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 «House Prices: Advanced Regression Techniques»

по направлению
09.03.03 Прикладная информатика

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

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

Студент группы ПИЭ-31БО

_____ Д. С. Топникова
«_____»_____ 2018 г.

Ярославль, 2018

Реферат

Объем 29 с., 5 гл., 11 рис., 1 табл., 14 источников, 1 прил.

Ключевые слова: **машинное обучение.**

Содержание

Введение

Для анализа методов машинного обучения была выбрана задача «House Prices: Advanced Regression Techniques» с сайта kaggle.com, который является платформой для проведения конкурсов по машинному обучению. Цель задачи – определение конечной стоимости жилых помещений на основе предложенных параметров. При решении задачи мы будем использовать несколько алгоритмов машинного обучения, чтобы проанализировать эффективность каждого из них и выявить лучшие 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,
    - практическая часть,
    - результаты и обсуждения;
6) заключение;
7) список использованных источников (список литературы);
8) приложения.

2. О задаче

2.1. Постановка задачи

Задача, которую мы решаем звучит следующим образом: «Попросите покупателя дома описать дом своей мечты, и они, вероятно, не начнут с высоты потолка, подвала или близости к железной дороге Восток-Запад. Но анализ представленных данных доказывает, что гораздо больше влияет на цену количество спален, чем забор белого цвета. На основе 79 параметров, описывающим (почти) каждый аспект жилых домов в Эймсе, штат Айова, этот конкурс ставит перед вами задачу предсказать конечную цену каждого дома.»
Для решения задачи мы используем данные с сайта kaggle.com. Они представляют собой:

- train.csv – обучающий набор данных
- test.csv - тестовый набор
- data_description.txt - полное описание каждого столбца
- sample_submission.csv – пример вывода данных

Наборы данных состоят из 79 полей, характеризующих жилищные помещения. Примеры некоторых полей:

- SalePrice - цена продажи недвижимости в долларах. Это переменная, которую вы пытаетесь предсказать.
- Heating – тип отопления;
- Kitchen – количество кухонь;
- KitchenQual - качество кухни;
- PavedDrive - асфальтированная подъездная дорожка;
- И т.д.

2.2. Используемые програмные средства

Задача будет решена с использованием языка программирования Python, а также с использованием подключаемых библиотек. В ходе решения будем использовать библиотеки:

- NumPy (для первичной обработки исходных данных и приозведения нужных расчетов);
- Pandas (также для первичной обработки);
- Seaborn (для визуализации);
- Matplotlib (для визуализации);

– Scikit-learn (для решения задачи методами машинного обучения).

3. Теоретическая часть

3.1. Машинное обучение

Машинное обучение (MACHINE LEARNING, ML) — класс методов искусственного интеллекта, характерной чертой которых является не прямое решение задачи, а обучение в процессе применения решений множества сходных задач. Для построения таких методов используются средства математической статистики, численных методов, методов оптимизации, теории вероятностей, теории графов, различные техники работы с данными в цифровой форме.
Выделяют два типа обучения:

- Обучение по прецедентам (индуктивное обучение) основано на выявлении общих закономерностей по частным эмпирическим данным.
- Дедуктивное обучение предполагает формализацию знаний экспертов и их перенос в компьютер в виде базы знаний.

Общая постановка задачи обучения по прецедентам:
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.
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. Алгоритмы машинного обучения

Машинное обучение основано на алгоритмах. Эти алгоритмы можно разделить на три основные категории:

1. Контролируемые алгоритмы: набор обучающих данных имеет входные данные, а также желаемый результат. Во время обучения модель будет корректировать свои переменные для сопоставления входных данных с соответствующим выходом.
2. Неконтролируемые алгоритмы: в этой категории нет желаемого результата. Алгоритмы будут группировать набор данных на разные группы.
3. Алгоритмы с подкреплением: эти алгоритмы обучаются на готовых решениях. На основе этих решений алгоритм будет обучаться на основе успеха/ошибки результата. В конечном счете алгоритм сможет давать хорошие прогнозы.

3.2.1. Линейная регрессия

Алгоритм линейной регрессии будет использовать точки данных для поиска оптимальной линии для создания модели. Линию можно представить уравнением y = m*x + c, где y – зависимая переменная, а x – независимая переменная. Базовые теории исчисления применяются для определения значений для m и c с использованием заданного набора данных. Существует два типа линейной регрессии: простая линейная регрессия с одной независимой переменной и множественная линейная регрессия, где используется несколько независимых переменных.

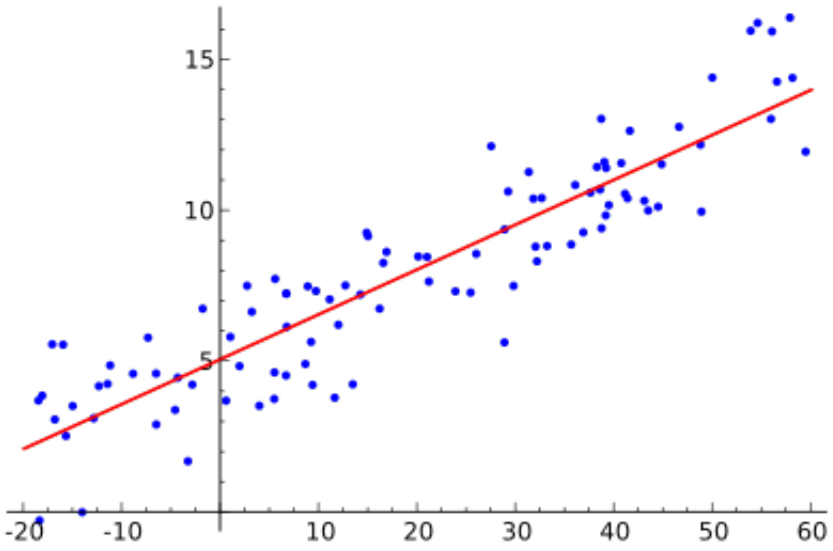

Рис. 1 — Линейная регрессия

3.2.2. Метод опорных векторов

Он принадлежит к алгоритму классификационного типа. Алгоритм будет разделять точки данных, используя линию. Эта линия должна быть максимально удаленной от ближайших точек данных в каждой из двух категорий.

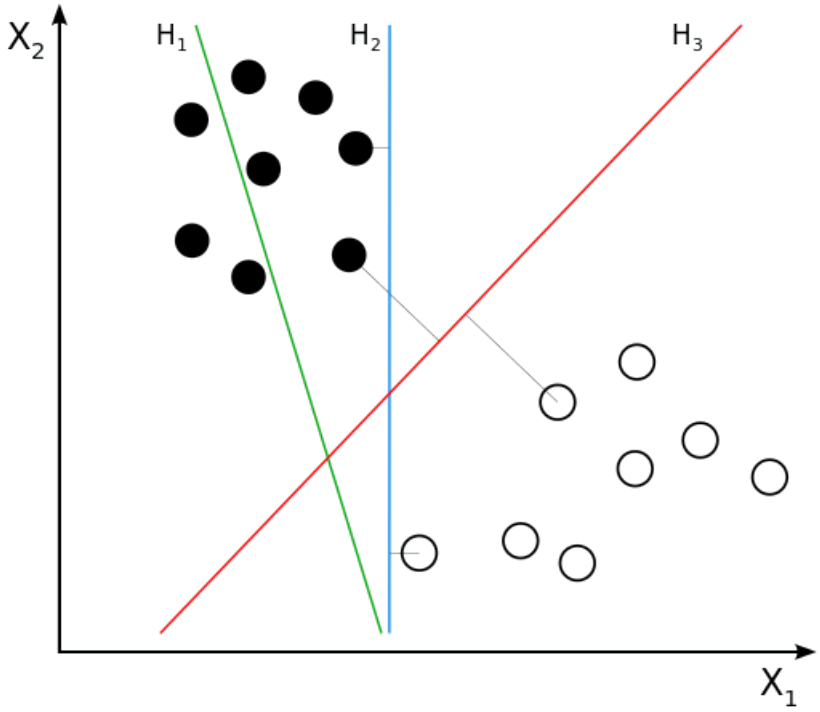

Рис. 2 — Метод опорных векторов

На рисунке выше красная линия подходит лучше всего, так как она больше всего удалена от всех точек. На основе этой линии данные делятся на две группы.

3.2.3. Метод k-ближайших соседей (KNN - k-nearest neighbors)

Это простой алгоритм, который предсказывает неизвестную точку данных на основе её k ближайших соседей. Значение k здесь критически важный фактор, который определяет точность предсказания. Ближайшая точка определяется исходя из базовых функций расстояния, вроде Евклидовой.

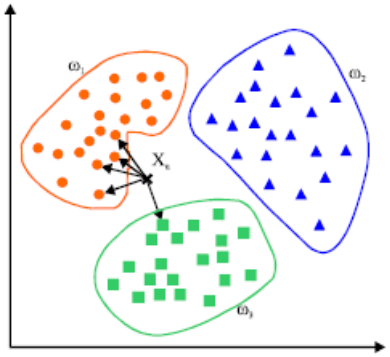

Рис. 3 — Метод k-ближайших соседей

Однако этот алгоритм требует высокой вычислительной мощности, и нам необходимо сначала нормализовать данные, чтобы каждая точка данных была в том же диапазоне.

3.2.4. Логистическая регрессия (logistic regression)

Логистическая регрессия используется, когда ожидается дискретный результат, например, возникновение какого-либо события (пойдет дождь или нет). Обычно логистическая регрессия использует функцию, чтобы поместить значения в определенный диапазон.

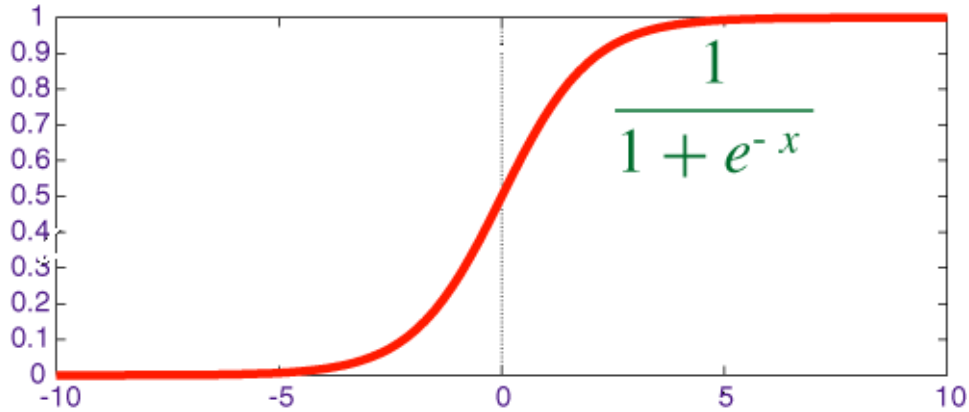

Рис. 4 — Логистическая регрессия

“Сигмоид” — это одна из таких функций в форме буквы S, которая используется для бинарной классификации. Она конвертирует значения в диапазон от 0 до 1, что является вероятностью возникновения события.
$y = e^{b_0+b_1*x}/(1+e^{b_0+b_1*x})$

Выше находится простое уравнение логистической регрессии, где b0 и b1 — это постоянные.Во время обучения значения для них будут вычисляться таким образом, чтобы ошибка между предсказанием и фактическим значением становилась минимальной.

3.2.5. Дерево решений (decision tree)

Этот алгоритм распределяет данные на несколько наборов на основе каких-либо свойств (независимых переменных). Обычно этот алгоритм используется для решения проблем классификации. Категоризация используется на основе методов вроде Джини, Хи-квадрат, энтропия и так далее. Возьмем группу людей и используем алгоритм дерева решений, чтобы понять, кто из них имеет кредитную карту. Например, возьмем возраст и семейное положение в качестве свойств. Если человеку больше 30 лет и он/она замужем или женат, то вероятность того, что у них есть кредитная карта, выше.

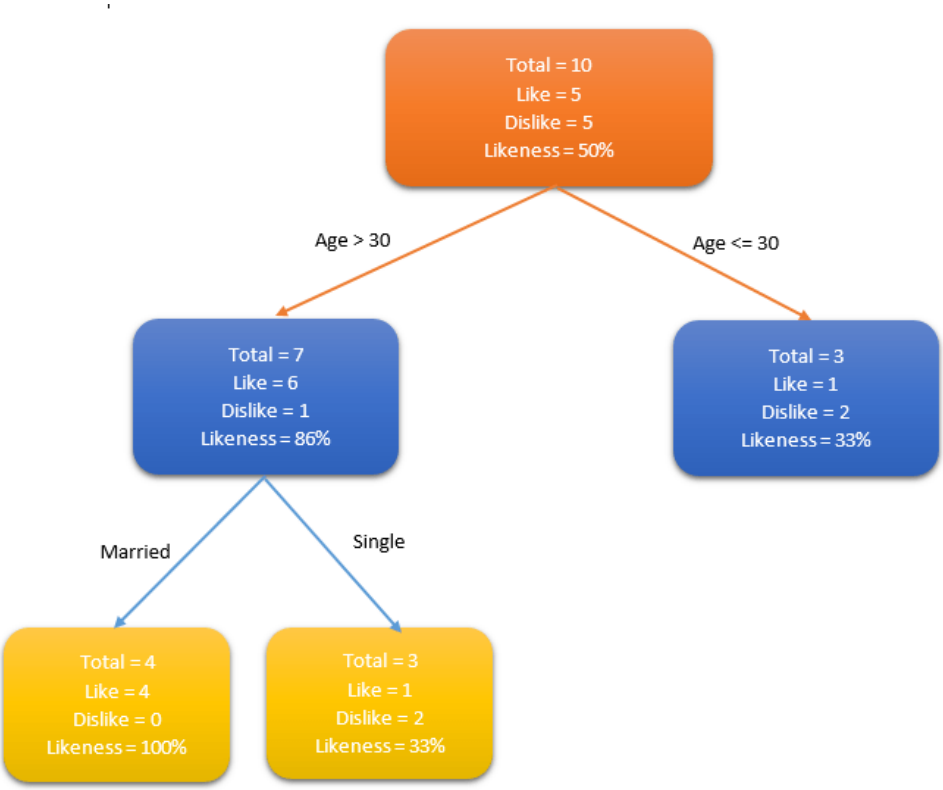

Рис. 5 — Дерево решений

Дерево решений можно расширить и добавить подходящие свойства и категории. Например, если человек женат и ему больше 30 лет, то он, вероятно, имеет кредитную карту. Данные тестирования будут использоваться для создания этого дерева решений.

3.2.6. Метод k- средних (k-means)

Это алгоритм, работающий без присмотра, который предоставляет решение проблемы группировки. Алгоритм формирует кластеры, которые содержат гомогенные точки данных.

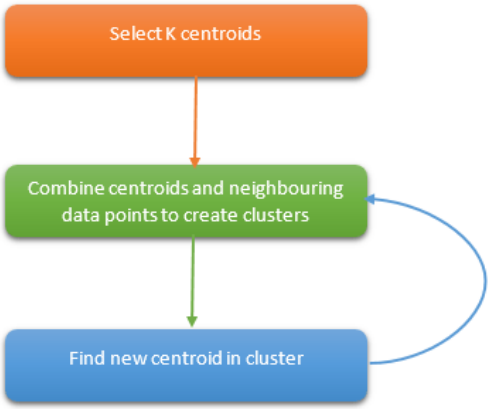

Рис. 6 — метод k-средних

Входные данные алгоритма — это значение k. На основе этого алгоритм выбирает k центроидов. Затем центроид и его соседние точки данных формируют кластер, а внутри каждого кластера создается новый центроид. Потом точки данных, близкие к новому центроиду снова комбинируются для расширения кластера. Процесс продолжается до тех пор, пока центроиды не перестанут изменяться.

3.2.7. Случайный лес (Random Forest)

Random Forest — это коллекция деревьев решений. Каждое дерево пытается оценить данные, и это называется голосом. В идеале мы рассматриваем каждый голос от каждого дерева и выбираем классификацию с большим количеством голосов.

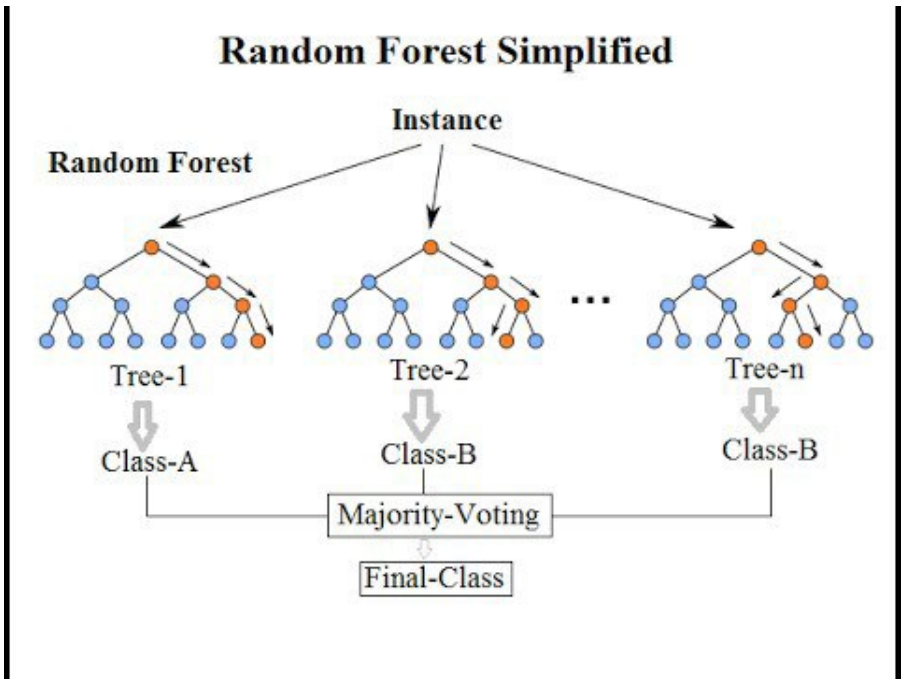

Рис. 7 — Слуйчайный лес

3.2.8. Наивный байесовский классификатор (Naive Bayes)

Этот алгоритм основан на теореме Байеса. Благодаря этому наивный байесовский классификатор можно применить, только если функции независимы друг от друга. Если мы попытаемся предсказать вид цветка на основе длины и ширины его лепестка, мы сможем использовать этот алгоритм, так как функции не зависят друг от друга. Наивный байесовский алгоритм также является классификационным. Он используется, когда проблема содержит несколько классов.

3.2.9. Алгоритмы сокращения размеров (dimensional reduction algorithms)

Некоторые наборы данных содержат много переменных, которыми сложно управлять. Особенно сейчас, когда системы собирают большое количество детализированных данных из разных источников. В таких случаях данные могут содержать тысячи переменных, многие из которых будут не нужны. В таком случае почти невозможно определить переменные, которые будут иметь наибольшее влияние на наше предсказание. в таких ситуациях используются алгоритмы сокращения размеров. Они применяют алгоритмы вроде случайного леса или дерева решений, чтобы определить самые важные переменные.

3.2.10. Алгоритмы усиления градиента (gradient boosting algorithms)

Алгоритм усиления градиента использует множество слабых алгоритмов, чтобы создать более точный, стабильный и надежный алгоритм. Существует несколько алгоритмов усиления градиента:

– GBoost — использует линейные алгоритмы и дерево решений

– ightGBM — использует только алгоритмы, основанные на деревьях

Особенность алгоритмов усиления градиента — это их высокая точность. Более того, алгоритмы вроде LightGBM имеют и высокую производительность.

3.3. Практические сферы применения

Целью машинного обучения является частичная или полная автоматизация решения сложных профессиональных задач в разнообразных областях деятельности человека. Машинное обучение имеет большой список практических сфер его применения. Например:

– Виртуальные ассистенты в смартфонах – Siri , Cortana, Google Now.
– Распознавание жестов
– Распознавание образов – определение лиц на фотографиях в соц. сетях (в частности соц. сеть ВКонтакте)
– Обнаружение спама – фильтрация сообщений в соц. сетях, а также в почтовых ящиках пользователей.
– Мессенджеры – Facebook Messenger – один из самых интересных продуктов крупнейшей социальной платформы в мире, мессенджер стал своеобразной лабораторией чатботов. При общений с некоторыми из них сложно понять, что ты разговариваешь не с человеком.
– Ранжирование данных – используется при сортировке поисковых запросов, а также для пользователей соц. сетей: сейчас пользователи могут сортировать отображаемый контент по популярности или по времени публикации. Искусственный интеллект анализирует каждый пост в реальном времени и оценивает его по нескольким показателям. Алгоритм в первую очередь показывает те записи, которые с большей вероятностью понравятся пользователю, при этом выбор основывается на его личных предпочтениях. Данная технология используется в соц. сетях ВКонтакте и Twitter.
– И многие другие сферы практического применения.

4. Практическая часть

Решение задачи «House Prices: Advanced Regression Techniques» можно разделить на два этапа:

1. Анализ и подготовка исходных данных
2. Обучение моделей.

А после завершения работы программы провести анализ полученных результатов. Рассмотрим каждый из этапов более подробно.
1) Анализ и подготовка исходных данных. При рассмотрении файла train.csv с исходными данными можно увидеть, что в этом датасете достаточно много ячеек, которые имеют значения «NA», которые Python воспринимает как «None», что не является верным, т.к. согласно документации на данные, значение «NA» являются значимыми, а не пустыми ячейками. На рисунке 8 представлена часть статистика значений в датасете (1460 – общее число строк в таблице; далее – название столбца с числом «ненулевых» значений и тип данных в ячейках текущего столбца).

```
RangeIndex: 1460 entries, 0 to 1459
Data columns (total 81 columns):
Id              1460 non-null int64
MSSubClass      1460 non-null int64
MSZoning        1460 non-null object
LotFrontage     1201 non-null float64
LotArea         1460 non-null int64
Street          1460 non-null object
Alley           91 non-null object
LotShape        1460 non-null object
LandContour     1460 non-null object
Utilities       1460 non-null object
LotConfig       1460 non-null object
LandSlope       1460 non-null object
Neighborhood    1460 non-null object
```

Рис. 8 — Статистика значений в наборе данных train.csv

Ниже приведён фрагмент кода, с помощью которого мы заменяем «NA» на значимые значения (Рисунок 9).

```
26      # Преобразование Nane (NA)
27      # Alley: NA на No
28      print(train.info())
29      train.loc[:, "Alley"] = train.loc[:, "Alley"].fillna("None")
30      # BedroomAbvGr: NA на 0
31      train.loc[:, "BedroomAbvGr"] = train.loc[:, "BedroomAbvGr"].fillna(0)
32      # BsmtQual: Na na No or 0
33      train.loc[:, "BsmtQual"] = train.loc[:, "BsmtQual"].fillna("No")
34      train.loc[:, "BsmtCond"] = train.loc[:, "BsmtCond"].fillna("No")
35      train.loc[:, "BsmtExposure"] = train.loc[:, "BsmtExposure"].fillna("No")
36      train.loc[:, "BsmtFinType1"] = train.loc[:, "BsmtFinType1"].fillna("No")
37      train.loc[:, "BsmtFinType2"] = train.loc[:, "BsmtFinType2"].fillna("No")
38      train.loc[:, "BsmtFullBath"] = train.loc[:, "BsmtFullBath"].fillna(0)
39      train.loc[:, "BsmtHalfBath"] = train.loc[:, "BsmtHalfBath"].fillna(0)
40      train.loc[:, "BsmtUnfSF"] = train.loc[:, "BsmtUnfSF"].fillna(0)
41      # CentralAir Na na No
42      train.loc[:, "CentralAir"] = train.loc[:, "CentralAir"].fillna("N")
43      # Condition Na na Norm
44      train.loc[:, "Condition1"] = train.loc[:, "Condition1"].fillna("Norm")
45      train.loc[:, "Condition2"] = train.loc[:, "Condition2"].fillna("Norm")
46      # External stuff : NA na TA
47      train.loc[:, "ExterCond"] = train.loc[:, "ExterCond"].fillna("TA")
48      train.loc[:, "ExterQual"] = train.loc[:, "ExterQual"].fillna("TA")
49      # Fence : Na na No
50      train.loc[:, "Fence"] = train.loc[:, "Fence"].fillna("No")
51      # FireplaceQu : Na na No or 0
```

Рис. 9 — Обработка значений «NA»

Следующее, что мы корректируем в исходном файле – значения, которые сильно отличаются от основной выборки. Согласно рекомендациям рекомендациям автора датасета, необходимо установить значение поля «GrLivArea» меньше 4000. Значения больше 4000 мы удаляем из датасета. На рисунке 10 представлен график зависимости стоимости жилого помещения от параметра «GrLivArea».

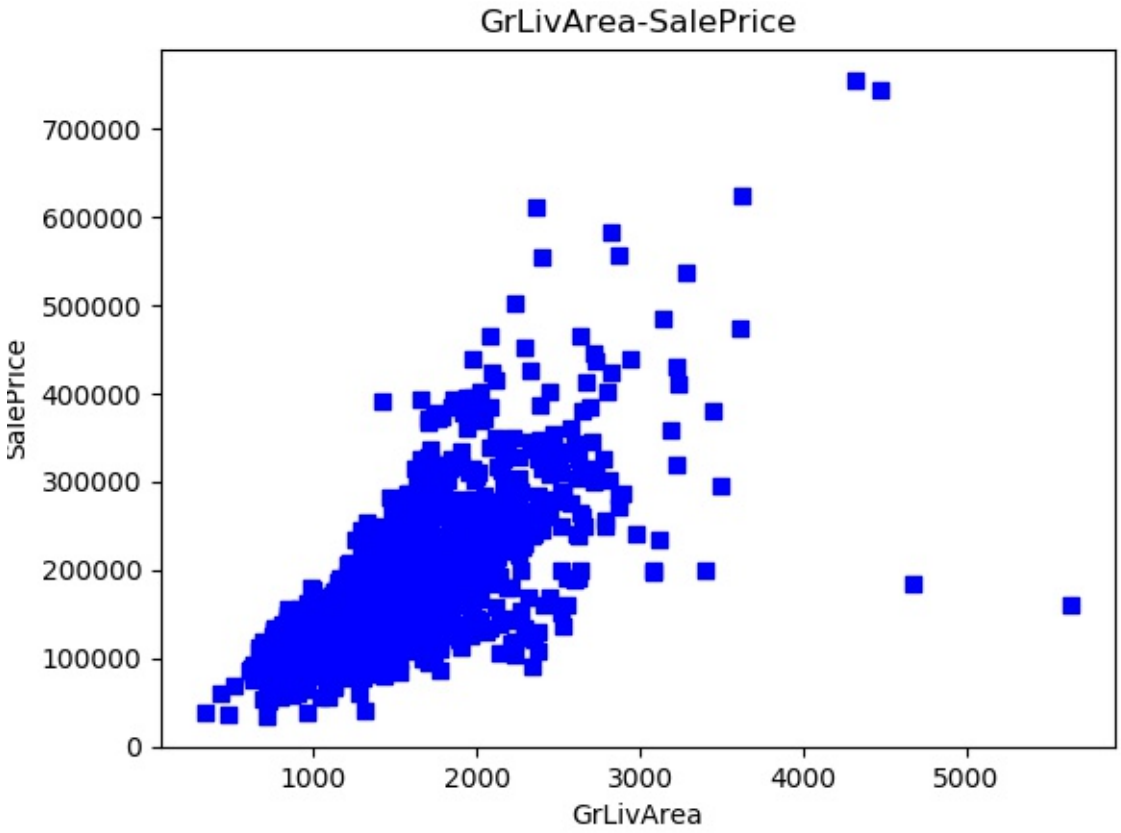

Рис. 10 — Зависимость стоимости жилого помещения от параметра «GrLivArea»

Затем мы производим логарифмирование стоимости домов, т.к. обучение удобнее производить на логарифмированных данных, и заменяем категорийные признаки на числовые, используя класс LabelEncoder из библиотеки sklearn. На этом подготовка исходных данных закончена.

2)Обучение Для решения данной задачи используется 7 алгоритмов:

- Lasso – линейная регрессия с регуляризацией для того, чтобы избежать переобучения (менее значимые признаки перестают влиять на модель).
- ElasticNetCV – модель регрессии с двумя регуляризаторами. Данная модель будет стремиться выбирать больше переменных, следовательно, приводит к более крупным моделям, но также более точным в целом. В частности, Lasso очень чувствителен к корреляции между функциями и может случайно выбирать одну из 2 очень коррелированных информационных характеристик, в то время как ElasticNetCV будет с большей вероятностью выбирать и то, и другое, что должно привести к более стабильной модели.
- SVR – регрессионный метод опорных векторов. Данный алгоритм разделяет данные, используя линию. Эта линия должна быть максимально удаленной от ближайших точек данных в каждой из категорий.
- NeighborsRegressor - алгоритм, который предсказывает неизвестную точку

данных на основе её k ближайших соседей. Значение k здесь критически важный фактор, который определяет точность предсказания. Ближайшая точка определяется исходя из базовых функций расстояния.

- DecisionTreeRegressor - алгоритм распределяет данные на несколько наборов на основе каких-либо свойств. Обычно этот алгоритм используется для решения проблем классификации.
- RandomForestRegressor - Random Forest — это коллекция деревьев решений. Каждое дерево пытается оценить данные, и это называется голосом. В идеале рассматривается каждый голос от каждого дерева и выбирается классификация с большим количеством голосов.
- GradientBoostingRegressor - алгоритм усиления градиента использует множество слабых алгоритмов, чтобы создать более точный, стабильный и надежный алгоритм. Особенность алгоритмов усиления градиента — это их высокая точность.

В результате работы этих алгоритмов мы получаем коэффициент среднеквадратической ошибки – CrossValMeans, который показывает отклонение полученного в результате работы алгоритма значения стоимости дома от его реальной стоимости.
Ниже представлена таблица с алгоритмами и значением их среднеквадратической ошибки.

Таблица 4.1

Значения среднеквадратической ошибки

Algorithm	**CrossValMeans**
LinearRegression	0.395600
ElasticNeetCV	0.123882
SVR	0.159193
KNeighborsRegressor	0.181701
DecisionTreeRegressor	0.201995
RandomForestRegressor	0.145954
GradientBoostingRegressor	0.122972

5. Результаты и обсуждения

Для оценки эффективность работы каждого из алгоритмов я использую коэффициент среднеквадратической ошибки – CrossValMeans. В предыдущем разделе была приведена таблица с значением данного коэффициента для каждого алгоритма. Чтобы более наглядно выглядели данные значения рассмотрим следующий график и сделаем соответствующие выводы (Рис.11).

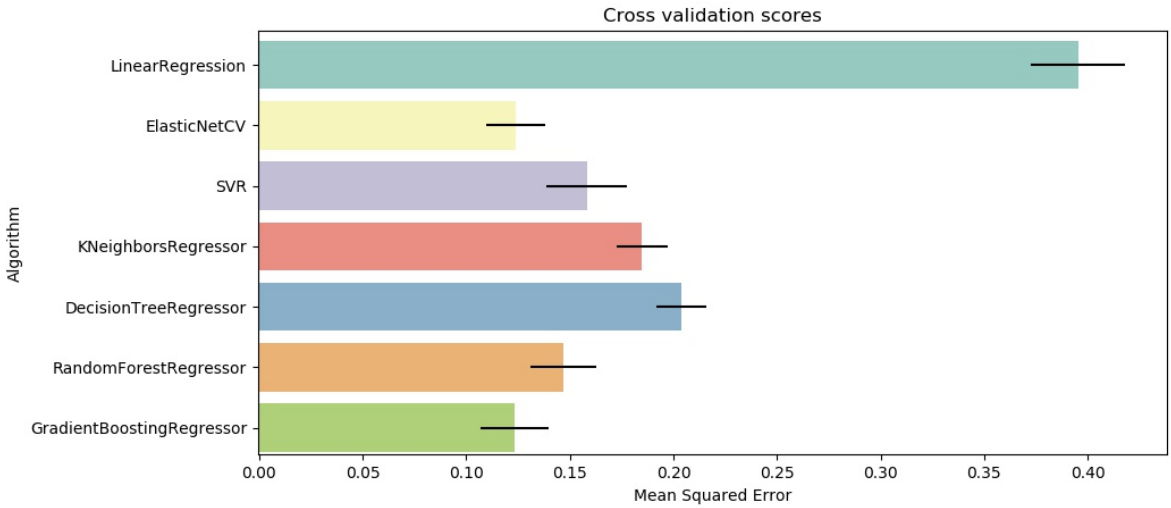

Рис. 11 — Значения среднеквадратической ошибки

На основе этого графика можно сделать следующие выводы:

- Лучше всего с поставленной задачей справился GradientBoostingRegressor
- Хуже всех отработал LinearRegression.

Заключение

В ходе решения выбранной задачи «House Prices: Advanced Regression Techniques» были изучены методы машинного обучения, применены 7 алгоритмов машинного обучения. Выполнен анализ результатов работы алгоритмов с определением лучшего и худшего алгоритмов для данной задачи.

Список литературы

[1] House Prices: Advanced Regression Techniques [Электронный ресурс] URL: https://www.kaggle.com/c/house-prices-advanced-regression-techniques (дата доступа: 03.07.2018).

[2] Машинное обучение [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%88%D0%B8%D0%BD%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).

[3] Методы отбора фич [Электронный ресурс] URL: https://habr.com/post/264915/ (дата доступа: 03.07.2018).

[4] Константин Воронцов. Курс «машинное обучение» школы анализа данных компании Яндекс.(видеолекции) [Электронный ресурс] URL: https://www.coursera.org/learn/vvedenie-mashinnoe-obuchenie (дата доступа: 25.06.2018).

[5] Machine Learning [Электронный ресурс] URL: http://www.machinelearning.ru/wiki/index.php?title=Machine_Learning (дата доступа: 03.07.2018).

[6] 10 алгоритмов машинного обучения [Электронный ресурс] URL: https://apptractor.ru/info/articles/desyat-algoritmov-mashinnogo-obucheniya-kotoryie-vam-nuzhno-znat.html (дата доступа: 03.07.2018).

[7] Примеры использования машинного обучения [Электронный ресурс] URL: https://rusability.ru/internet-marketing/budushhee-mashinnogo-obucheniya-i-10-krutyh-primerov-ego-ispolzovani (дата доступа: 03.07.2018).

[8] Александр Дьяконов. Лекция «Введение в scikit-Learn» МГУ им. М.В. Ломоносова (курс за 2015 год). [Электронный ресурс] URL: https://alexanderdyakonov.files.wordpress.com/2015/04/ama2015_scikit.pdf (дата доступа: 03.07.2018).

[9] Орельен Жерон Прикладное машинное обучение с помощью Scikit-Learn и TensorFlow: концепции, инструменты и техники для создания интеллектуальных систем// издательский дом "Вильямс 02.2018 с.199-222.

[10] Метод Lasso [Электронный ресурс] URL: http://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Lasso.html (дата доступа: 03.07.2018).

[11] Метод ElasticNetCV [Электронный ресурс] URL: http://scikit-learn.org/dev/modules/generated/sklearn.linear_model.ElasticNetCV.html#sklearn.linear_model.ElasticNetCV (дата доступа: 03.07.2018).

[12] Siri URL: https://ru.wikipedia.org/wiki/Siri (дата доступа: 03.07.2018)

[13] Cortana URL: https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D1%80%D1%82%D0%B0%D0%BD%D0%B0_(%D0%B3%D0%BE%D0%BB%D0%BE%D1%81%D0%BE%D0%B2%D0%B0%D1%8F_%D0%BF%D0%BE%D0%BC%D0%BE%D1%89%D0%BD%D0%B8%D1%86%D0%B0) (дата доступа: 03.07.2018)

[14] Google Now URL: https://ru.wikipedia.org/wiki/Google_Now (дата доступа: 03.07.2018)

Приложение А

Исходный код

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder
from sklearn.model_selection import KFold
from sklearn.linear_model import ElasticNetCV
from sklearn.svm import SVR
from sklearn.neighbors import KNeighborsRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.linear_model import Lasso
from tqdm import tqdm
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import StandardScaler

sample_submission = pd.read_csv("./sample_submission.csv")
test = pd.read_csv("./test.csv")
train = pd.read_csv("./train.csv")









# Transformation Nane (NA)
# Alley: NA PSP° No
# print(train.info())
train.loc[:, "Alley"] = train.loc[:, "Alley"].fillna("None")
# BedroomAbvGr: NA PSP° 0
train.loc[:, "BedroomAbvGr"] = train.loc[:, "BedroomAbvGr"].fillna(0)
# BsmtQual: Na na No or 0
train.loc[:, "BsmtQual"] = train.loc[:, "BsmtQual"].fillna("No")
train.loc[:, "BsmtCond"] = train.loc[:, "BsmtCond"].fillna("No")
train.loc[:, "BsmtExposure"] = train.loc[:, "BsmtExposure"].\
```

```
    fillna("No")
train.loc[:, "BsmtFinType1"] = train.loc[:, "BsmtFinType1"].\
    fillna("No")
train.loc[:, "BsmtFinType2"] = train.loc[:, "BsmtFinType2"].\
    fillna("No")
train.loc[:, "BsmtFullBath"] = train.loc[:, "BsmtFullBath"].fillna(0)
train.loc[:, "BsmtHalfBath"] = train.loc[:, "BsmtHalfBath"].fillna(0)
train.loc[:, "BsmtUnfSF"] = train.loc[:, "BsmtUnfSF"].fillna(0)
# CentralAir Na na No
train.loc[:, "CentralAir"] = train.loc[:, "CentralAir"].fillna("N")
# Condition Na na Norm
train.loc[:, "Condition1"] = train.loc[:, "Condition1"].fillna("Norm")
train.loc[:, "Condition2"] = train.loc[:, "Condition2"].fillna("Norm")
# External stuff : NA na TA
train.loc[:, "ExterCond"] = train.loc[:, "ExterCond"].fillna("TA")
train.loc[:, "ExterQual"] = train.loc[:, "ExterQual"].fillna("TA")
# Fence : Na na No
train.loc[:, "Fence"] = train.loc[:, "Fence"].fillna("No")
# FireplaceQu : Na na No or 0
train.loc[:, "FireplaceQu"] = train.loc[:, "FireplaceQu"].fillna("No")
train.loc[:, "Fireplaces"] = train.loc[:, "Fireplaces"].fillna(0)
# Functional : Na na Typ
train.loc[:, "Functional"] = train.loc[:, "Functional"].fillna("Typ")
# GarageType etc : NA na No or 0
train.loc[:, "GarageType"] = train.loc[:, "GarageType"].fillna("No")
train.loc[:, "GarageFinish"] = train.loc[:, "GarageFinish"].\
    fillna("No")
train.loc[:, "GarageQual"] = train.loc[:, "GarageQual"].fillna("No")
train.loc[:, "GarageCond"] = train.loc[:, "GarageCond"].fillna("No")
train.loc[:, "GarageArea"] = train.loc[:, "GarageArea"].fillna(0)
train.loc[:, "GarageCars"] = train.loc[:, "GarageCars"].fillna(0)
# HalfBath : NA na 0
train.loc[:, "HalfBath"] = train.loc[:, "HalfBath"].fillna(0)
# HeatingQC : NA na TA
train.loc[:, "HeatingQC"] = train.loc[:, "HeatingQC"].fillna("TA")
# KitchenAbvGr : NA na 0
train.loc[:, "KitchenAbvGr"] = train.loc[:, "KitchenAbvGr"].fillna(0)
# KitchenQual : NA na TA
train.loc[:, "KitchenQual"] = train.loc[:, "KitchenQual"].fillna("TA")
# LotFrontage : NA na 0
```

```
train.loc[:, "LotFrontage"] = train.loc[:, "LotFrontage"].fillna(0)
# LotShape : NA na Reg
train.loc[:, "LotShape"] = train.loc[:, "LotShape"].fillna("Reg")
# MasVnrType : NA na None or 0
train.loc[:, "MasVnrType"] = train.loc[:, "MasVnrType"].fillna("None")
train.loc[:, "MasVnrArea"] = train.loc[:, "MasVnrArea"].fillna(0)
# MiscFeature : NA na No or 0
train.loc[:, "MiscFeature"] = train.loc[:, "MiscFeature"].fillna("No")
train.loc[:, "MiscVal"] = train.loc[:, "MiscVal"].fillna(0)
# OpenPorchSF : NA na 0
train.loc[:, "OpenPorchSF"] = train.loc[:, "OpenPorchSF"].fillna(0)
# PavedDrive : NA na N
train.loc[:, "PavedDrive"] = train.loc[:, "PavedDrive"].fillna("N")
# PoolQC : NA na No or 0
train.loc[:, "PoolQC"] = train.loc[:, "PoolQC"].fillna("No")
train.loc[:, "PoolArea"] = train.loc[:, "PoolArea"].fillna(0)
# SaleCondition : NA na normal
train.loc[:, "SaleCondition"] = train.loc[:, "SaleCondition"].\
    fillna("Normal")
# ScreenPorch : NA na 0
train.loc[:, "ScreenPorch"] = train.loc[:, "ScreenPorch"].fillna(0)
# TotRmsAbvGrd : NA na 0
train.loc[:, "TotRmsAbvGrd"] = train.loc[:, "TotRmsAbvGrd"].fillna(0)
# Utilities : NA na AllPub
train.loc[:, "Utilities"] = train.loc[:, "Utilities"].fillna("AllPub")
# WoodDeckSF : NA na 0
train.loc[:, "WoodDeckSF"] = train.loc[:, "WoodDeckSF"].fillna(0)
# Electrical : NA na FuseA
train.loc[:, "Electrical"] = train.loc[:, "Electrical"].fillna("FuseA")

# Graph of the ratio of living space and cost
 plt.scatter(train.GrLivArea, train.SalePrice, c = "blue",
             marker = "s")
 plt.title("GrLivArea-SalePrice")
 plt.xlabel("GrLivArea")
 plt.ylabel("SalePrice")
 plt.show()
train = train[train.GrLivArea < 4000]

# Price dispersion graph
```

```
# plt.plot(train.SalePrice)
# plt.show()


ans = np.log(train.SalePrice.to_frame())
features = train.drop(["Id","SalePrice"], 1).copy()


# Translation of the category characteristics into numerical
features[features.keys()] = features[features.keys()].\
    apply(LabelEncoder().fit_transform)


scalar = StandardScaler()

features[features.keys()] = scalar.fit_transform(features[features.
                                                    keys()])

print(features)


kfold = KFold(n_splits=10, shuffle=True)
regs = []
random_state = 0
regs.append(Lasso())
regs.append(ElasticNetCV())
regs.append(SVR())
regs.append(KNeighborsRegressor())
regs.append(DecisionTreeRegressor(random_state=random_state))
regs.append(RandomForestRegressor(random_state=random_state))
regs.append(GradientBoostingRegressor(max_features='sqrt'))

results = []
for reg in tqdm(regs):
    results.append(np.sqrt(-cross_val_score
    (reg, features,
     y=ans.values.ravel(),
     scoring="neg_mean_squared_error",
     cv=kfold)))

```

```
156
157 means = []
158 errors = []
159 for result in results:
160     means.append(result.mean())
161     errors.append(result.std())
162 res_frame = pd.DataFrame({"CrossValMeans": means,
163                           "CrossValerrors": errors,
164                           "Algorithm": ["LinearRegression",
165                                         "ElasticNetCV",
166                                         'SVR',
167                                        'KNeighborsRegressor',
168                                        'DecisionTreeRegressor',
169                                        'RandomForestRegressor',
170                                        'GradientBoostingRegressor']})
171
172 print(res_frame)
173 g = sns.barplot("CrossValMeans", "Algorithm", data=res_frame,
174                 palette="Set3", orient="h",
175                 **{'xerr': errors} , label="medium")
176 g.set_xlabel("Mean Squared Error")
177 g = g.set_title("Cross validation scores")
178 plt.show()
```