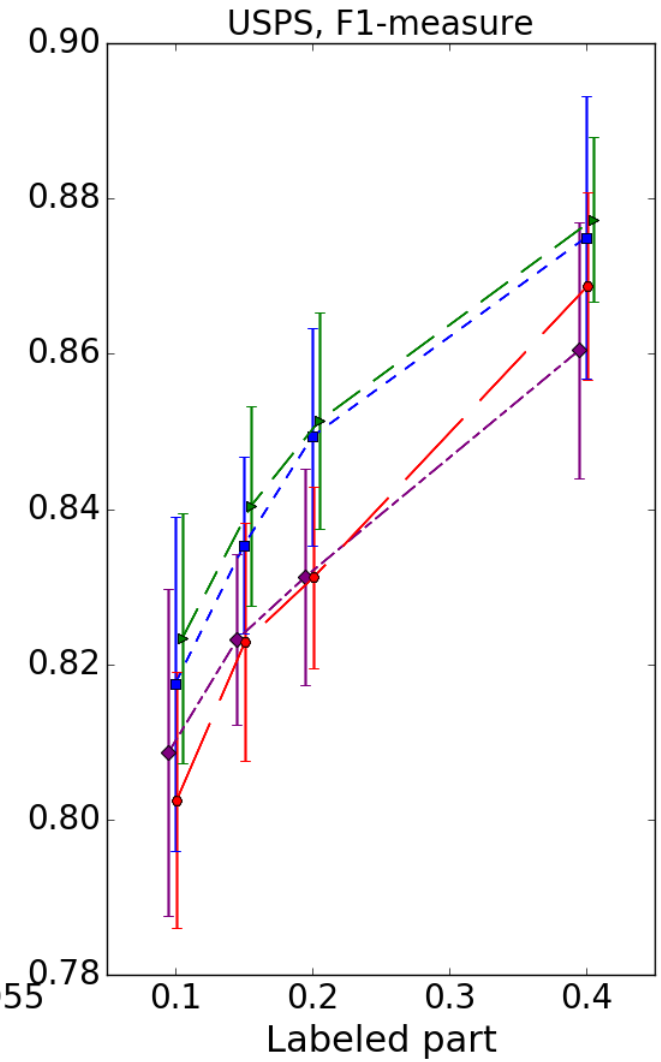




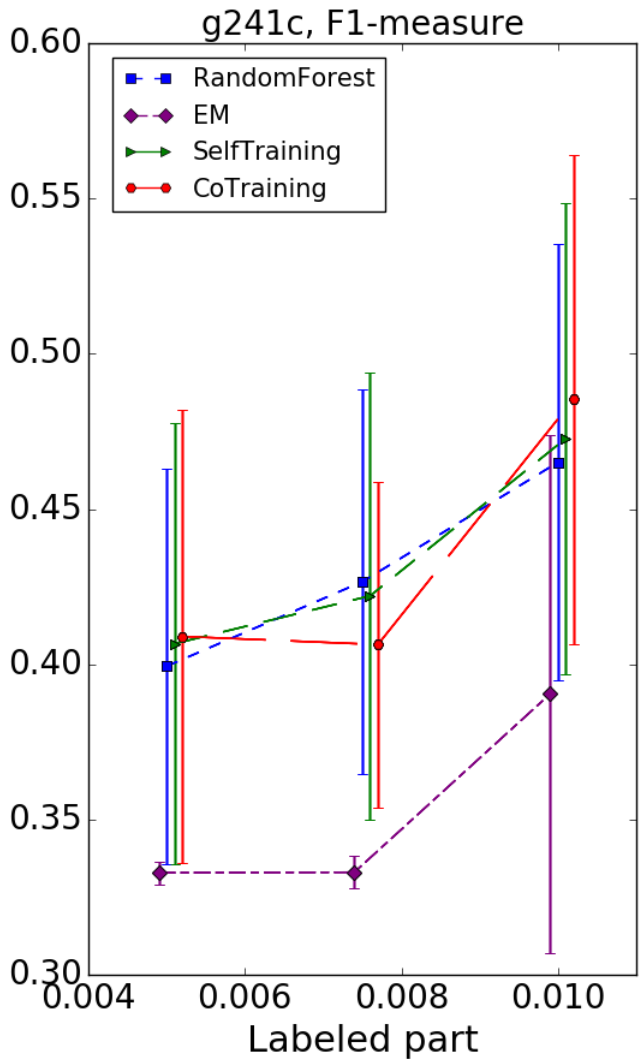
Результаты тестирования: USPS

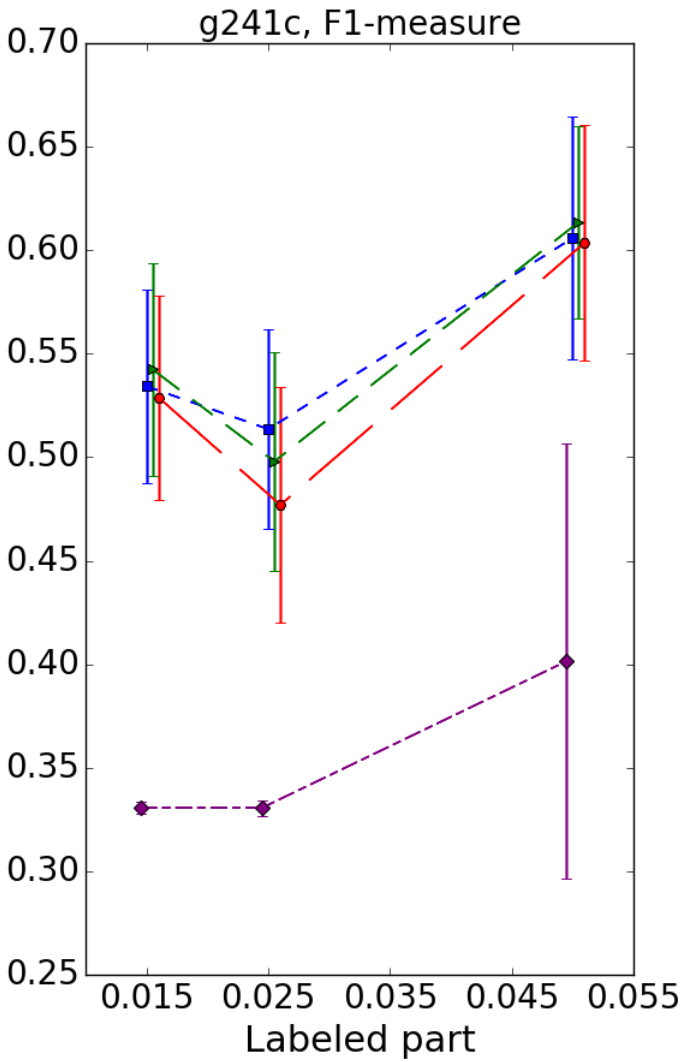


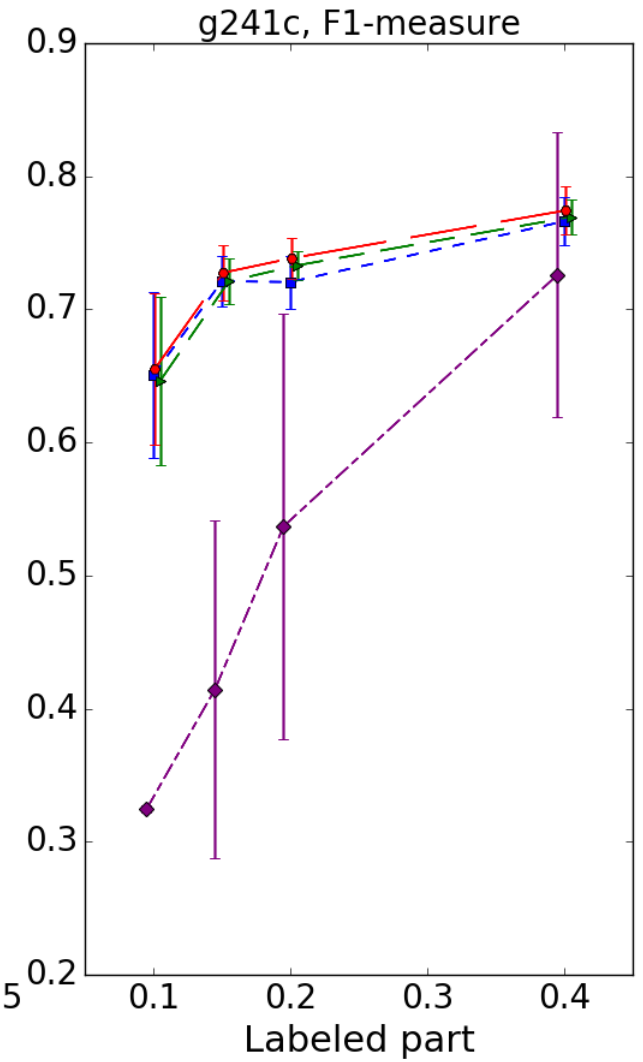




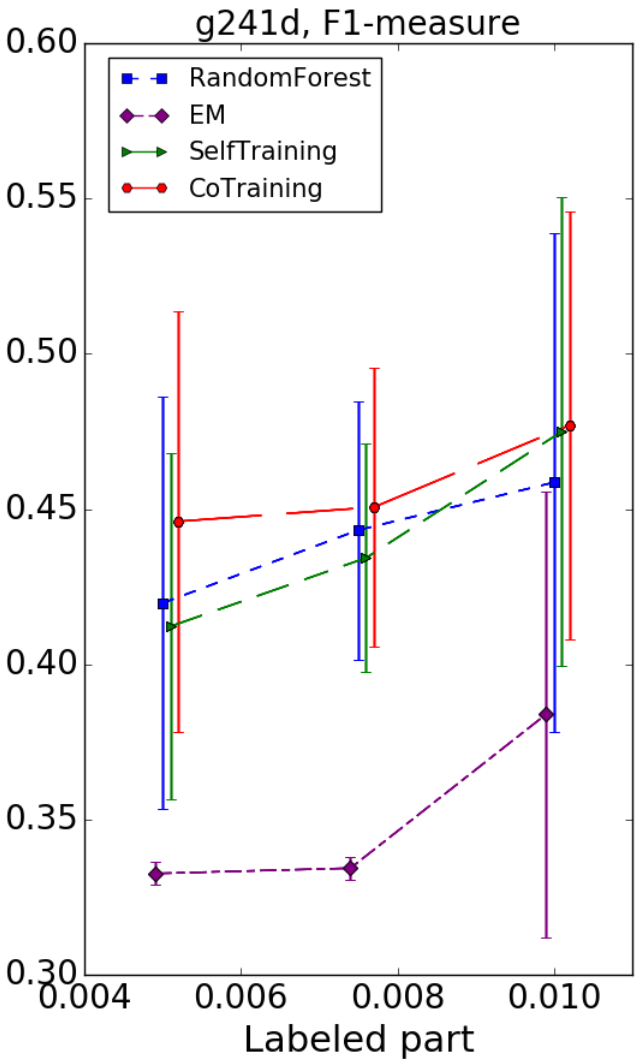
Результаты тестирования: g241c

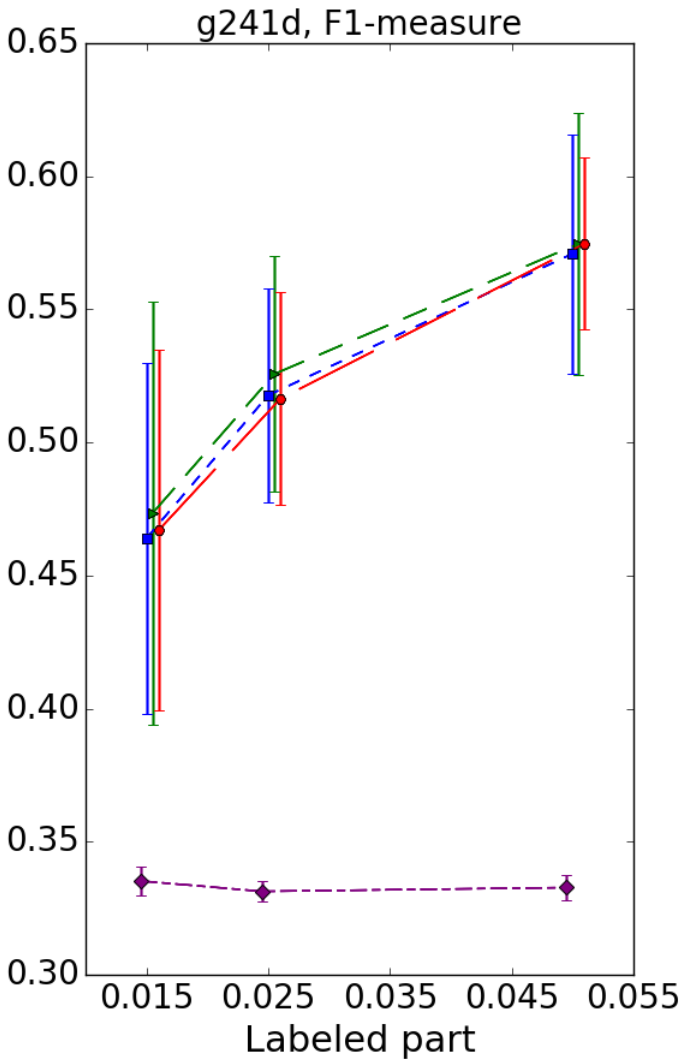


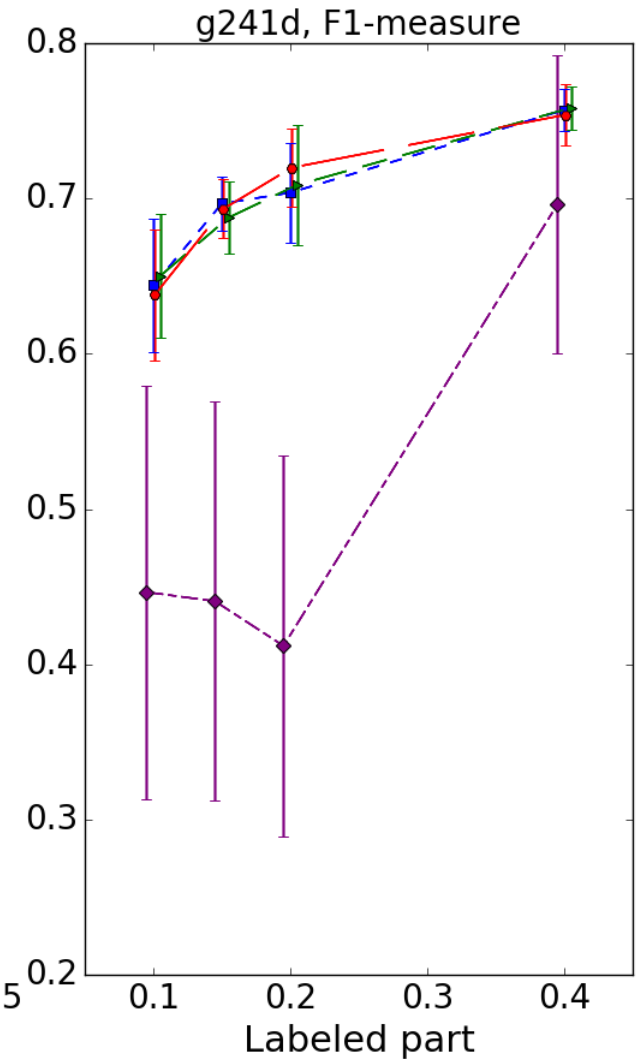




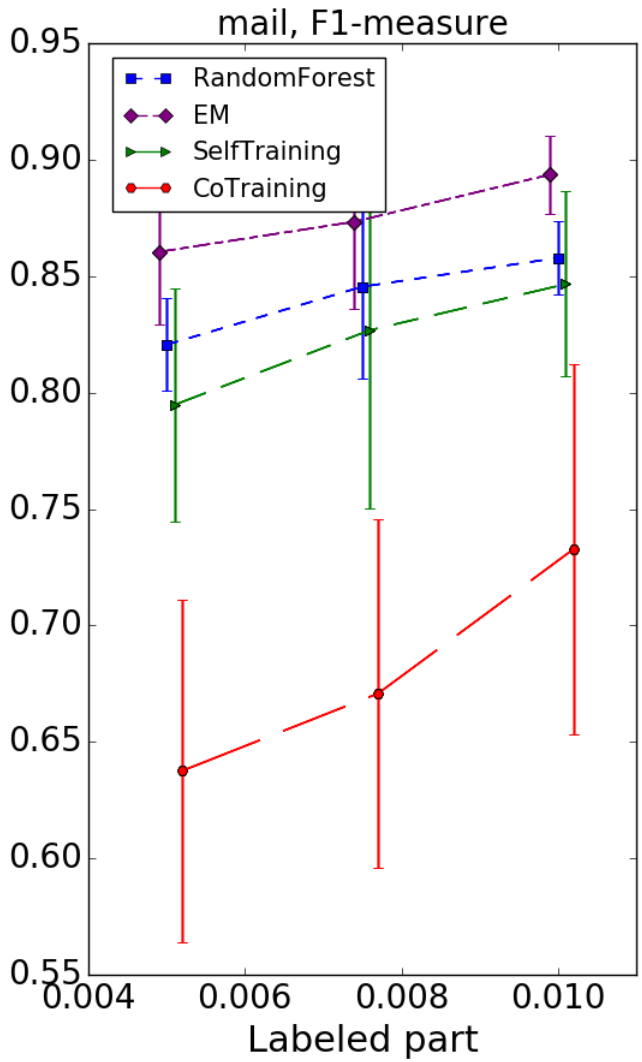
Результаты тестирования: g241d

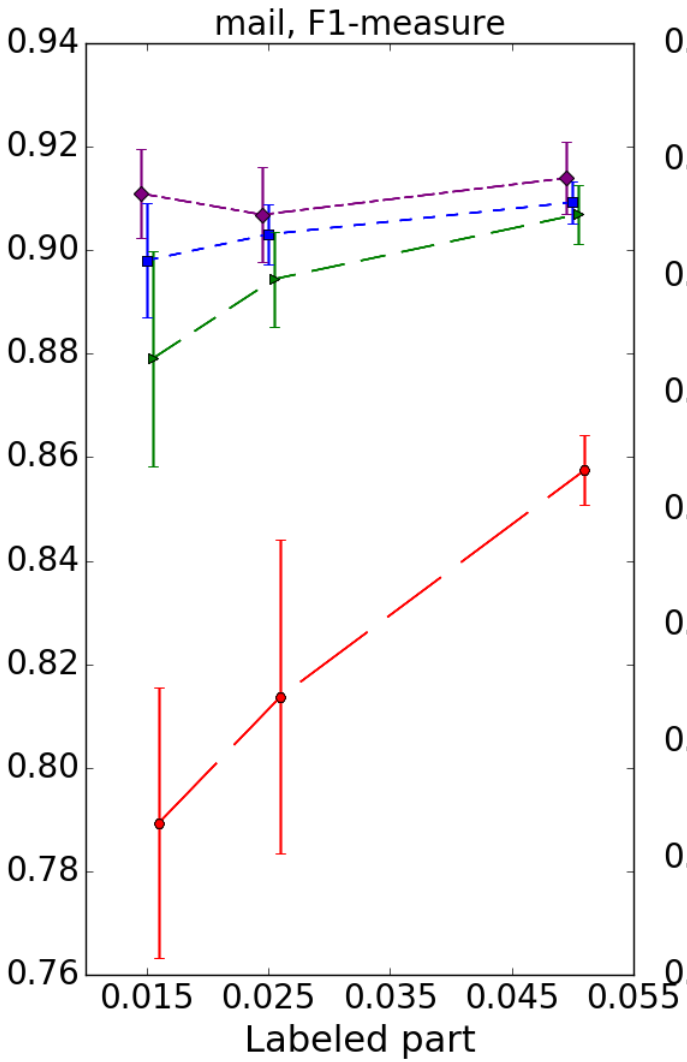


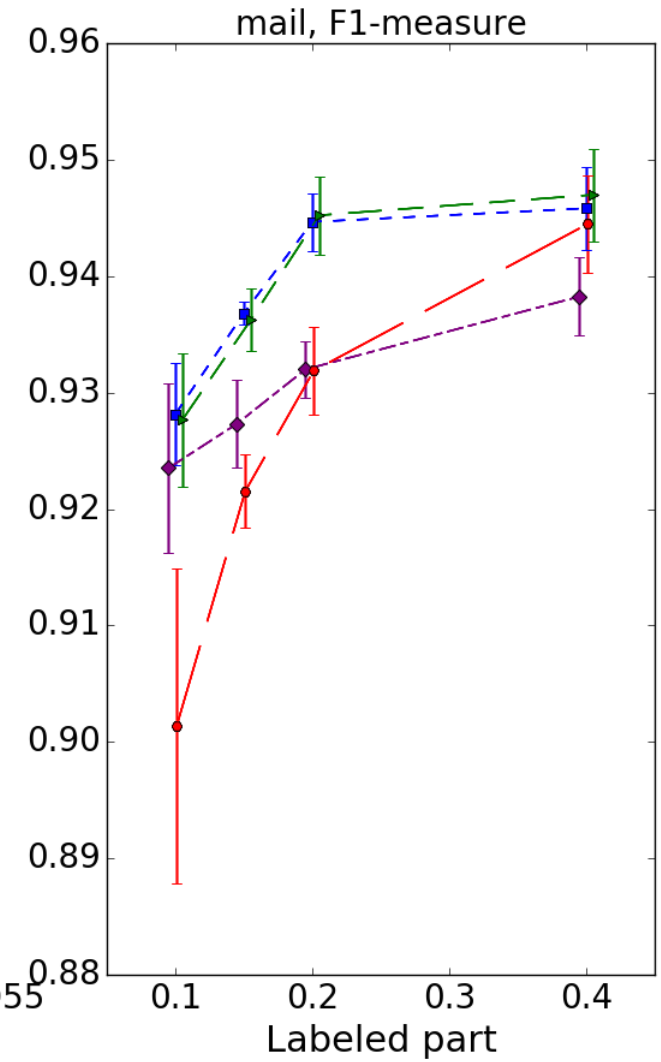




Результаты тестирования: mailSpam







Результаты тестирования: итог

	Random Forest	*Self-Training*	*Co-Training*	*EM*
DIGIT1	0.760 ± 0.052	0.763 ± 0.050	0.766 ± 0.052	0.726 ± 0.068
g241c	0.580 ± 0.047	0.583 ± 0.047	0.581 ± 0.048	0.412 ± 0.060
g241d	0.567 ± 0.045	0.570 ± 0.045	**0.574** ± 0.043	0.405 ± 0.057
USPS	**0.783** ± 0.020	0.778 ± 0.024	0.767 ± 0.019	0.776 ± 0.024
mailSpam	0.899 ± 0.010	0.890 ± 0.021	0.820 ± 0.032	**0.908** ± 0.012

Таблица: Средняя мера F1 по датасетам

Возможные причины "слабой"работы:

- преобладают нерепрезентативные размеченные выборки при случайных разбиениях
- неоптимальный выбор параметров
- много (241) признаков для маленьких (1500 точек) выборок

Результаты тестирования: время

	Random Forest	Self-Training	Co-Training	EM
DIGIT1	**12.29**	46.45	117.34	140.17
g241c	**11.65**	27.12	53.49	131.23
g241d	**12.49**	28.79	56.30	145.91
USPS	**17.57**	101.12	214.51	136.41
mailSpam	**12.61**	68.08	134.34	137.28

Таблица: Среднее время обучения по датасетам

Результат работы

1. Проведен анализ существующих алгоритмов частичного обучения, составлен подробный обзор
2. Написана реализация алгоритмов *Self-Training*, *Co-Training*, *EM* для случая бинарной классификации
   - Составлен обзор фреймворка *Apache Spark*
   - Подробно рассмотрена структура *MLlib*
   - Спроектирована полностью удовлетворяющая интерфейсу *MLlib* структура классов выбранных алгоритмов
3. Получены и проанализированы результаты работы реализованных алгоритмов на 5-ти наборах данных
   - Реализован универсальный тестер для задач бинарной классификации
   - Проведен сравнительный анализ с базовыми алгоритмами
   - Проведены тесты на зависимость качества от задаваемых параметров

Спасибо за внимание

Параметры алгоритмов

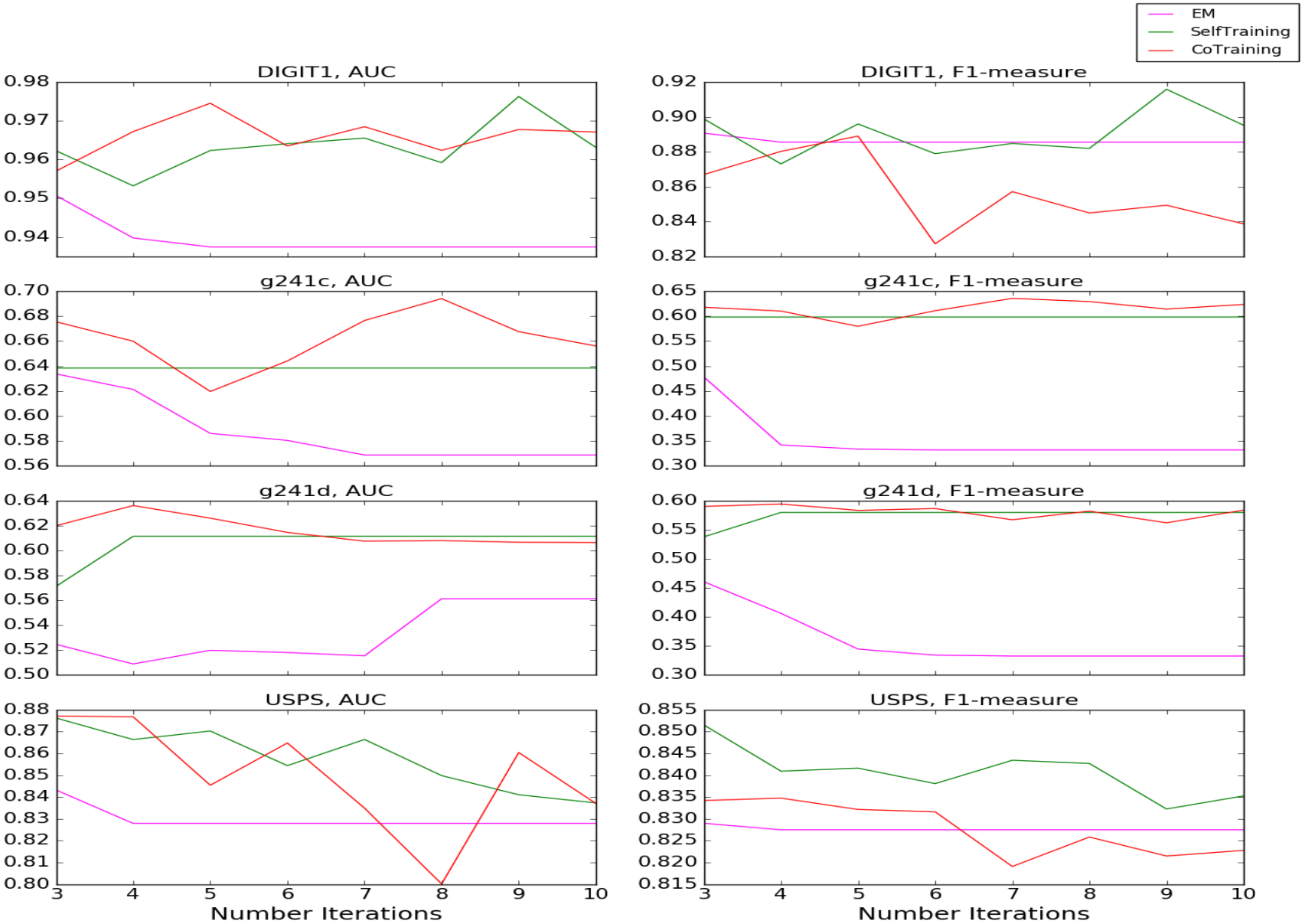
Количество деревьев RandomForest: 128.

	базовая м.	**max. итераций**	**гр. принятия**
Self-Training	RF	3	0.95
Co-Training	RF	3	0.90
EM	RF	10	–

Таблица: Параметры алгоритмов

Разделение по признакам в алгоритме Co-Training случайное, на 2 равномощных множества.
Окрестность сходимости EM: 3%

Тест по итерациям



Тест по границе принятия

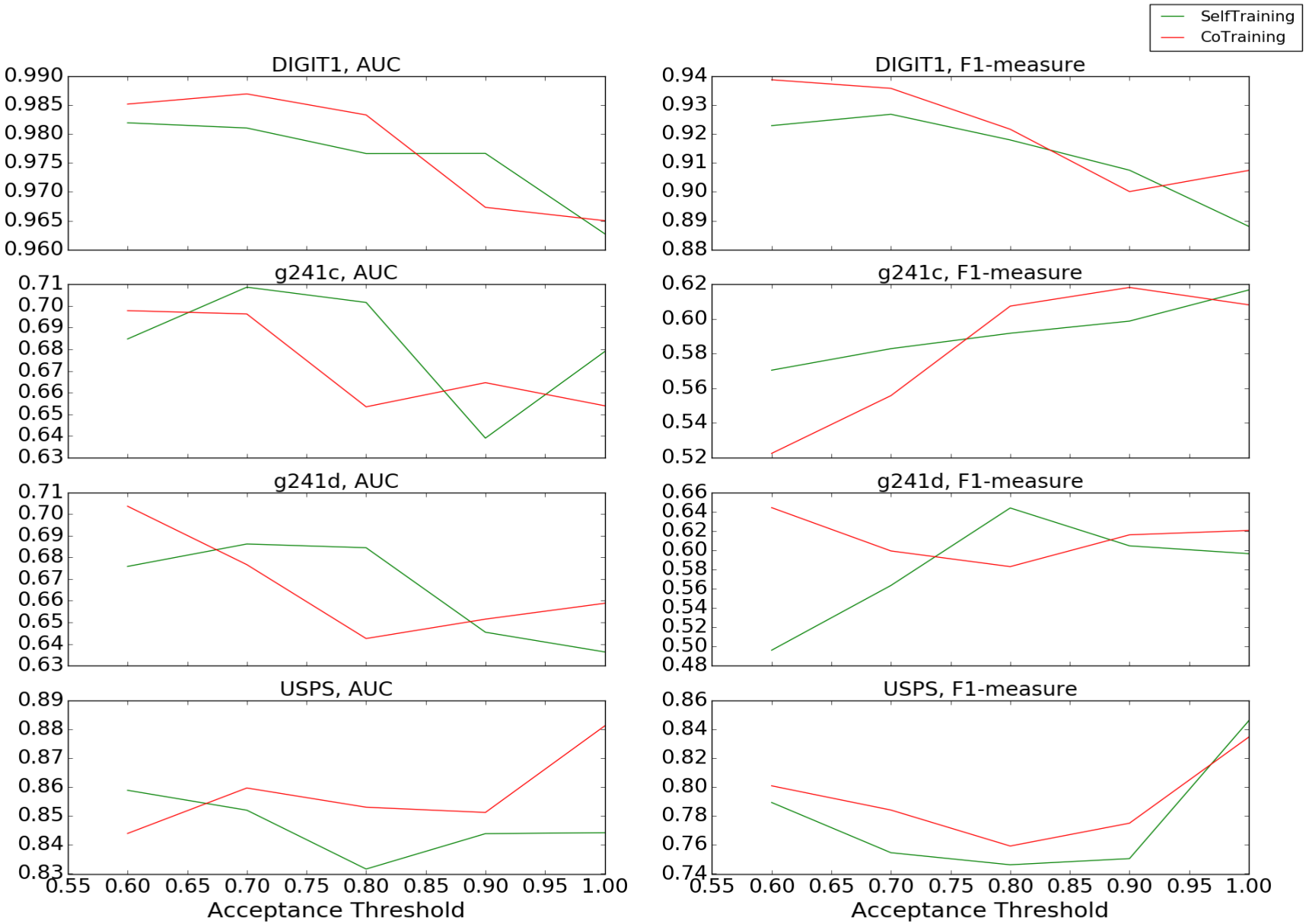
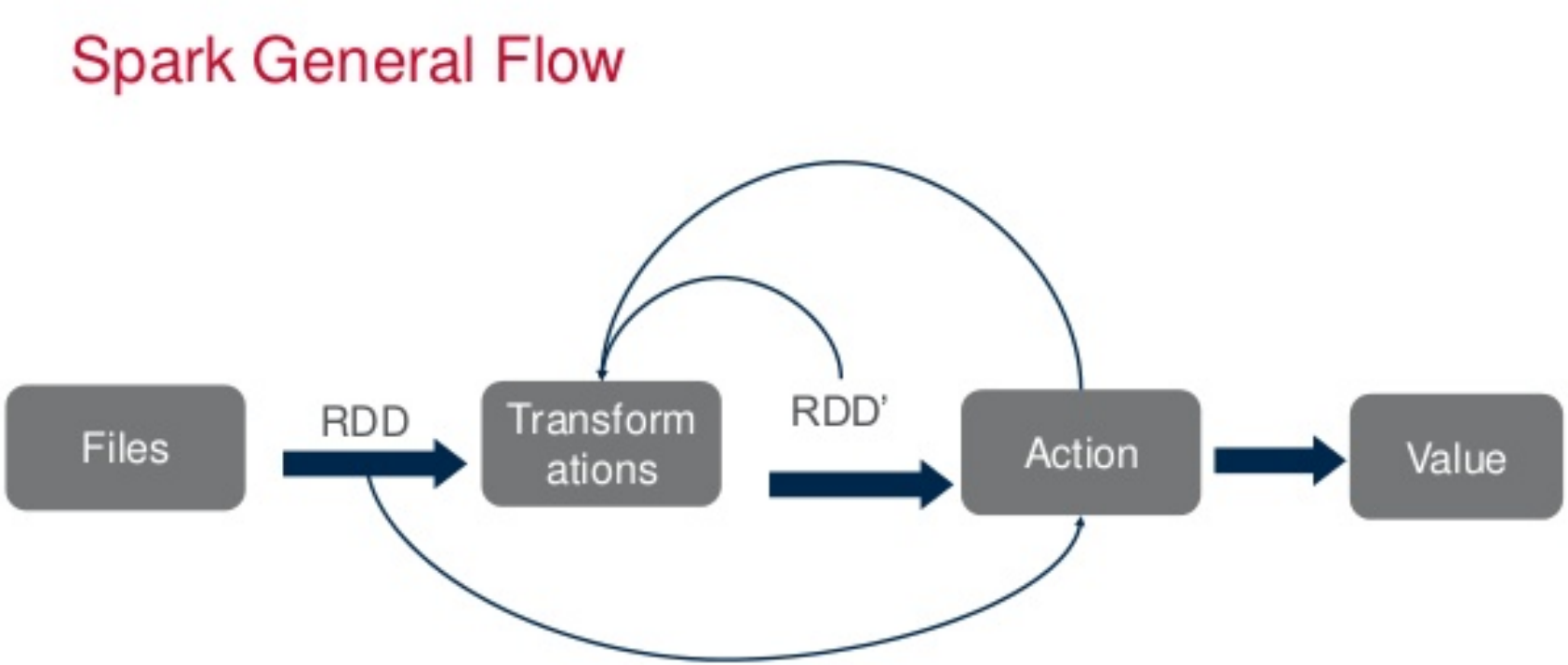


Схема работы Apache Spark



 11