МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное
учреждение высшего образования
«Ярославский государственный университет имени П. Г. Демидова»

Кафедра информационных сетей и технологий

Сдано на кафедру
«_____»_____ 2018 г.
Заведующий кафедрой,
к. ф.-м. н., декан

_____ Д. Ю. Чалый

Курсовая работа

Анализ методов машинного обучения для решения задачи “Predict future sales”

по направлению
09.03.03 Фундаментальная информатика и информационные технологии

Научный руководитель
к. ф.-м. н., декан

_____ Д. Ю. Чалый
«_____»_____ 2018 г.

Студент группы ПИЭ-31БО

_____ Н. А. Езжев
«_____»_____ 2018 г.

Ярославль, 2018

Реферат

Объем 31 с., 4 гл., 3 рис., 2 табл., 40 источников, 1 прил.

Ключевые слова: **Машиннное обучение.**

Содержание

Введение

Для исследования методов машинного обучения была выбрана задача “Predict future sales” выложенная на ресурсе Kaggle. Цель задачи – определение будущих месячных продаж товаров на основе статистики продаж за несколько лет.
В ходе решения задачи предстоит проанализировать эффективность нескольких алгоритмов машинного обучения, чтобы выявить тот который лучше всего подходит для решения данной задачи.

1. Структура курсовой работы

Курсовая работа включает следующие структурные элементы:

1) титульный лист;
2) реферат;
3) содержание;
4) введение;
5) основную часть:
    - О задаче,
    - Теоретическая часть,
    - Решение задачи,
    - Результаты решения задачи;
6) заключение;
7) список использованных источников (список литературы);
8) приложения.

2. О задаче

2.1. Постановка задачи

Необходимо предсказать продажи в следующем месяце на основе статистики о дневных продажах товаров. Набор данных предоставленный для решения поставленной задачи включает следующие файлы:

- sales_train.csv – файл с обучающим набором данных, включает дневную статистику продаж с января 2013 по октябрь 2015 года;
- test.csv – тренировочный файл, содержащий индексы магазинов для которых должны быть предсказаны продажи;
- items.csv – файл содержит подробную информацию о товарах;
- shops.csv – файл содержит подробную информацию о магазинах.

Наборы данных состоят из следующего набора полей:

1. ID – поле-идентификатор для строки значений в таблице
2. shop_id – уникальный идентификатор магазина
3. item_id – уникальный идентификатор товара
4. item_category_id – уникальный идентификатор категории
5. item_cnt_day – дневные продажи товара
6. item_price – стоимость товара
7. date – дата продажи в формате dd/mm/yyyy
8. date_block_num – номер месяца в последовательности. Январь 2013 – 0,. . . , Октябрь 2015 – 33
9. item_name – название товара
10. shop_name – название магазина
11. item_category_name – название категории товара

2.2. Используемые программные средства

В качестве основного инструмента для анализа будет использоваться язык программирования Python с дополнительным набором библиотек. При первичной обработке данных будут использоваться библиотеки NumPy, Pandas. Для визуализации служат библиотеки seaborn и matplotlib. При решении задачи будут использоваться модели машинного обучения, реализованные в библиотеке scikit-learn.

3. Теоретическая часть

3.1. Машинное обучение

Машинное обучение – подраздел искусственного интеллекта, изучающий методы построения обучающихся алгоритмов. Выделяют два способа обучения. Обучение по прецедентам или индуктивное обучение, основано на выявлении общих закономерностей в наборах данных. Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний. Дедуктивное обучение принято относить к области экспертных систем, поэтому термины машинное обучение и обучение по прецедентам можно считать синонимами.

3.1.1. Обучение по прецедентам

Задача обучения по прецедентам будет звучать следующим образом дано конечное множество прецедентов (объектов, ситуаций), по каждому из которых собраны (измерены) некоторые данные. Данные о прецеденте называют также его описанием. Совокупность всех имеющихся описаний прецедентов называется обучающей выборкой. Требуется по этим частным данным выявить общие зависимости, закономерности, взаимосвязи, присущие не только этой конкретной выборке, но вообще всем прецедентам, в том числе тем, которые еще не наблюдались.

Наиболее распространенным способом описания прецедентов является признаковое описание. Фиксируется совокупность n показателей, измеряемых у всех прецедентов. Если все n показателей числовые, то признаковые описания представляют собой числовые векторы размерности n. Возможны и более сложные случаи, когда прецеденты описываются временными рядами или сигналами, изображениями, видеорядами, текстами, попарными отношениями сходства или интенсивности взаимодействия и т.д.

Для решения задачи обучения по прецедентам в первую очередь фиксируется модель восстанавливаемой зависимости. Затем вводится функционал качества, значение которого показывает, насколько хорошо модель описывает наблюдаемые данные. Алгоритм обучения ищет такой набор параметров модели, при котором функционал качества на заданной обучающей выборке принимает оптимальное значение. Процесс настройки модели по выборке данных в большинстве случаев сводится к применению численных методов оптимизации.

Существует следующая типология задач обучения по прецедентам:

– Обучение с учителем (supervised learning) - наиболее распространённый случай. Каждый прецедент представляет собой пару «объект, ответ». Требуется найти функциональную зависимость ответов от описаний объектов и построить алгоритм, принимающий на входе описание объекта и выдающий на выходе ответ. Функционал качества обычно определяется как средняя ошибка ответов, выданных алгоритмом, по всем объектам выборки.
– Обучение без учителя (unsupervised learning) - в этом случае ответы не задаются, и требуется искать зависимости между объектами.
– Частичное обучение (semi-supervised learning) занимает промежуточное положение между обучением с учителем и без учителя. Каждый прецедент представляет собой пару «объект, ответ», но ответы известны только на части прецедентов. Пример прикладной задачи — автоматическая рубрикация большого количества текстов при условии, что некоторые из них уже отнесены к каким-то рубрикам.
– Трансдуктивное обучение (transductive learning). Дана конечная обучающая выборка прецедентов. Требуется по этим частным данным сделать предсказания относительно других частных данных – тестовой выборки. В отличие от стандартной постановки, здесь не требуется выявлять общую закономерность, поскольку известно, что новых тестовых прецедентов не будет. С другой стороны, появляется возможность улучшить качество предсказаний за счёт анализа всей тестовой выборки целиком, например, путём её кластеризации. Во многих приложениях трансдуктивное обучение практически не отличается от частичного обучения.
– Обучение с подкреплением (reinforcement learning). Роль объектов играют пары «ситуация, принятое решение», ответами являются значения функционала качества, характеризующего правильность принятых решений (реакцию среды). Как и в задачах прогнозирования, здесь существенную роль играет фактор времени. Примеры прикладных задач: формирование инвестиционных стратегий, автоматическое управление технологическими процессами, самообучение роботов, и т.д.
– Динамическое обучение (online learning) может быть как обучением с учителем, так и без учителя. Специфика в том, что прецеденты поступают потоком. Требуется немедленно принимать решение по каждому прецеденту и одновременно доучивать модель зависимости с учётом новых прецедентов. Как и в задачах прогнозирования, здесь существенную роль играет фактор времени.
– Активное обучение (active learning) отличается тем, что обучаемый имеет возможность самостоятельно назначать следующий прецедент, который станет известен.
– Метаобучение (meta-learning или learning-to-learn) отличается тем, что

прецедентами являются ранее решённые задачи обучения. Требуется определить, какие из используемых в них эвристик работают более эффективно. Конечная цель — обеспечить постоянное автоматическое совершенствование алгоритма обучения с течением времени.

3.2. Подходы и методы машинного обучения

Подход к задачам обучения — это концепция, точка зрения на процесс обучения, приводящая к набору базовых предположений, гипотез, эвристик, на основе которых строится модель, функционал качества и методы его оптимизации.

Разделение методов «по подходам» довольно условно. Разные подходы могут приводить к одной и той же модели, но разным методам её обучения. В некоторых случаях эти методы отличаются очень сильно, в других — совсем немного и «плавно трансформируются» друг в друга путём незначительных модификаций.

3.2.1. Статистическая классификация

В статистике решение задач классификации принято называть дискриминантным анализом.

Байесовская теория классификации основана на применении оптимального байесовского классификатора и оценивании плотностей распределения классов по обучающей выборке. Различные методы оценивания плотности порождают большое разнообразие байесовских классификаторов. Среди них можно выделить три группы методов: Параметрическое оценивание плотности:

- квадратичный дискриминант;
- линейный дискриминант Фишера.

Непараметрическое оценивание плотности:

- метод парзеновского окна.

Оценивание плотности как смеси параметрических плотностей:

- EM-алгоритм;
- метод радиальных базисных функций.

Несколько особняком стоит наивный байесовский классификатор, который может быть как параметрическим, так и непараметрическим. Он основан на нереалистичном предположении о статистической независимости признаков. Благодаря этому метод чрезвычайно прост. Другие теоретико-вероятностные и статистические подходы:

- скрытая марковская цепь;
- байесовская сеть.

3.2.2. Классификация на основе сходства

Метрические алгоритмы классификации применяются в тех задачах, где удаётся естественным образом задавать объекты не их признаковыми описаниями, а матрицей попарных расстояний между объектами. Классификация объектов по их сходству основана на гипотезе компактности, которая гласит, что в «хорошей задаче» схожие объекты чаще лежат в одном классе, чем в разных.
Метрические алгоритмы относятся к методам рассуждения на основе прецедентов, они способны дать ответ на вопрос «почему объект u был отнесён к классу y?». Алгоритм может дать понятный эксперту ответ: «потому, что имеются прецеденты — схожие с ним объекты, принадлежащие классу y», и предъявить список этих прецедентов.
Наиболее известные метрические алгоритмы классификации:

- метод ближайших соседей;
- метод парзеновского окна;
- метод потенциальных функций.

3.2.3. Классификация на основе разделимости

Большая группа методов классификации основана на явном построении разделяющей поверхности в пространстве объектов. Из них чаще всех применяются Линейные классификаторы:

- однослойный персептрон — это линейный алгоритм классификации, принцип работы которого основан на модели нервной клетки - нейрона. Представляет собой пример нейронной сети с одним скрытым слоем;
- логистическая регрессия - метод построения линейного классификатора, позволяющий оценивать апостериорные вероятности принадлежности объектов классам;
- машина опорных векторов (SVM) - является одной из наиболее популярных методологий обучения по прецедентам. Основная идея метода — перевод исходных векторов в пространство более высокой размерности и поиск разделяющей гиперплоскости с максимальным зазором в этом пространстве.

3.2.4. Нейронные сети

Нейронные сети основаны на принципе коннективизма — в них соединяется большое количество относительно простых элементов, а обучение сводится к построению оптимальной структуры связей и настройке параметров связей.

- однослойный персептрон;
- многослойный персептрон– сеть прямого распространения где входной сигнал распространяется от слоя к слою. Многослойный персептрон представляет собой обобщение однослойного персептрона;

– Нейронная сеть Кохонена - класс нейронных сетей, основным элементом которых является слой Кохонена. Слой Кохонена состоит из адаптивных линейных сумматоров. Как правило, выходные сигналы слоя Кохонена обрабатываются по правилу: наибольший сигнал превращается в единичный, остальные обращаются в ноль.

3.2.5. Индукция правил (поиск закономерностей)

Логические алгоритмы классификации представляют собой композиции простых, легко интерпретируемых правил.

– решающее дерево - метод решение задачи обучения с учителем, основанный на том, как решает задачи прогнозирования человек. В общем случае — это k-ичное дерево с решающими правилами в нелистовых вершинах (узлах) и некотором заключении о целевой функции в листовых вершинах (прогнозом). Решающее правило — некоторая функция от объекта, позволяющее определить, в какую из дочерних вершин нужно поместить рассматриваемый объект;
– решающий лес - это множество решающих деревьев. В задаче регрессии их ответы усредняются, в задаче классификации принимается решение голосованием по большинству.

3.2.6. Кластеризация

Кластеризация - задача разбиения заданной выборки объектов (ситуаций) на непересекающиеся подмножества, называемые кластерами, так, чтобы каждый кластер состоял из схожих объектов, а объекты разных кластеров существенно отличались.

– Алгоритм k средних (k-means) - это метод кластерного анализа, цель которого является разделение m наблюдений на k кластеров, при этом каждое наблюдение относится к тому кластеру, к центру (центроиду) которого оно ближе всего;
– Нейронная сеть Кохонена;

3.2.7. Регрессия

– линейная регрессия - это метод, используемый для моделирования отношений между одной независимой входной переменной (переменной функции) и выходной зависимой переменной;
– нелинейная регрессия - это вид регрессионного анализа, в котором экспериментальные данные моделируются функцией, являющейся нелинейной

комбинацией параметров модели и зависящей от одной и более независимых переменных. Данные аппроксимируются методом последовательных приближений;
- логистическая регрессия.

3.2.8. Алгоритмические композиции

Алгоритмические композиции – класс алгоритмов машинного обучения являющихся комбинацией нескольких алгоритмов.

- Бустинг - это процедура последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов. Бустинг представляет собой жадный алгоритм построения композиции алгоритмов. Изначально понятие бустинга возникло в работах по вероятно почти корректному обучению в связи с вопросом: возможно ли, имея множество плохих (незначительно отличающихся от случайных) алгоритмов обучения, получить хороший;
- Бэггинг - это технология классификации, использующая композиции алгоритмов, каждый из которых обучается независимо. Результат классификации определяется путем голосования. Бэггинг позволяет снизить процент ошибки классификации в случае, когда высока дисперсия ошибки базового метода.

3.2.9. Сокращение размерности

Уменьшение размерности - это преобразование данных очень высокой размерности в данные с гораздо меньшей размерностью, так что каждый из нижних размеров передает гораздо больше информации. Это обычно делается при решении проблем машинного обучения, чтобы получить лучшие функции для задачи классификации или регрессии. Уменьшение размерности производится с помощью следующих приемов:

- селекция признаков = отбор признаков;
- метод главных компонент;
- метод независимых компонент;
- многомерное шкалирование.

4. Решение задачи

Решение задачи “Predict future sales” заключается в определение объема товаров, которые будут проданы магазинами в следующем месяце на основе данных продаж прошлых месяцев. Процесс решения задачи “Predict future sales” можно разбить на четыре основных этапа:

- Анализ исходных данных
- Подготовка обучающей выборки
- Обучение моделей
- Анализ работы алгоритмов

4.1. Анализ исходных данных

Первый шаг анализа исходных данных - проверка файла с тренировочной выборкой на наличие неопределенных значений Рис. 1. Как видим, необходимые данные не содержат пропусков и неопределенных значений.
Следующий шаг - это проверка и удаление дубликатов строк в обучающей выборке, если таковые присутствуют. В результате в выборке было найдено и удалено 24 дублирующие строки.
Теперь необходимо проверить не содержатся ли в выборке, какие-то нелогичные значения, выбросы. Для этого найдём минимальные и максимальные значения столбцов, а также среднее и медианное значение столбца табл. 4.1.

Таблица 4.1

Средние, минимальные и максимальные значения в наборе

Параметр	MIN	MAX	MEAN	MEDIAN
item_price	-1.0	307980.0	890.8558	399.0
item_cnt_day	-22.0	2169.0	1.2426	1.0

В результате получаем противоречивые данные по следующим столбцам:

- item_price
- item_cnt_day

Очевидно, что цена на товар, как и число купленных товаров не могут быть меньше 0, скорее всего эти значения ошибочны. Так же вызывает сомнение достоверность максимального значения стоимости товаров, так как оно слишком большое. Построим график, чтобы посмотреть является ли данное значение выбросом Рис. 2.
Проанализировав полученный график, можно сказать, что значение стоимости

```
______Train______
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2935849 entries, 0 to 2935848
Data columns (total 6 columns):
date              object
date_block_num    int64
shop_id           int64
item_id           int64
item_price        float64
item_cnt_day      float64
dtypes: float64(2), int64(3), object(1)
memory usage: 134.4+ MB
None
___Categories____
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 84 entries, 0 to 83
Data columns (total 2 columns):
item_category_name    84 non-null object
item_category_id      84 non-null int64
dtypes: int64(1), object(1)
memory usage: 1.4+ KB
None
______Items______
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 22170 entries, 0 to 22169
Data columns (total 3 columns):
item_name           22170 non-null object
item_id             22170 non-null int64
item_category_id    22170 non-null int64
dtypes: int64(2), object(1)
memory usage: 519.7+ KB
None
______Shops______
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 2 columns):
shop_name    60 non-null object
shop_id      60 non-null int64
dtypes: int64(1), object(1)
memory usage: 1.0+ KB
None
```

Рис. 1 — Состав данных в основных наборах

307980 является выбросом, поэтому его можно удалить из обучающего набора данных. Так же удаляем строки, в которых значения столбцов «item_price» и «item_cnt_day» меньше нуля.

Тестовая выборка содержит поле «Date», которое необходимо привести к единому определенному формату даты для дальнейшего удобства при формировании тестовых данных для моделей машинного обучения.

4.2. Подготовка обучающей выборки

По условию задачи необходимо предсказывать результаты месячных продаж товаров. Исходные данные содержат ежедневные объемы продаж, поэтому необходимо сформировать выборку для предсказания будущих продаж по меся-

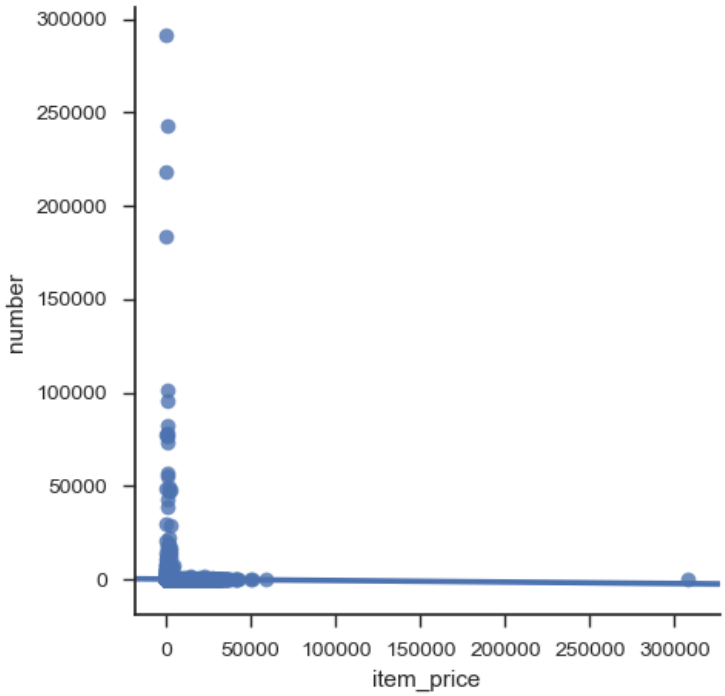

Рис. 2 — График item_price - number

цам.

Набор данных формируется из всех уникальных пар товар-магазин для каждого уникального значения колонки «date_block_num». Таким образом, получаем, что для каждого месяца у нас есть всевозможные сочетания товаров и магазинов.

Далее нам необходимо подсчитать, какое количество каждой единицы товара было куплено в каждом магазине за месяц и добавить эти данные в формируемую выборку. Данную операцию производим путем сложения данных по количеству проданных товаров за день, для текущих магазина (shop_id), товара (item_id) и номера периода (date_block_num).

Следующим шагом будет разбиение полей столбца «Date» на три отдельные столбца:

- month,
- year,
- days_of_month.

и добавление их к формируемой выборке. Последним шагом перед построением моделей будет процесс стандартизации атрибутов

. Данный процесс является общим для многих алгоритмов машинного обучения, так как большинство алгоритмов очень чувствительны к тому, что атрибуты учебной выборки представляли собой нормально распределенные данные. Так, например, многие функции, используемые в функциях алгоритма обучения (ядро опорных векторных машин или регуляризаторы линейных моделей L1 и L2), пред-

полагают, что все атрибуты центрированы вокруг 0 и имеют дисперсию в том же порядке. В случае если атрибут имеет дисперсию, которая на несколько порядков больше, чем другие, он может доминировать над целевым атрибутом и сделать невозможным правильное обучение модели. В данной задаче стандартизации подвергаются все поля обучающей выборки, кроме поля date_block_num.

4.3. Обучение моделей

В качестве моделей для решения данной задачи были выбраны следующие:

- SVC – машина опорных векторов;
- DecisionTreeClassifier – дерево принятия решений;
- AdaBoostClassifier - Этот алгоритм может использоваться в сочетании с несколькими алгоритмами классификации для улучшения их эффективности. Алгоритм усиливает классификаторы, объединяя их в «комитет»;
- RandomForestClassifier - случайный лес;
- ExtraTreesClassifier;
- GradientBoostingClassifier - градиентный бустинг деревьев решений;
- MLPClassifier – многослойный перцептрон;
- KNeighborsClassifier – к - ближайших соседей;
- LogisticRegression – логистическая регрессия;
- LinearDiscriminantAnalysis - классификатор с линейной границей принятия решений, использующий подгонку условных плотностей данных и правила Байеса;

Обучение и тестирование моделей будет производится последовательно на обучающей выборке, разбитой на блоки с использованием алгоритма KFold. В качестве критерия качества моделей будем использовать accuracy алгоритма, то есть точность распознавания данных в тестовых наборах.

4.4. Анализа работы алгоритмов

Итоговый отчет по работе представлен в виде сводного графика Рис. 4.2, где по шкале X указывается точность распознавания данных обученной моделью, а по шкале Y названия тестируемых моделей и таблицы с более точными результатами.

Как видим, 5 алгоритмов:

- SVC
- GradientBoosting
- MultipleLayerPerceptron

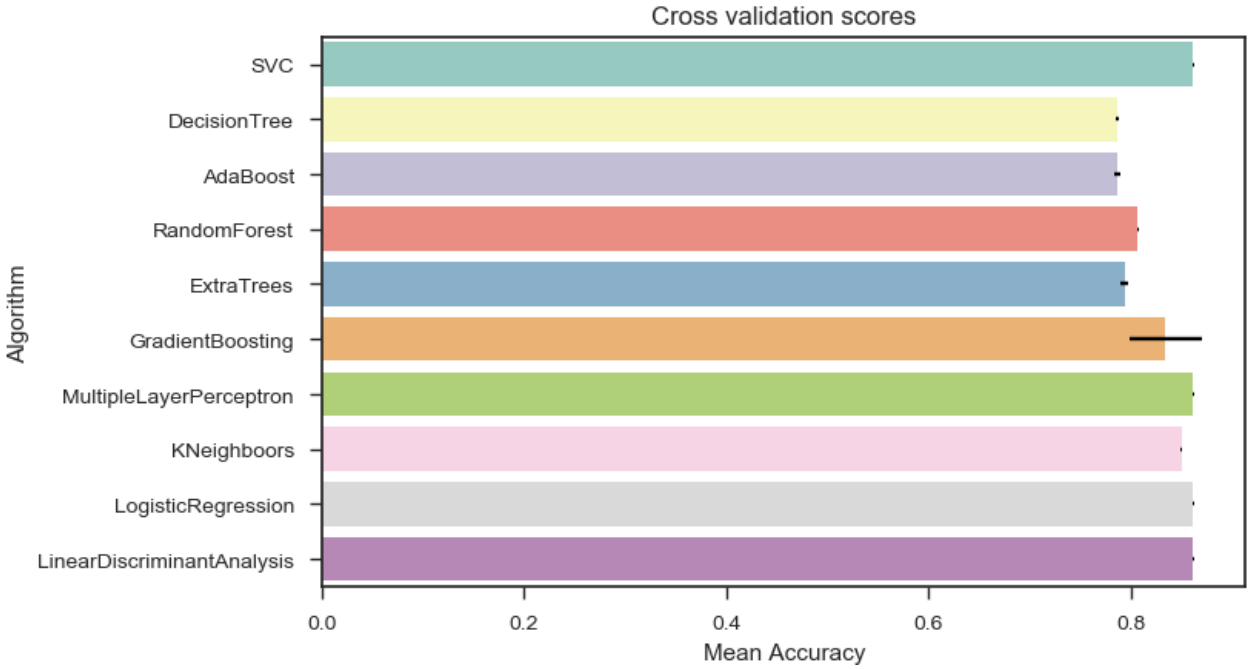

Рис. 3 — Результаты тестирования моделей

– LogisticRegression
– LinearDiscriminantAnalysis

показали наилучшие результаты с точностью распознавания данных около 86Данные результаты получены на алгоритмах, которые не подвергались дополнительной настройке, путем указания каких-либо дополнительных параметров, способных иногда значительно улучшить качество обученных моделей. Отсюда следует, что данные результаты могут быть улучшены при проведении соответствующей настройки каждой из моделей, а так же путем настройки данных обучающей выборки наилучшим образом, подходящим под каждую из моделей.

Таблица 4.2

Результаты тестирования моделей

Алгоритм	**Accuracy**
SVC	0.86240
DecisionTree	0.78894
AdaBoost	0.78611
RandomForest	0.80712
ExtraTrees	0.79450
GradientBoosting	0.86104
MultipleLayerPerceptron	0.86240
KNeighboors	0.84950
LogisticRegression	0.86240
LinearDiscriminantAnalysis	0.86240

Заключение

В ходе работы над решением задачи было выполнено следующее:

- Изучены методы и средства машинного обучения;
- Изучены библиотеки python для работы со средствами машинного обучения;
- Была решена задача “Predict future sales”, с применением моделей машинного обучения;
- Проанализированы результаты работы различных алгоритмов машинного обучения, выявлены те которые решают поставленную задач наилучшим образом.

Список литературы

1. "Predict Future Sales"[Электронный ресурс]. URL: https://www.kaggle.com/c/competitive-data-science-predict-future-sales (дата доступа: 03.07.2018).

2. Видеолекции по машинному обучению [Электронный ресурс]. URL: https://ru.coursera.org/specializations/machine-learning-data-analysis#courses (дата доступа: 03.07.2018).

3. Лекции по курсу "Введение в науку о данных"[Электронный ресурс] URL: https://ru.coursera.org/learn/vvedeniye-v-nauku-o-dannykh (дата доступа: 03.07.2018).

4. Документация по языку программирования Python [Электронный ресурс] URL: https://docs.python.org/3/index.html (дата доступа: 03.07.2018)

5. Документация библиотеки NumPy [Электронный ресурс] URL: http://www.numpy.org (дата доступа: 03.07.2018).

6. Документация библиотеки Pandas [Электронный ресурс] URL: http://pandas.pydata.org/pandas-docs/stable/ (дата доступа: 03.07.2018).

7. Документация библиотеки Seaborn [Электронный ресурс] URL: https://seaborn.pydata.org (дата доступа: 03.07.2018).

8. Документация библиотеки Matplotlib [Электронный ресурс] URL: https://matplotlib.org (дата доступа: 03.07.2018).

9. Документация библиотеки ScikitLearn [Электронный ресурс] URL: http://scikit-learn.org/stable/documentation.html (дата доступа: 03.07.2018).

10. Machine Learning [Электронный ресурс]. URL: http://www.machinelearning.ru/wiki/index.php?title=Machine_Learning#.D0.A0.D0.B5.D0.B3.D1.80.D0.B5.D1.81.D1.81.D0.B8.D1.8F (дата доступа: 03.07.2018).

11. Дьяконов, А. Г. Анализ данных, обучение по прецедентам, логические игры, системы WEKA, RapidMiner и MatLab (практикум на эвм кафедры математических методов прогнозирования). — МАКСПресс, 2010. — 278 с. URL: http://www.machinelearning.ru/wiki/images/7/7e/Dj2010up.pdf

12. Обучение с учителем (supervised learning) [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D1%83%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D0%B5%D0%BC (дата доступа: 03.07.2018).

13. Обучение без учителя (unsupervised learning) [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D0%B1%D0%B5%D0%B7_%D1%83%D1%87%D0%B8%D1%82%D0%B5%D0%BB%D1%8F (дата доступа: 03.07.2018).

14. Частичное обучение (semi-supervised learning) [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%A7%D0%B0%D1%81%D1%82%D0%B8%D1%87%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 (дата доступа: 03.07.2018).

15. Трансдуктивное обучение (transductive learning) [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%A2%D1%80%D0%B0%D0%BD%D1%81%D0%B4%D1%83%D0%BA%D1%82%D0%B8%D0%B2%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5 (дата доступа: 03.07.2018).

16. Обучение с подкреплением (reinforcement learning) [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%9E%D0%B1%D1%83%D1%87%D0%B5%D0%BD%D0%B8%D0%B5_%D1%81_%D0%BF%D0%BE%D0%B4%D0%BA%D1%80%D0%B5%D0%BF%D0%BB%D0%B5%D0%BD%D0%B8%D0%B5%D0%BC (дата доступа: 03.07.2018).

17. Квадратичный дискриминант [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%B2%D0%B0%D0%B4%D1%80%D0%B0%D1%82%D0%B8%D1%87%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA%D1%80%D0%B8%D0%BC%D0%B8%D0%BD%D0%B0%D0%BD%D1%82 (дата доступа: 03.07.2018).

18. Линейный дискриминант Фишера [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D0%B4%D0%B8%D1%81%D0%BA%D1%80%D0%B8%D0%BC%D0%B8%D0%BD%D0%B0%D0%BD%D1%82_%D0%A4%D0%B8%D1%88%D0%B5%D1%80%D0%B0 (дата доступа: 03.07.2018).

19. Метод парзеновского окна [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D0%B0%D1%80%D0%B7%D0%B5%D0%BD%D0%BE%D0%

B2%D1%81%D0%BA%D0%BE%D0%B3%D0%BE_%D0%BE%D0%BA%D0%BD%D0%B0 (дата доступа: 03.07.2018).

20. Метод потенциальных функций [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9 (дата доступа: 03.07.2018).

21. EM-алгоритм [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=EM-%D0%B0%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC (дата доступа: 03.07.2018).

22. Скрытая марковская цепь[Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%A1%D0%BA%D1%80%D1%8B%D1%82%D0%B0%D1%8F_%D0%BC%D0%B0%D1%80%D0%BA%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C (дата доступа: 03.07.2018).

23. Байесовская сеть[Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%91%D0%B0%D0%B9%D0%B5%D1%81%D0%BE%D0%B2%D1%81%D0%BA%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C (дата доступа: 03.07.2018).

24. Метод ближайших соседей[Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B1%D0%BB%D0%B8%D0%B6%D0%B0%D0%B9%D1%88%D0%B8%D1%85_%D1%81%D0%BE%D1%81%D0%B5%D0%B4%D0%B5%D0%B9 (дата доступа: 03.07.2018).

25. Метод потенциальных функций[Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BF%D0%BE%D1%82%D0%B5%D0%BD%D1%86%D0%B8%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D1%85_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B9 (дата доступа: 03.07.2018).

26. Однослойный персептрон[Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9E%D0%B4%D0%BD%D0%BE%D1%81%D0%BB%D0%BE%D0%B9%D0%BD%D1%8B%D0%B9_%D0%BF%D0%B5%D1%80%D1%81%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD (дата доступа: 03.07.2018).

27. Логистическая регрессия[Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%BE%D0%B3%D0%B8%D1%81%D1%82%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%

D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F (дата доступа: 03.07.2018).

28. Машина опорных векторов (SVM) [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%D0%B0_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2 (дата доступа: 03.07.2018).

29. Многослойный персептрон [Электронный ресурс] URL: https://neuralnet.info/chapter/%D0%BF%D0%B5%D1%80%D1%81%D0%B5%D0%BF%D1%82%D1%80%D0%BE%D0%BD%D1%8B (дата доступа: 03.07.2018).

30. Нейронная сеть Кохонена [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%B9%D1%80%D0%BE%D0%BD%D0%BD%D0%B0%D1%8F_%D1%81%D0%B5%D1%82%D1%8C_%D0%9A%D0%BE%D1%85%D0%BE%D0%BD%D0%B5%D0%BD%D0%B0 (дата доступа: 03.07.2018).

31. Логические алгоритмы классификации [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%BE%D0%B3%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B0%D1%8F_%D0%B7%D0%B0%D0%BA%D0%BE%D0%BD%D0%BE%D0%BC%D0%B5%D1%80%D0%BD%D0%BE%D1%81%D1%82%D1%8C (дата доступа: 03.07.2018).

32. Решающее дерево [Электронный ресурс] URL: https://ru.coursera.org/lecture/supervised-learning/rieshaiushchiie-dieriev-ia-HZxD1 (дата доступа: 03.07.2018).

33. Решающий лес [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/Random_forest (дата доступа: 03.07.2018).

34. Кластеризация [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9A%D0%BB%D0%B0%D1%81%D1%82%D0%B5%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F (дата доступа: 03.07.2018).

35. Алгоритм k средних (k-means) [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_k-%D1%81%D1%80%D0%B5%D0%B4%D0%BD%D0%B8%D1%85 (дата доступа: 03.07.2018).

36. Линейная регрессия [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9B%D0%B8%D0%

BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F (дата доступа: 03.07.2018).

37. Нелинейная регрессия [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9D%D0%B5%D0%BB%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D1%80%D0%B5%D0%B3%D1%80%D0%B5%D1%81%D1%81%D0%B8%D1%8F (дата доступа: 03.07.2018).

38. Бустинг [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%91%D1%83%D1%81%D1%82%D0%B8%D0%BD%D0%B3 (дата доступа: 03.07.2018).

39. Бэггинг [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%91%D1%8D%D0%B3%D0%B3%D0%B8%D0%BD%D0%B3 (дата доступа: 03.07.2018).

40. Метод главных компонент [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%B3%D0%BB%D0%B0%D0%B2%D0%BD%D1%8B%D1%85_%D0%BA%D0%BE%D0%BC%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82 (дата доступа: 03.07.2018).

Приложение А

Исходный код

```
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import pandas as pd
from itertools import product
from tqdm import tqdm
from sklearn.model_selection import StratifiedKFold
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import AdaBoostClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.model_selection import cross_val_score
from sklearn.ensemble import VotingClassifier
from sklearn.externals import joblib
pd.options.mode.chained_assignment = None


categories = pd.read_csv("item_categories.csv")
items = pd.read_csv("items.csv")
shops = pd.read_csv("shops.csv")
train = pd.read_csv("sales_train_v2.csv")
sample_submission = pd.read_csv("sample_submission.csv")
test = pd.read_csv("test.csv")
testID = test['ID']


# DataSet info
print("______Train______")
```

```
36 print(train.info())
37 print("___Categories____")
38 print(categories.info())
39 print("______Items______")
40 print(items.info())
41 print("______Shops______")
42 print(shops.info())
43
44 # Shape of train and test datasets
45 print(train.shape)
46 print(test.shape)
47
48 print(train.head())
49 print(test.head())
50
51 # Check NaN and Null values
52 print(train.isnull().sum())
53 print(test.isnull().sum())
54
55 # Delete duplicates
56 print("Train size before dropping duplicates:",  train.shape)
57 train.drop_duplicates(subset=["date", "date_block_num", "shop_id",
58                               "item_id", "item_cnt_day"],
59                       keep="first",
60                       inplace=True)
61 print("Train size after dropping duplicates:", train.shape)
62
63 # checking other csv files on NaN and null values
64 print(shops.isnull().sum())
65 print(items.isnull().sum())
66 print(categories.isnull().sum())
67
68 # Searcing unreasonable values
69 print("____price_____")
70 print("price_min: ", train.item_price.min())
71 print("price_max: ", train.item_price.max())
72 print("price_mean: ", train.item_price.mean())
73 print("price_median: ", train.item_price.median())
74 print("____cnt_____")
75 print("cnt_min: ",train.item_cnt_day.min())
```

```
print("cnt_min: ",train.item_cnt_day.max())
print("cnt_min: ",train.item_cnt_day.mean())
print("cnt_min: ",train.item_cnt_day.median())

dif1 = train.item_price.value_counts().sort_index(ascending=False)
print(dif1)
dif2 = train.item_cnt_day.value_counts().sort_index(ascending=False)
print(dif2)
ds = dif1.to_frame().reset_index()
ds.columns.values[1] = "number"
ds.columns.values[0] = "item_price"
sns.set(style="ticks")
sns.lmplot("item_price", "number", ds)

plt.show()

print("Train size before deleting unreasonable values:", train.shape)
train = train[(train.item_price > 0) & (train.item_price < 300000) &
              (train.item_cnt_day > 0)]
print("Train size after deleting unreasonable values:", train.shape)

# Date format
print('Data format changing start')
train['date'] = pd.to_datetime(train['date'], format='%d.%m.%Y')
print('Data format changing ended')


print("Agregating data")
month_table = []
for block_num in train["date_block_num"].unique():
    month_shops = train[train['date_block_num']==
                        block_num]['shop_id'].unique()
    month_items = train[train['date_block_num']==
                        block_num]['item_id'].unique()
    month_table.append(np.array(list(product(*[month_shops,
                                               month_items,
                                               [block_num]])),
                                dtype='int32'))


```

```
116
117 indexes = ['shop_id', 'item_id', 'date_block_num']
118 month_table = pd.DataFrame(np.vstack(month_table), columns=indexes,
119                            dtype=np.int32)
120
121
122 train['item_cnt_day'] = train['item_cnt_day'].clip(0, 20)
123 # Counting month value
124 month_cnt = train.groupby(indexes)['item_cnt_day'].\
125     agg(['sum']).reset_index().\
126     rename(columns = {'sum': 'item_cnt_month'})
127 month_cnt['item_cnt_month'] = \
128     month_cnt['item_cnt_month'].astype(int)
129
130 new_train = pd.merge(month_table,month_cnt,how='left',on=indexes).\
131     fillna(0)
132 new_train['item_cnt_month'] = new_train['item_cnt_month'].astype(int)
133
134 new_train.sort_values(['date_block_num', 'shop_id', 'item_id'],
135                       inplace=True)
136
137
138 print(train['item_cnt_day'].sum())
139 print(new_train['item_cnt_month'].sum())
140 print(month_cnt['item_cnt_month'].sum())
141
142 print("Aggreating ended")
143
144
145 # adding item_category_id
146 new_train = new_train.merge(items[['item_id', 'item_category_id']],
147                             on=['item_id'], how = 'left')
148 test = test.merge(items[['item_id', 'item_category_id']],
149                   on=['item_id'], how = 'left')
150
151 new_train = new_train.drop_duplicates(subset=["shop_id",
152                                               "item_id",
153                                               "date_block_num"] )
154
155 print('_____Train______')
```

```
156 print(new_train)
157 print("_____Test_______")
158 print(test)
159
160
161 print(new_train.reset_index().groupby(['item_id', 'date_block_num',
162                                        'shop_id']).mean())
163
164
165
166
167 def TrainTestUnion(new_train, test):
168     test["date_block_num"] = 34
169     train_test = pd.concat([new_train, test], axis=0)
170     train_test = train_test.drop(columns=['ID'])
171     print(train_test)
172     train_test = train_test.fillna(0)
173     return train_test
174
175 index = ['shop_id', 'item_id', 'item_category_id', 'date_block_num']
176
177
178 # Date adding
179 def DataFeatures(train_test):
180     dates_train = train[['date', 'date_block_num']].drop_duplicates()
181     dates_test = dates_train[dates_train['date_block_num'] == 34-12]
182     dates_test['date_block_num'] = 34
183     dates_test['date'] = dates_test['date'] + pd.DateOffset(years=1)
184     dates_all = pd.concat([dates_train, dates_test])
185
186     dates_all['dow'] = dates_all['date'].dt.dayofweek
187     dates_all['year'] = dates_all['date'].dt.year
188     dates_all['month'] = dates_all['date'].dt.month
189     dates_all = pd.get_dummies(dates_all, columns=['dow'])
190     dow_col = ['dow_' + str(x) for x in range(7)]
191     date_features = dates_all.groupby(['year', 'month',
192                                        'date_block_num'])[dow_col].\
193         agg('sum').reset_index()
194     date_features['days_of_month'] = date_features[dow_col].sum(axis=1)
195     date_features['year'] = date_features['year'] - 2013
```

```
196
197     date_features = date_features[['month', 'year', 'days_of_month',
198                                    'date_block_num']]
199
200     train_test = train_test.merge(date_features, on='date_block_num',
201                                   how='left')
202     print(train_test)
203     date_columns = date_features.columns.difference(set(index))
204     print(date_columns)
205     return train_test, date_columns
206
207 # Scale features
208 def ScaleFeatures(train_test, date_columns, index):
209     train = train_test[train_test['date_block_num']!=
210                        train_test['date_block_num'].max()]
211     test = train_test[train_test['date_block_num']==
212                       train_test['date_block_num'].max()]
213     scalar = StandardScaler()
214     col = ['date_block_num']
215     feature_columns = list(set(index +
216                                list(date_columns)).difference(col))
217     train[feature_columns] = scalar.\
218         fit_transform(train[feature_columns])
219     test[feature_columns] = scalar.transform(test[feature_columns])
220     train_test = pd.concat([train, test], axis=0)
221     # print(train_test)
222     return train_test
223
224
225 def ModelBuilder(train_test):
226     kfold = StratifiedKFold(n_splits=5, shuffle=True)
227     classifiers = []
228     random_state = 0
229     classifiers.append(SVC(random_state=random_state))
230     classifiers.append(DecisionTreeClassifier(random_state=random_state
231     classifiers.append(AdaBoostClassifier(DecisionTreeClassifier(
232         random_state=random_state), random_state=random_state,
233                                               learning_rate=0.1))
234     classifiers.append(RandomForestClassifier(random_state=random_state
235     classifiers.append(ExtraTreesClassifier(random_state=random_state))
```

```
    classifiers.append(GradientBoostingClassifier(
        random_state=random_state))
    classifiers.append(MLPClassifier(random_state=random_state))
    classifiers.append(KNeighborsClassifier())
    classifiers.append(LogisticRegression(random_state=random_state))
    classifiers.append(LinearDiscriminantAnalysis())
    cv_results = []
    train_test = train_test[train_test['date_block_num']!=
                            train_test['date_block_num'].max()]
    X_train = train_test[['item_category_id', 'item_id', 'shop_id',
                          'month', 'year', 'days_of_month']][:100000]
    # print('_________X_train_______ ')
    # print(X_train)
    Y_train = (train_test['item_cnt_month'][:100000])
    # print('_________Y_train_______ ')
    # print(Y_train)
    for classifier in tqdm(classifiers):
        cv_results.append(cross_val_score(classifier,
                                          X_train,
                                          y=Y_train,
                                          scoring="accuracy",
                                          cv=kfold,
                                          n_jobs=1))
    cv_means = []
    cv_std = []
    for cv_result in cv_results:
        cv_means.append(cv_result.mean())
        cv_std.append(cv_result.std())

    cv_res = pd.DataFrame(
        {"CrossValMeans": cv_means,
         "CrossValerrors": cv_std,
         "Algorithm": ["SVC", "DecisionTree", "AdaBoost",
                       "RandomForest", "ExtraTrees",
                        "GradientBoosting",
                        "MultipleLayerPerceptron", "KNeighboors",
                        "LogisticRegression",
                        "LinearDiscriminantAnalysis"]})

    print(cv_res)
```

```
276     g = sns.barplot("CrossValMeans", "Algorithm",
277                     data=cv_res,
278                     palette="Set3",
279                     orient="h", **{'xerr': cv_std})
280     g.set_xlabel("Mean Accuracy")
281     g = g.set_title("Cross validation scores")
282     plt.show()
283
284 train_test = TrainTestUnion(new_train, test)
285 train_test, date_columns = DataFeatures(train_test)
286 train_test = ScaleFeatures(train_test, date_columns, index)
287
288 ModelBuilder(train_test)
```