ОЛИМПИАДА ШКОЛЬНИКОВ «ШАГ В БУДУЩЕЕЕ»

НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ
«ШАГ В БУДУЩЕЕ, МОСКВА»

115

регистрационный номер

Информатика и системы управления

название факультета

Кафедра ИУ7
Программное обеспечение ЭВМ и информационные технологии

название факультета

СРАВНЕНИЕ МЕТОДОВ МАШИННОГО ОБУЧЕНИЯ ДЛЯ ПРЕДСКАЗАНИЯ С ИСПОЛЬЗОВАНИЕМ НЕНАБЛЮДАЕМЫХ ПРИЗНАКОВ

наименование работы

Автор: Григорьев Роман Дмитриевич

фамилия, имя, отчество

ГБОУ г. Москвы «Бауманская инженерная школа №1580», класс 11 «Р»

наименование учебного заведения, класс

Москва – 2020

Сравнение методов машинного обучения для предсказания с использованием наблюдаемых признаков

Аннотация

Содержание

Введение

В данной проектной работе было проведено сравнение моделей машинного обучения для предсказания исхода матча в многопользовательской игре Defense Of The Ancients (DOTA 2). DOTA 2 - это киберспорт, где 2 команды по 5 человек играют друг против друга. Глядя на игру команд, а также на характеры каждого игрока в игре и историческую производительность, мы можем предсказать результаты матчей лучше, чем случайность. Результатом является то, что производительность нашей модели находится на одном уровне с производительностью на человеческом уровне этой задачи прогнозирования.

Популярность игры велика и быстро растет: самые крупные турниры могут похвастаться призами больше, чем PGA Tour и Wimbledon, а зрительская аудитория находится на одном уровне с MLB World Series и NBA Finals.

Краткий обзор DOTA 2 и ее игрового процесса необходим для контекста и понимания наших данных и наборов функций.

В матче по и DOTA 2 грают 2 команды по 5 человек каждый (аналогично баскетболу). Каждая команда может играть на одной из двух сторон («лучистой» или «страшной»), и их цель в игре - разрушить здание в базе противоположной команды «древней». Каждый матч проходит в течение двух различных фаз: стадии драфта и игровой фазы.

Во время игры каждый человек-игрок управляет 1 внутриигровым персонажем, или «героем». На момент создания данной проектной работы, в игре есть 119 героев каждый с уникальными способностями и характеристиками (герои продолжают добавляться в игру в регулярных обновлениях программы). Например, некоторые герои хороши в быстром разрушении зданий, в то время как другие герои хороши в разведке карты и получении ценной информации для остальной части своей команды. В результате некоторые герои объединяются друг с другом или противостоят общим стратегиям. Герой, которым управляет каждый игрок, определяется в ходе драфта.

В проекте каждый капитан команды будет по очереди запрещать и выбирать героев из 114 доступных. Запрет героя удаляет его из пула, поэтому ни одна из команд не может иметь человеческий контроль над ним, и выбор его выбирает героя для одного из своих человеческих игроков, чтобы контролировать. Поскольку каждый выбор или запрет сделан, капитаны команд будут реагировать и корректировать свой предстоящий выбор проекта, чтобы наилучшим образом удовлетворить свою стратегию, одновременно пытаясь противостоять стратегии, которую разрабатывают их противники. В общей сложности процесс драфта занимает около 10 минут, итогом которого сейчас являются обе команды есть свои 5 герой линии ИБП, с помощью которых можно выполнить свою стратегию уничтожения противостоящих древних.

Рисунок 1 - Пример команды Radiant

Рисунок 2 - Пример команды Dire

Таким образом, при прогнозировании результатов матчей крайне важно разрабатывать функции на основе каждой игры, которые представляют, какие команды играют, выбранный проект и насколько хорошие человеческие игроки исторически накапливали золото и опыт во время игрового процесса.

Рисунок 3 – Карта DOTA 2

Уже выполнялись работы по предсказаниям исхода игры и предыдущие попытки предсказать исход двух командных соревнований были средними. Если изучить спортивную аналитику, то легко увидеть параллель: строгий статистический анализ дал возможность управления бейсболом в 1930-1940-е годы, что затем привело к революции в статистических приложениях в других видах спорта и открыло новые области исследований в самой статистике. Был достигнут прогресс в предсказании результатов двух игр игроков. Игры с несколькими товарищами по команде в каждой команде оказываются более

трудными для прогнозирования. Сложность взаимодействия с товарищем по команде и индивидуальное исполнение игроков тех или иных действий резко усложняют эту проблему. Наиболее близким видом спорта к DOTA 2 является баскетбол. Предыдущие исследователи пытались предсказать баскетбольные матчи с помощью нейронных сетей. Я использую в проекте этим модели, когда экспериментирую с конфигурациями нашей нейронной сети. Команда CS 229 ранее пыталась предсказать исходы матчей DOTA 2, используя логистическую регрессию.

Человеческое представление на задаче

Важным диагностическим тестом для многих приложений машинного обучения является сравнение алгоритмической производительности с производительностью человека по интересующей проблеме. Эта диагностика особенно полезна для задач, над которыми люди работают хорошо, таких как распознавание лиц или речи. Для прогнозирования DOTA 2 я использую коэффициенты, найденные учеными Стендфордского университета на сайтах ставок 1win, в качестве базовой линии. Эти коэффициенты представляют собой коллективное мнение многих игроков о результатах матчей и будут столь же точным, как прогноз большинства индивидуальных игроков, при условии, что рынки ставок достаточно велики. А сравнение с букмекерскими сайтами крайне актуально, поскольку напрямую связано с потенциальным случаем использования наших моделей, который заключается в информировании игроков о принимаемых решениях.

Я взял данные которые собрали ученые Стендфордского университета, а именно данные ставок с сайта Gosugamers.net, который предоставляет услуги по ставкам на игры DOTA 2 и записывает данные ставок на 16800 и более матчей с мая 2013 по ноябрь 2017 года. Здесь матч относится к набору игр DOTA 2 между двумя командами (что эквивалентно матчу и набору в теннисе), и формат матча может отличаться. Некоторые примеры возможных форматов матчей-best-of-three (в котором первая команда, выигравшая две игры, выигрывает общую серию), best-of-one (в котором выигрывает серию та команда, которая выигрывает матч), best-of-two и так далее. Отметим две вещи: во-первых, у игроков есть только информация перед матчем (то есть у них нет информации о выбор героев противника, внутриигровые действия или результаты любой игры в рамках этого матча). Во-вторых, в то время как две команды не могут играть в ничью в одной игре, есть возможность иметь ничью в матче (например, в лучшем из двух матчей). Чтобы оценить производительность человека на Gosugamers.net, для каждого матча мы выбирали исход (либо победа команды 1, либо победа команды 2, либо ничья двух команд) с наибольшей суммой ставок в качестве коллективного прогноза игроков. Затем мы сравнили этот прогноз с фактическими результатами и обнаружили, что коллективный прогноз игроков

имеет точность 62,8%. Этот результат имеет смысл, так как он лучше, чем случайное предсказание, но не очень близок к идеальной точности, отражающей конкурентный характер игр DOTA 2 .

Наборы данных и объектов

Наши данные - это все профессиональные игры DOTA 2, сыгранные с ноября 2011 по октябрь 2017 года, или 47440 игр. Профессиональные матчи определяются как матчи, проводимые профессиональными командами в турнирах, которые официально санкционированы корпорацией Valve, программной компанией, ответственной за разработку DOTA 2. Мы разделили наши данные на три группы, которые будем называть train, dev и test. Поскольку наша модель будет наиболее полезна, если она может предсказать игры, которые еще предстоит сыграть в будущем, мы разделили наш тестовый набор на 4862 игры, сыгранные после 1 мая 2017 года. Из оставшихся 42578 игр сыгранных до 21 мая 2017 года, набор поездов – это случайно выбранные 90% этих игр, а набор dev-это другие 10%.

Набор функциональных возможностей

Решающее значение для успеха нашей модели имеет разработка функций, которые содержат структуру с предсказательной силой победы в игре. Мы построили 3 набора признаков, чтобы проверить каждую из наших гипотез о том, какие данные обеспечат наиболее прогностическую структуру.

Исходным набором данных для разработанной модели была публичная статистика матчей. Для устранения шума и предотвращения переобучения модели, был проведен отбор и выделение признаков из всего набора данных. Затем я нормализовал значения каждого из признаков, для того что бы заменить номинальные признаки так, чтобы каждый из них лежал в диапазоне от 0 до 1. Полученные после фильтрации признаков и нормализации значений данные были использованы для тренировки таких алгоритмов машинного обучения, как линейная и логистическая регрессия, а также алгоритма решающего дерева и его дополнений.

В итоге, приводится сравнение точности предсказания результатов матча с помощью реализованных алгоритмов.

Разбор исходного набора данных (схема связи между данными представлена на рисунке 4):

1) Hero_names.csv - в данном наборе данных у нас представлены сами герои (имена героев) и их id, с помощью которого мы можем узнавать в какой игре какие герои были выбраны. Так же с помощью этой информации нам удалось создать самый полезный признак hero_relative_strength
2) Match.csv - это наш самый главный набор данных. В нем лежит вся информация сыгранных матчах, а так же наш самый главный признак

(radiant_win), который показывает какая команда выиграла в том или ином матче.

3) Players.csv - в этом наборе данных у нас находится вся информация об игроках (в каком матче они участвовали, на каком герое, сколько золота заработали и.т.д). С помощью этого набора данных мы можем связывать игрока, матч и героя.

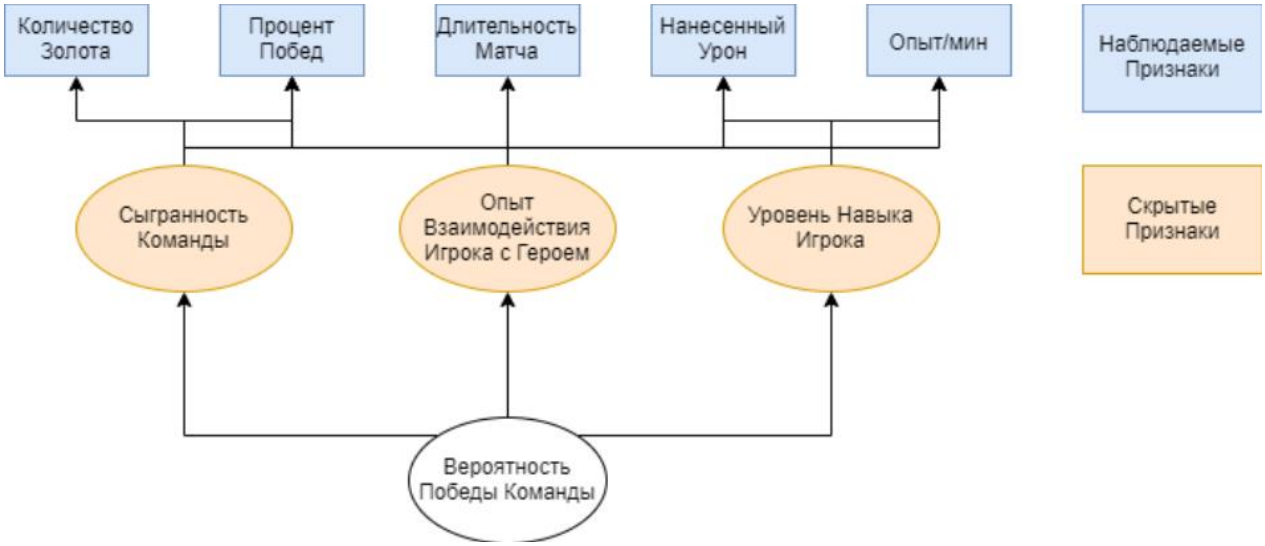


Рисунок 4 - Схема связи между нашими данными

Цели данной разработки:

- Получение действующего прототипа программы для сравнения моделей машинного обучения.
- Получение опыта разработки с использованием современных технологий и платформ

Выбор технологий

В соответствии с критериями выбора платформы, при создании проекта были использованы следующие технологии:

- Python – в качестве языка разработки
- Google colab – в качестве платформы разработки
- Microsoft Word – в качестве средства описания проекта
- Scikit-learn – в качестве библиотеки для обучения математических моделей
- Pandas – в качестве библиотеки для работы с наборами данных

Нормализация

Нормализация предполагает замену номинальных признаков так, чтобы каждый из них лежал в диапазоне от 0 до 1.

Для выполнения данного проекта можно было использовать один из методов……., таких как

- Нормализация: десятичное масштабирование - одинаково смещающее десятичную точку в числах, до попадания всех в единичную окрестность нуля

$$v_i' = \frac{v_i}{10^k}$$

- Мин-Макс Нормализация: Min-max нормализация - заполняющая единичный отрезок от края до края, в отличии от предыдущего метода

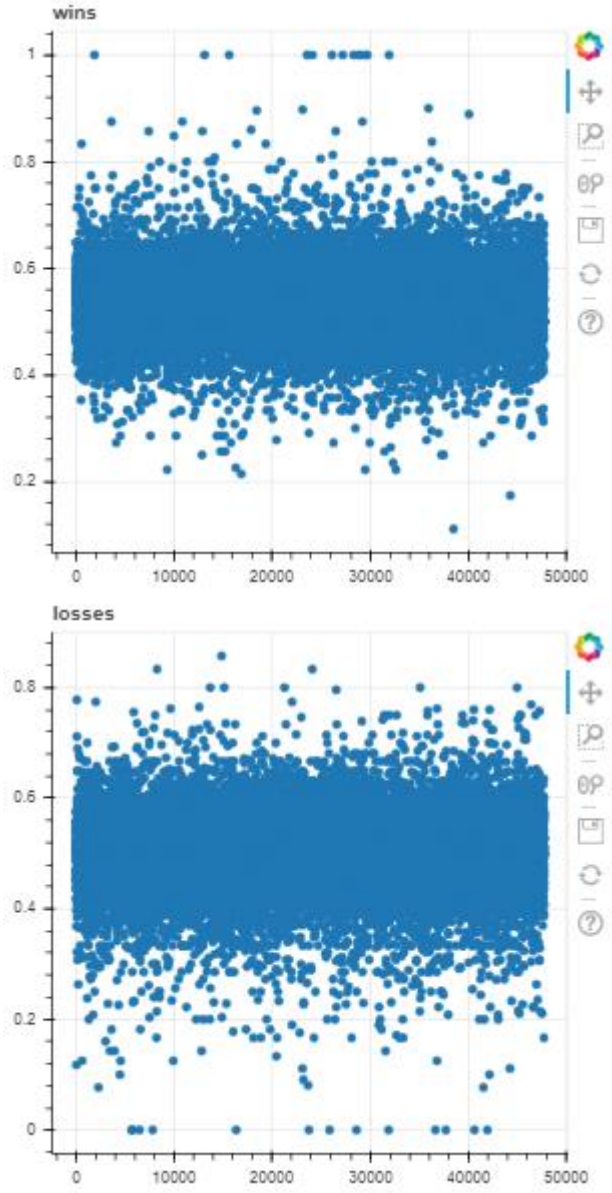
$$v_i' = \frac{v_i - min_i v_i}{max_i v_i - min_i v_i}$$

где минимальное и максимальное величины вычисляются либо автоматически по предоставленным данным, либо экспертно оцениваются исходя из области задачи.

- Нормализация Единичных Векторов: Нормализация Единичных Векторов – это масштабирование до единичной длины сжимает или растягивает вектор (строку данных можно рассматривать как D-мерный вектор) до единичной сферы.

$$\hat{X}[j,:] = \frac{X[j,:]}{\|X[j,:]\|}$$

В данном проекте мной был выбран метод Мин-Макс Нормализация, так как этот метод наиболее всего подходит для структуры используемых мной данных. А именно, выборка и так близка к нормальному распределению и Мин-Макс Нормализация не меняет распределение исходных данных, а только повышает численную стабильность.



Примеры нормализованных данных

Корреляции между признаками

Корреляция - это статистический термин, который в обычном употреблении относится к тому, насколько близки две переменные к линейной связи друг с другом. Например, две переменные, которые линейно зависят (x и y, которые зависят друг от друга как x = 2y), будут иметь более высокую корреляцию, чем две переменные, которые нелинейно зависят (например, u и v, которые зависят друг от друга как u = v2).

Признаки с высокой корреляцией являются более линейно зависимыми и, следовательно, оказывают почти такое же влияние на зависимую переменную. Когда два признака имеют высокую корреляцию, мы можем отбросить один из них.

Например:

1)gold_kill_creeps и last_hit имеют корреляцию 0.99

2)xp_per_min и gold_per_min имеют корреляцию 0.86

Коррелированные признаки сами по себе не влияют на точность классификации. Проблема в реалистичных ситуациях заключается в том, что у нас есть конечное число обучающих примеров, с помощью которых можно обучить классификатор. Для фиксированного числа обучающих примеров увеличение числа объектов обычно увеличивает точность классификации до точки, но поскольку число объектов продолжает увеличиваться, точность классификации в конечном итоге будет уменьшаться, потому что мы затем недооцениваем относительно большого числа объектов.

Если две числовые характеристики идеально коррелируют, то одна не добавляет никакой дополнительной информации (она определяется другой). То есть, если число объектов слишком велико (относительно размера обучающей выборки), то полезно уменьшить число объектов с помощью метода извлечения объектов.

Пример зависимости признаков друг от друга, используемый в нашем проекте (рисунок 5):

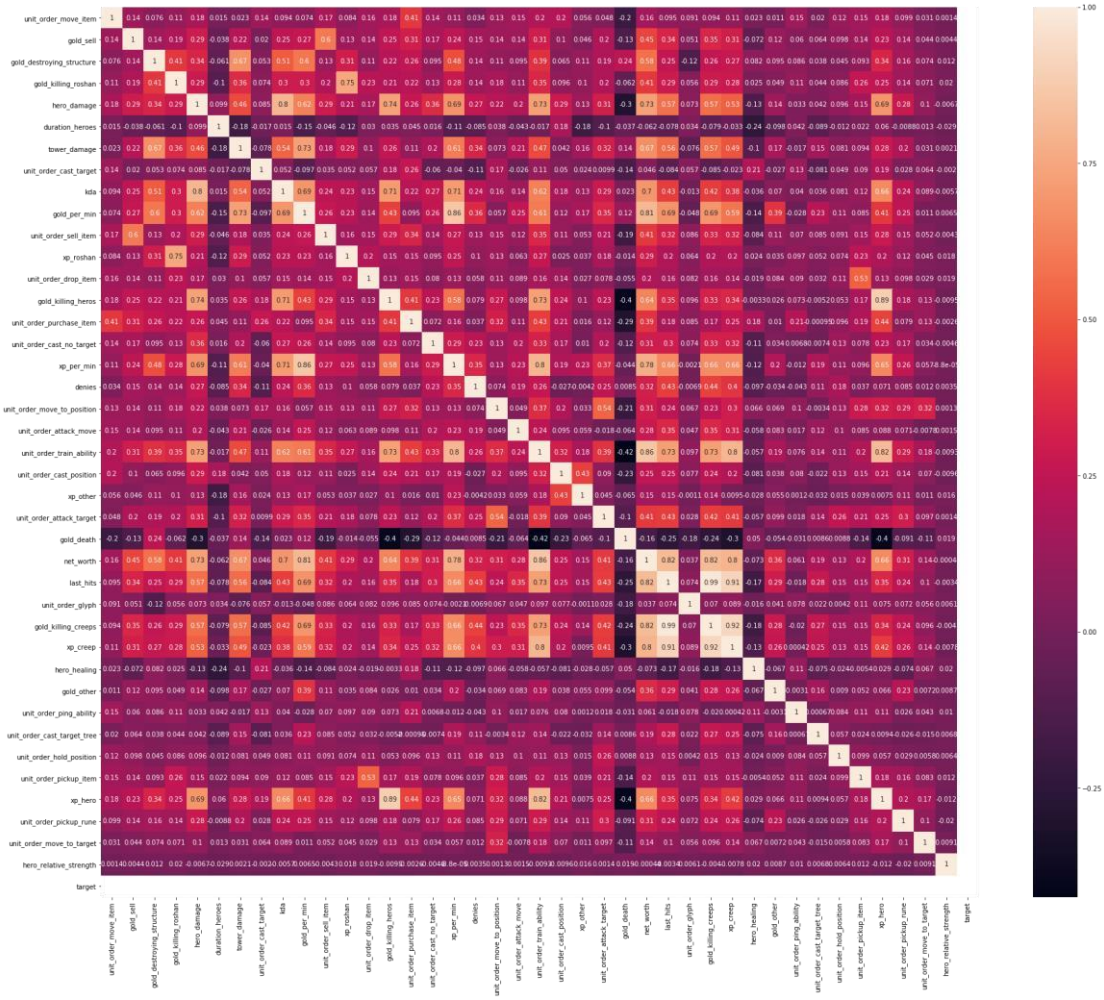


Рисунок 5 – Таблица корреляций между признаками, выделенными в проекте

На данном рисунке мы можем увидеть, как признаки зависят друг от друга. Некоторые признаки можно убрать или заменить без потери точности предсказания из-за высокой взаимной корреляции.

Описание алгоритмов

Линейная регрессия

Рассмотрим две непрерывные переменные

$$x = (x_1,\ x_2,\ldots\ldots x_n), \qquad y = (y_1,\ y_2,\ldots\ldots\ y_n)$$

Разместим точки на двумерном графике рассеяния и скажем, что мы имеем линейное соотношение, если данные аппроксимируются прямой линией.

Если мы полагаем, что у зависит от x, причём изменения в y вызываются именно изменениями в x, мы можем определить линию регрессии (регрессия y на x), которая лучше всего описывает прямолинейное соотношение между этими двумя переменными.

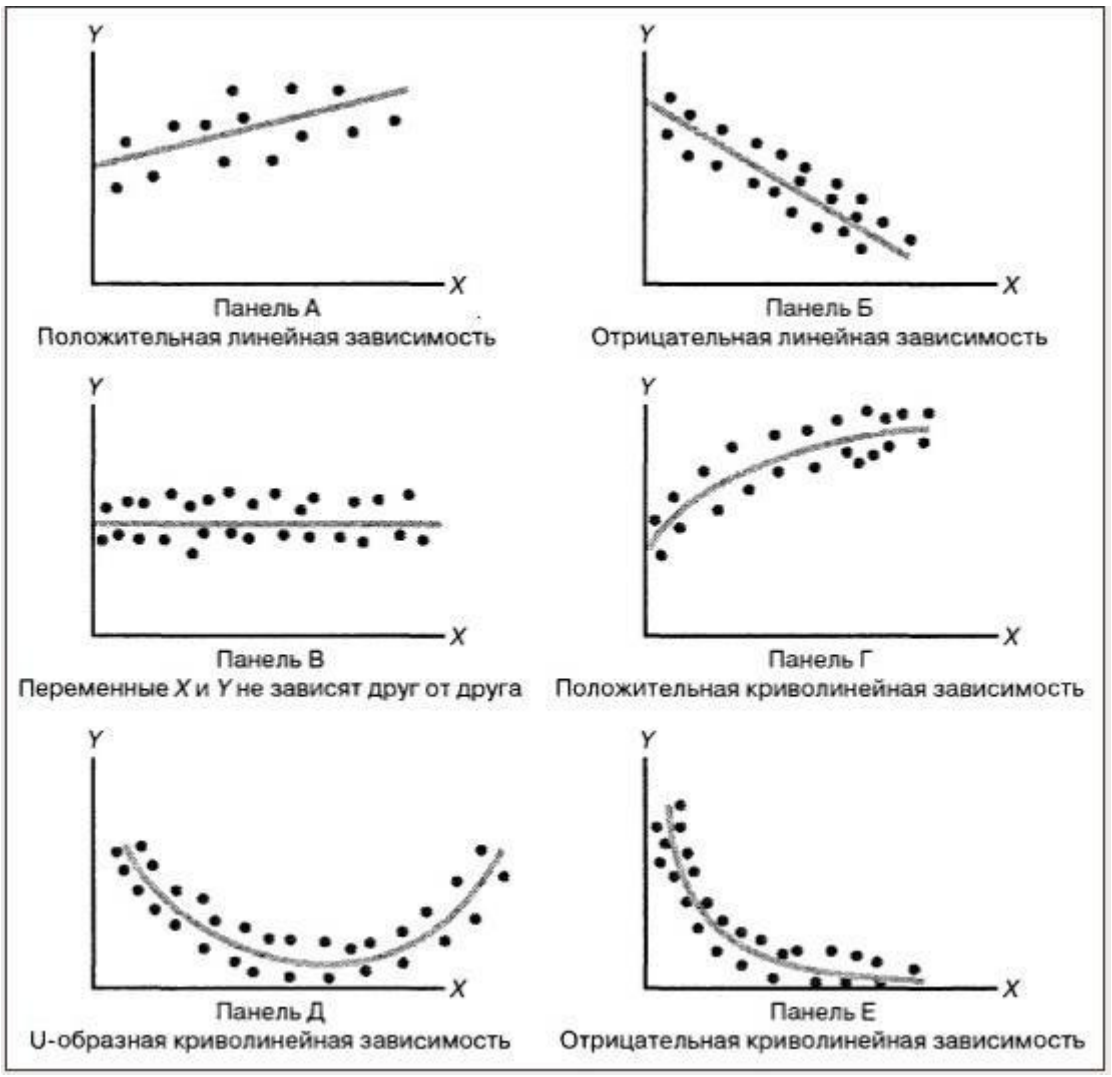


Рисунок 6 – Линейная регрессия

Обучение линейной регрессии

Метод наименьших квадратов — метод нахождения оптимальных параметров линейной регрессии, таких, что сумма квадратов ошибок (регрессионных остатков) минимальна. Метод заключается в минимизации евклидова расстояния $|Aw-y| \|$ $|A_w|$ между двумя векторами — вектором восстановленных значений зависимой переменной и вектором фактических значений зависимой переменной.

Задача метода наименьших квадратов состоит в выборе вектора w, минимизирующего ошибку $S = |A_{w-y}|^2$

Эта ошибка есть расстояние от вектора Y до вектора A_w.

Многомерная линейная регрессия.

Многомерная линейная регрессия — один из основополагающих методов машинного обучения.

В задаче многомерной линейной регрессии требуется восстановить неизвестную зависимость наблюдаемой вещественной величины (которая является значением неизвестной целевой функции) $Y*$ от набора вещественных же признаков $f_1,f_2,...,f_ь$.

Обычно для каждого отдельного наблюдения записываются значения признаков и целевой функции. Тогда из значений признаков для всех наблюдений можно составить матрицу признаков X, а из значений целевой функции — вектор её значений Y:

$$Xij=fj(xi),1\leq i\leq n,1\leq j\leq m$$

$$yi=y*(xi),1\leq i\leq n,1\leq j\leq m$$

Здесь x_i — некоторое конкретное наблюдение. Таким образом, строчки нашей матрицы соответствуют наблюдениям, а столбцы — признакам, а общее количество наблюдений и, следовательно, строк в матрице — n.

Найти решение задачи означает построить некоторую решающую функцию a: $R_m\rightarrow R$. Сейчас мы говорим о задаче линейной регрессии, поэтому будем считать, что решающая функция является линейной. На самом деле, это значит, что решающая функция представляет из себя просто скалярное произведение вектора признаков на некоторый вектор весов:

$$aw(xi)=\sum j=1mwj\cdot fj(xi)$$

Стало быть, решением нашей задачи является вектор весов признаков $w=\langle w_1,w_2,...,w_m\rangle$.

Изобразить многомерную линейную зависимость графически достаточно сложно (для начала, представим себе 100-мерное пространство признаков), однако иллюстрации для одномерного случая вполне достаточно, чтобы понять происходящее.

Чтобы закончить формулировку задачи машинного обучения, требуется лишь указать функционал потерь, который отвечает на вопрос, насколько хорошо конкретная решающая функция предсказывает значения целевой функции. Сейчас мы будем использовать традиционный для регрессионного анализа среднеквадратический функционал потерь:

$$Q(a,X,y)=1n\sum i=1n(a(xi)-yi)2$$

Итак, задачей многомерной линейной регрессии является нахождение набора весов w такого, что значение функционала потерь Q(aw,A,y) достигает на нём своего минимума. Для анализа обычно удобно отказаться от радикала и усреднения в функционале потерь:

$$Q1(a,X,y)=\sum i=1n(a(xi)-yi)2$$

Для дальнейшего изложения будет удобно отождесвить наблюдения с наборами соответствующих им факторов:

$$xij=fj(i)$$

Нетрудно видеть, что функционал потерь теперь записывается просто, как скалярный квадрат: вектор предсказаний можно записать как Xw, а вектор отклонений от правильных ответов — как Xw−y. Тогда:

$$Q1(a,X,y)=\langle Xw-y,Xw-y\rangle$$

Реализация метода в проекте

Для того, чтобы использовать метод линейной регрессии для задачи классификации, необходимо установить порог, разделяющий классы. В моем случае, был выбран порог в 0.5 – то есть, если регрессия предсказывает значение 0.6, то оно округляется до 1, а если 0.4, то до 0. Таким образом происходит преобразование непрерывных ответов в категориальные.

Логистическая регрессия

В отличие от линейной регрессии, в методе логистической регрессии не производится предсказание значения числовой переменной исходя из выборки исходных значений. Вместо этого, значением функции является вероятность того, что данное исходное значение принадлежит к определенному классу.

Основная идея логистической регрессии заключается в том, что пространство исходных значений может быть разделено линейной границей (т.е. прямой) на две соответствующих классам области. Что же имеется ввиду под линейной границей? В случае двух измерений — это просто прямая линия без изгибов, в случае трех — плоскость, и так далее. Граница задается в зависимости от имеющихся исходных данных и обучающего алгоритма. Чтобы все работало, точки исходных данных должны разделяться линейной границей на две вышеупомянутые области. Если точки исходных данных удовлетворяют этому требованию, то их можно назвать линейно разделяемыми.

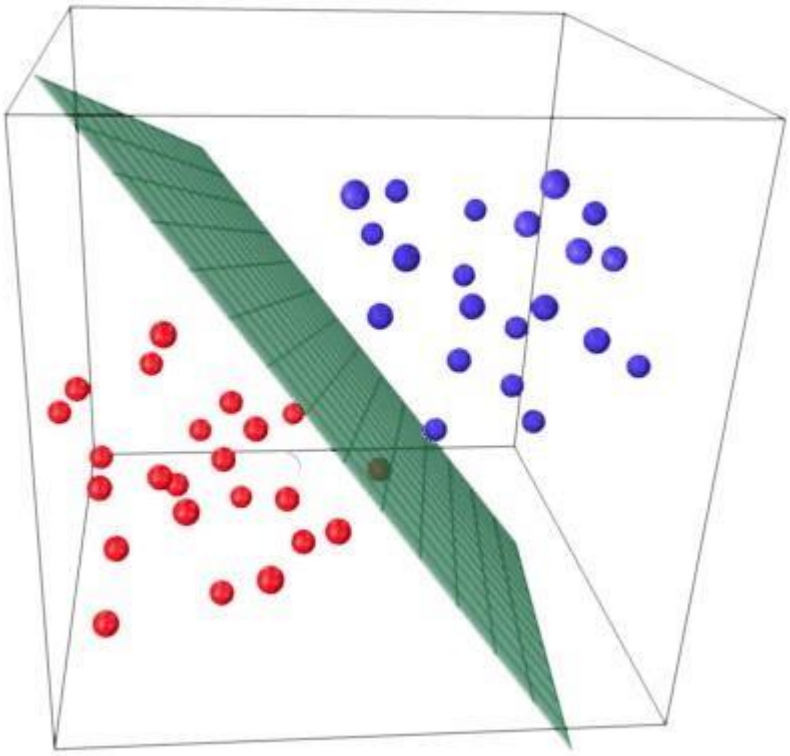


Рисунок 7 – Логистическая регрессия

Для начала рассмотрим, как происходит разделение, то есть геометрический подтекст «разделения» исходного пространства на две области. Для простоты (в отличие от показанного выше 3-мерного графика) возьмем две исходные переменные – x_1 и x_2, тогда функция, соответствующая границе, примет вид: $\beta_0+\beta_1x_1+\beta_2x_2$.

Рассмотрим точку. Подставляя значения x_1 и x_2 в граничную функцию, получим результат $\beta_0+\beta_1a+\beta_2b$.

Далее, в зависимости от положения (a,b) следует рассмотреть три варианта:
- (a,b) лежит в области, ограниченной точками класса "+". Тогда $\beta_0+\beta_1a+\beta_2b$, будет положительной, находясь где-то в пределах $(0, \infty)$. С математической точки зрения, чем больше величина этого значения, тем больше расстояние между точкой и границей. А это означает большую вероятность того, что (a,b) принадлежит классу "+". Следовательно, будет находиться в пределах (0.5, 1].
- (a,b) будет отрицательной, находясь в пределах $(-\infty, 0)$. Но, как и в случае с положительным значением, чем больше величина выходного значения по модулю, тем больше вероятность, что (a,b) принадлежит классу "-", и находится в интервале [0, 0.5)

$\beta_0+\beta_1a+\beta_2b=0$

(a,b)

- (a,b) лежит на самой границе. В этом случае, это означает, что модель действительно не может определить, принадлежит ли (a,b) к классу "+" или к классу "-". И в результате, P_+ будет равняться 0,5

Итак, мы имеем функцию, с помощью которой возможно получить значение в пределах (-∞, +∞) имея точку исходных данных. Теперь необходимо преобразовать полученное значение в вероятность P_+, пределы которой $[0, 1]$. Это делаем с помощью функции отношения шансов (OR). Обозначим P(X) вероятностью происходящего события X. Тогда, отношение шансов OR(X) определяется из $\frac{P(X)}{1-P(X)}$, а это — отношение вероятностей того, что произойдет ли событие или не произойдет. Вероятность и отношение шансов содержат одинаковую информацию. Но, в то время как P(X) находится в пределах от 0 до 1, OR (X) находится в пределах от 0 до ∞.

Это значит, что необходимо еще одно действие, так как используемая OR(X) нами граничная функция выдает значения от +∞ до -∞. Далее следует вычислить логарифм, что называется логарифмом отношения шансов.
Алгоритм логистической регрессии будет выглядеть следующим образом:

Шаг 1. Вычислить значение $\beta_0+\beta_1a+\beta_2b$ граничной функции (или, как вариант, функцию отношения шансов). Для простоты обозначим эту величину t.
Шаг 2. Вычислить отношение шансов
Шаг 3. Имея значение OR_+, вычислить P_+ с помощью простой зависимости:

$$P_+ = \frac{OR_+}{1+OR_+}$$

Получив значение t в шаге 1, можно объединить шаги 2 и 3:

$$P_+ = \frac{e^t}{1+e^t}$$

Правая часть уравнения, указанного выше, называется логистической функцией. Отсюда и название, данной этой модели обучения.

Для обучения логистической регрессии мною был использован метод максимального правдоподобия так как:

1) асимптотическая эффективность оценок максимального правдоподобия обеспечивает их преимущества в задачах накопления информации, при работе с большими массивами
2) связь оценок максимального правдоподобия с достаточными статистиками делает этот метод более приоритетным, чем остальные.

Дерево решений

Дерево решений - это тип управляемого алгоритма обучения, имеющего заранее определенную целевую переменную, который в основном используется в задачах классификации. Он работает как для категориальных, так и для непрерывных входных и выходных переменных. В этом методе мы разделяем популяцию или выборку на два или более однородных набора на основе наиболее значимого разделителя или дифференциатора входных переменных.

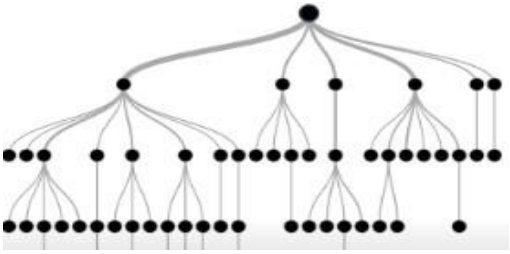


Рисунок 8 – Дерево решений

Типы принятия решения в деревьях

1. Категориальная переменная дерево решений: дерево решений, которое имеет категориальную целевую переменную, затем его называют категориальным переменным деревом решений.
2. Дерево решений с непрерывной переменной: дерево решений имеет непрерывную целевую переменную, затем оно называется деревом решений с непрерывной переменной.

Важная терминология, относящаяся к деревьям

- Корневой узел: он представляет всю выборку, и это в дальнейшем делится на два или более однородных набора.
- Разделение: это процесс разделения узла на два или более подузлов.
- Узел принятия решений: когда подузел разделяется на дальнейшие подузлы, то он называется узлом принятия решений.
- Конечный или терминальный узел: узлы не разделяются называется конечным или конечным узлом.
- Обрезка: когда мы удаляем подузлы узла принятия решений, этот процесс называется обрезкой. Можно сказать, обратный процесс расщепления.

- Ветвь или поддерево: подсекция всего дерева называется ветвью или поддеревом.

Преимущества и недостатки дерева решений

- Вывод дерева решений очень легко понять. Для их чтения и интерпретации не требуется никаких статистических знаний. Его графическое представление очень интуитивно, и можно легко связать свою гипотезу.
- Полезно в исследовании данных: дерево решений является одним из самых быстрых способов определения наиболее значимых переменных и связей между двумя или более переменными. С помощью деревьев решений мы можем создавать новые переменные или функции, которые обладают лучшей способностью предсказывать целевую переменную. Дерево решений также может быть использовано на этапе исследования данных. Например, мы работаем над проблемой, где у нас есть информация, доступная в сотнях переменных, там дерево решений поможет определить наиболее значимую переменную.
- Требуется меньше очистки данных по сравнению с некоторыми другими методами моделирования. Дерево решений не зависит от выбросов и пропущенных значений в достаточной степени.
- Тип данных не является ограничением: дерево решений может обрабатывать как числовые, так и категориальные переменные.
- Непараметрический метод: дерево решений считается непараметрическим методом. Это означает, что деревья принятия решений не имеют никаких предположений о распределении пространства и структуре классификатора.
- Чрезмерная подгонка: чрезмерная подгонка является одной из самых практических трудностей для моделей дерева решений. Эта проблема решается путем установки ограничений на параметры модели и обрезки (подробно рассматривается ниже).
- Не подходит для непрерывных переменных: при работе с непрерывными числовыми переменными дерево решений теряет информацию, когда оно классифицирует переменные в разных категориях.

Деревья регрессии и деревья классификации

- Конечные узлы (или листья) лежат в нижней части дерева решений. Это означает, что деревья принятия решений обычно рисуются вверх ногами так, что листья - это дно, а корни - вершины.
- Оба дерева работают почти аналогично друг другу.

Рассмотрим основные различия и сходство между классификационными и регрессионными деревьями:

- Деревья регрессии используются, когда зависимая переменная непрерывна. Деревья классификации используются, когда зависимая переменная категориальна.
- В случае дерева регрессии значение, полученное терминальными узлами в обучающих данных, представляет собой среднюю реакцию наблюдения, падающую в этой области. Таким образом, если невидимое наблюдение данных попадает в эту область, мы сделаем его прогноз со средним значением.
- В случае дерева классификации значение (класс), полученное терминальным узлом в обучающих данных, является режимом наблюдений, попадающих в эту область. Таким образом, если невидимое наблюдение данных попадает в эту область, мы сделаем его прогноз со значением режима.
- Оба дерева разделяют пространство независимых переменных на отдельные и неперекрывающиеся области.
- Оба дерева следуют каждому подходу сверху вниз, известному как рекурсивное двоичное расщепление. Называется оно «сверху вниз», потому что начинается с вершины дерева, когда все наблюдения доступны в одной области и последовательно разбивает пространство предиктора на две новые ветви вниз по дереву. Он известен как «жадный», потому что алгоритм заботится (ищет лучшую доступную переменную) только о текущем расщеплении, а не о будущих расщеплениях, которые приведут к лучшему дереву.
- Этот процесс разделения продолжается до тех пор, пока не будут достигнуты определенные пользователем критерии остановки. Например: мы можем сказать алгоритму остановиться, как только число наблюдений в узле станет меньше 50.
- В обоих случаях процесс расщепления приводит к появлению полностью выращенных деревьев до достижения критерия остановки. Но, полностью выросшее дерево, вероятно, будет перекрывать данные, что приводит к низкой точности на невидимых данных. Это приносит «обрезку».

Индекс Джини

$$G = 1 - 2 \sum_{i-1}^{n} x_i \, cumy_i + \sum_{i=1}^{n} x_i \, y_i$$

Индекс Джини говорит, что если мы выбираем два элемента из популяции случайным образом, то они должны быть одного класса и вероятность этого равна 1, если популяция чиста.

1. Данный индекс работает с категориальной целевой переменной «успех» или «неудача».
2. Индекс Джини выполняет только двоичные разбиения.
3. Чем выше значение индекса Джини, тем выше однородность.
4. CART (дерево классификации и регрессии) использует метод Джини для создания двоичных разбиений.

Шаги для расчета Джини для разделения

1. Вычислить индекс Джини для подузлов, используя формулу суммы квадратов вероятности успеха и неудачи
2. Вычислить индекс Джини для разделения, используя взвешенную оценку индекса каждого узла этого разделения

Хи-Квадрат

Это алгоритм для определения статистической значимости различий между подузлами и родительским узлом. Мы измеряем его суммой квадратов стандартизованных разностей между наблюдаемыми и ожидаемыми частотами целевой переменной.

$$X^2 = \sum_{i=1}^{k} \frac{(n_i - n * p_i)^2}{n * p_i}$$

1. Алгоритм работает с категориальной целевой переменной "успех" или "неудача".
2. Алгоритм может выполнять два или более расколов.
3. Чем выше значение Хи-Квадрат, тем выше статистическая значимость различий между подузлом и родительским узлом.
4. Вычислить Хи-Квадрат для каждого узла
5. Он генерирует дерево, называемое CHAID (Chi-square Automatic Interaction Detector)

Шаги для вычисления Хи-квадрат для разделения:

1. Вычислить Хи-квадрат для отдельного узла путем вычисления отклонения для успешного и неудачного
2. Вычислить Хи-квадрат разбиения с использованием суммы всех Хи-квадратов успеха и неудачи каждого узла разбиения

Являются ли модели на основе дерева лучше, чем линейные модели?

Некоторые ключевые факторы, которые помогут решить, какой алгоритм использовать:

- Если связь между зависимой и независимой переменной хорошо аппроксимируется линейной моделью, линейная регрессия превзойдет модель на основе дерева.
- Если существует высокая нелинейность и сложная взаимосвязь между зависимыми и независимыми переменными, модель дерева будет превосходить классический метод регрессии.
- Если нужно построить модель, которая легко объяснима людям, модель дерева решений всегда будет лучше, чем линейная модель. Модели дерева решений проще интерпретировать, чем линейную регрессию.

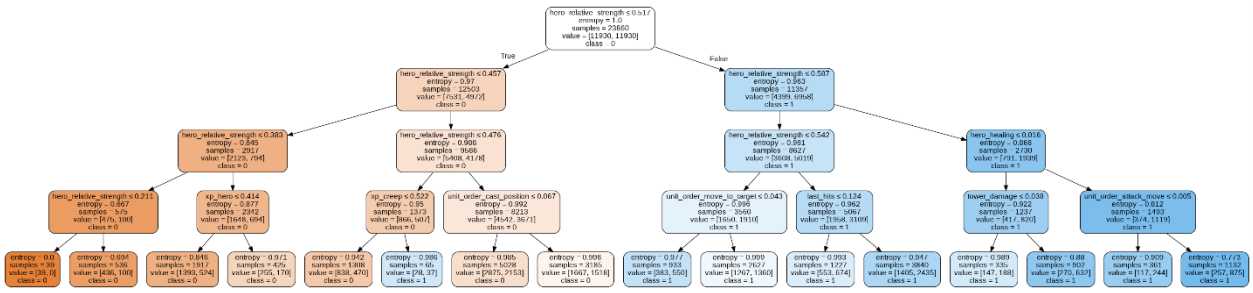


Рисунок 9 – Дерево решений в нашем проекте

Самым верхним является самое важное разделение, чем ниже мы спускаемся тем важность наших условий меньше. Мы основываемся на таблице полезности признаков. Из ние мы узнаем какие признаки наиболее важные. Благодаря этой инфомации мы задаем условия и строим дерево

Рандомный лес

- Этот алгоритм может решить оба типа задач т. е. классификации и регрессии и делает достойную оценку на обоих фронтах.
- Одно из преимуществ случайного леса - это возможность обрабатывать большой набор данных с более высокой размерностью. Он может обрабатывать тысячи входных переменных и определять наиболее значимые переменные, поэтому он рассматривается как один из методов уменьшения размерности. Кроме того, модель выводит значение переменной, которая может быть очень удобной функцией (на некотором случайном наборе данных).
- Алгоритм имеет эффективный метод для оценки недостающих данных и поддерживает точность, когда большая часть данных отсутствуют.

- Алгоритм имеет методы для балансировки ошибок в наборах данных, где классы несбалансированы. Описанные выше возможности могут быть расширены до немаркированных данных, что приводит к бесконтрольной кластеризации, представлению данных и обнаружению выбросов.
- Случайный лес включает выборку входных данных с заменой, называемой загрузочной выборкой. Здесь одна треть данных не используется для обучения и может быть использована для тестирования.

Недостатки случайного леса

- Алгоритм хорошо работает при классификации, но не так хорошо, как для регрессионной задачи, поскольку он не дает точных непрерывных прогнозов природы. В случае регрессии он не предсказывает выход за пределы диапазона в обучающих данных, и что они могут перекрывать наборы данных, которые являются особенно шумными.
- Случайный лес может ощущаться как подход черного ящика – очень мало контроля над тем, что делает модель, в лучшем случае можно попробовать различные параметры и случайную инициализацию

Что такое пакетирование?

Bagging - это метод, используемый для уменьшения дисперсии наших прогнозов путем объединения результатов нескольких классификаторов, смоделированных на разных подвыборках одного и того же набора данных.

1. Создание Нескольких Наборов Данных:
    - Выборка производится с заменой на исходные данные и формируются новые наборы данных.
    - Новые наборы данных могут содержать часть столбцов, а также строки, которые обычно являются гиперпараметрами в модели упаковки
    - Взятие фракций строк и столбцов менее 1 помогает в создании надежных моделей, менее склонных к подгонке
2. Построить Несколько Классификаторов:
    - Классификаторы построены на каждом наборе данных.
    - Как правило, один и тот же классификатор моделируется на каждом наборе данных и выполняются прогнозы.
3. Комбинации Классификаторов:
    - Предсказания всех классификаторов объединяются с использованием среднего, медианы или значения режима в зависимости от рассматриваемой проблемы.
    - Объединенные значения обычно более надежны, чем одна модель.

Количество построенных моделей не является гиперпараметром. Более высокое число моделей всегда лучше или может дать аналогичную производительность, чем более низкие числа. Теоретически можно показать, что дисперсия комбинированных прогнозов сводится к $\frac{1}{n}$ (n: число классификаторов) исходной дисперсии, при некоторых допущениях.

Что такое форсирование? Как это работает?

Определение: термин "бустинг" относится к семейству алгоритмов, которые преобразуют слабого ученика в сильных учеников. Чтобы преобразовать слабого ученика в сильного ученика, мы объединим предсказание каждого слабого ученика, используя такие методы, как:

- Использование среднего или взвешенного среднего
- Учитывая прогноз имеет более высокий голос

Чтобы найти слабое правило, мы применяем алгоритмы базового обучения (ML) с другим распределением. Каждый раз, когда применяется алгоритм базового обучения, он генерирует новое правило слабого предсказания. Это итерационный процесс. После многих итераций алгоритм форсирования объединяет эти слабые правила в одно сильное правило прогнозирования.

Существует много алгоритмов повышения, которые придают дополнительный импульс точности модели. В проекте были применены два алгоритма, таких как градиентный бустинг (GBM) и XGboost.

Почему мы расположили методы именно в таком порядке

1) Сначала мы используем очень примитивные методы такие как линейная регрессия и логистическая. Это сделано для того что бы наглядно показать, как изменяются результаты с повышением сложности метода. Ведь у нас в проекте наглядно видно, что с повышением сложности результат улучшается.
2) Это тема для меня новая и начинать сразу со сложных методов не прагматично. Ведь для того что бы выжать максимум из каждого метода нужно хорошо в нем разбираться, именно поэтому начал я с линейной регрессии, а закончил случайным лесом.

Метрики

В задачах машинного обучения для оценки качества моделей и сравнения различных алгоритмов используются метрики.

Accuracy, precision и recall

Перед переходом к самим метрикам необходимо ввести важную концепцию для описания этих метрик в терминах ошибок классификации — confusion matrix (матрица ошибок). Допустим, что у нас есть два класса и

алгоритм, предсказывающий принадлежность каждого объекта одному из классов, тогда матрица ошибок классификации будет выглядеть следующим образом:

	y=1	y=0
$\hat{y}$=1	True Positive (TP)	False Positive (FP)
$\hat{y}$=0	False Negative (FN)	True Negative (TN)

Таким образом, ошибки классификации бывают двух видов: False Negative (FN) и False Positive (FP).

Accuracy

Accuracy — доля правильных ответов алгоритма:

$$\text{accuracy} = \frac{TR+TN}{TP+TN+FP+FN}$$

Precision, recall и F-мера

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

$$\text{precision} = \frac{TR}{TR+FR}$$

$$\text{recall} = \frac{TR}{TR+FN}$$

Существует несколько различных способов объединить precision и recall в агрегированный критерий качества. F-score — среднее гармоническое precision и recall:

$$F_\beta = (1 + \beta^2) * \frac{precision*recall}{(\beta^2*precision)+ recall}$$

AUC-ROC и AUC-PR

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) — площадь под ROC-кривой один из самых популярных функционалов качества в задачах бинарной классификации. Как правило, объяснение начинают с введения разных терминов (FPR, TPR).

$$\text{TPR} = \frac{TP}{TP+FN}$$

$$\text{FPR} = \frac{FP}{FR+TN}$$

TPR Доля верных положительных классификаций (True Positive Rate, TPR):, а FPR показывает долю ложных положительных классификаций (False Positive Rate, FPR)

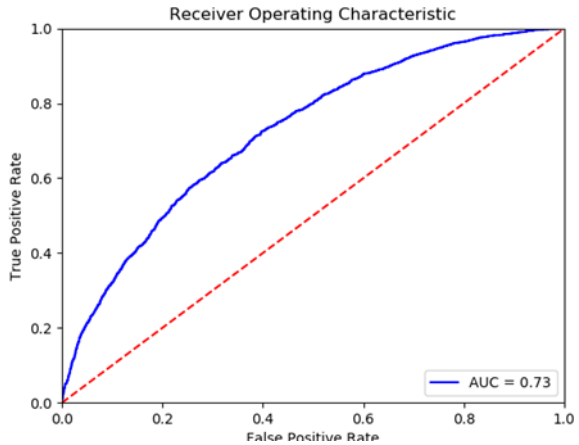


Рисунок 12 – ROC-AUC случайного леса

Precision и recall также используют для построения кривой и, аналогично AUC-ROC, находят площадь под ней. Площадь под ROC-кривой AUC (Area Under Curve) является агрегированной характеристикой качества классификации, не зависящей от соотношения цен ошибок. Чем больше значение AUC, тем «лучше» модель классификации. Данный показатель часто используется для сравнительного анализа нескольких моделей классификации.

Результаты работы

```
'gold_per_min', 'xp_per_min', 'denies', 'last_hits', 'hero_damage',
'hero_healing', 'tower_damage', 'kda', 'net_worth', 'duration_heroes',
```

Мною были выбраны признаки, которые исходя из опыта чаще всего влияют на результат матчка.. На мой взгляд это самый лучший набор для определения точности разных методов.

1)Linear Regression

	precision	recall	F1-score
0.0	0.51	0.99	0.68
1.0	0.88	0.10	0.18
Accuracy			0.55

ТОЧНОСТЬ: 53%

Матрица ошибок

	Positive	negative
positive	True positive - 1577	False positive - 23
negative	False negative - 1398	True negative - 171

Самый простой метод обладает самым худшим результатом, 53% правильных предсказаний.

2)Logistic Regression

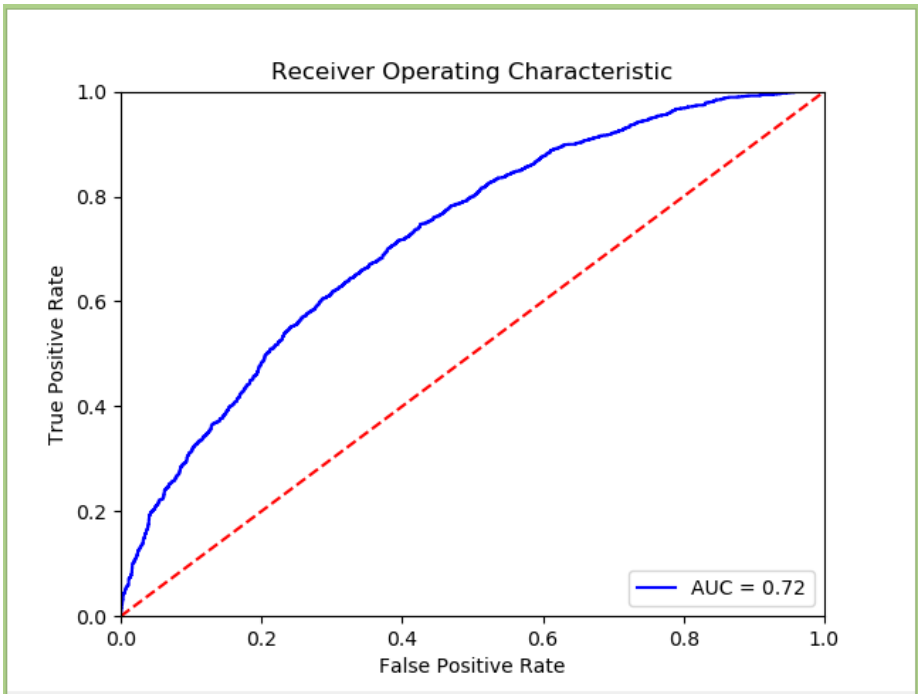
	precision	recall	F1-score
0.0	0.58	0.57	0.58
1.0	0.58	0.59	0.58
Accuracy			0.58

ТОЧНОСТЬ: 58%

Матрица ошибок

	Positive	negative
positive	True positive - 1278	False positive - 872
negative	False negative - 852	True negative - 1223

По результатам мы видим что логистическая регрессия лучше чем линейная, 58% правильных предсказаний, но все же это не самый лучший результат, с выбранными мною признаками

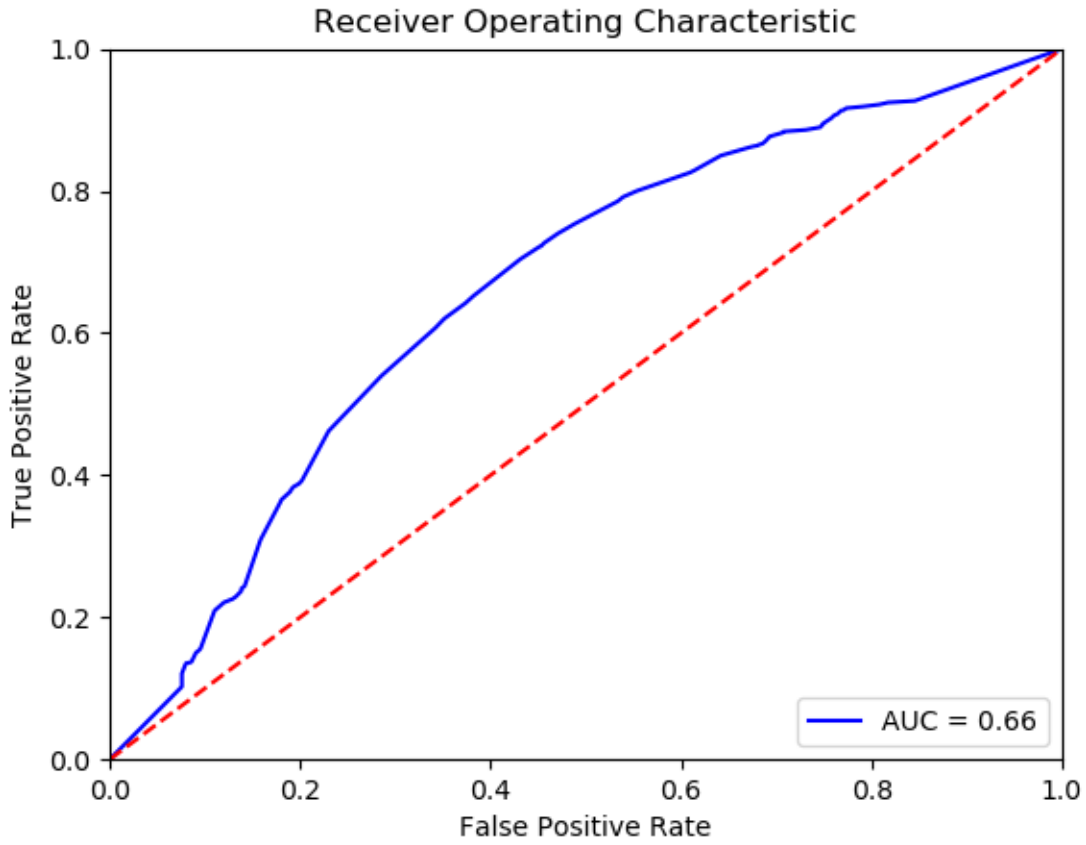


2)Decision tree

	precision	recall	F1-score
0.0	0.64	0.62	0.63
1.0	0.63	0.65	0.64
Accuracy			0.64

ТОЧНОСТЬ: 64%

	positive	negative
positive	True positive - 1398	False positive - 712
negative	False negative - 801	True negative - 1314



Дерево решений обладает результатом получше чем у линейной регрессии, но все же хуже чем большинство остальных методов

3)Random forest

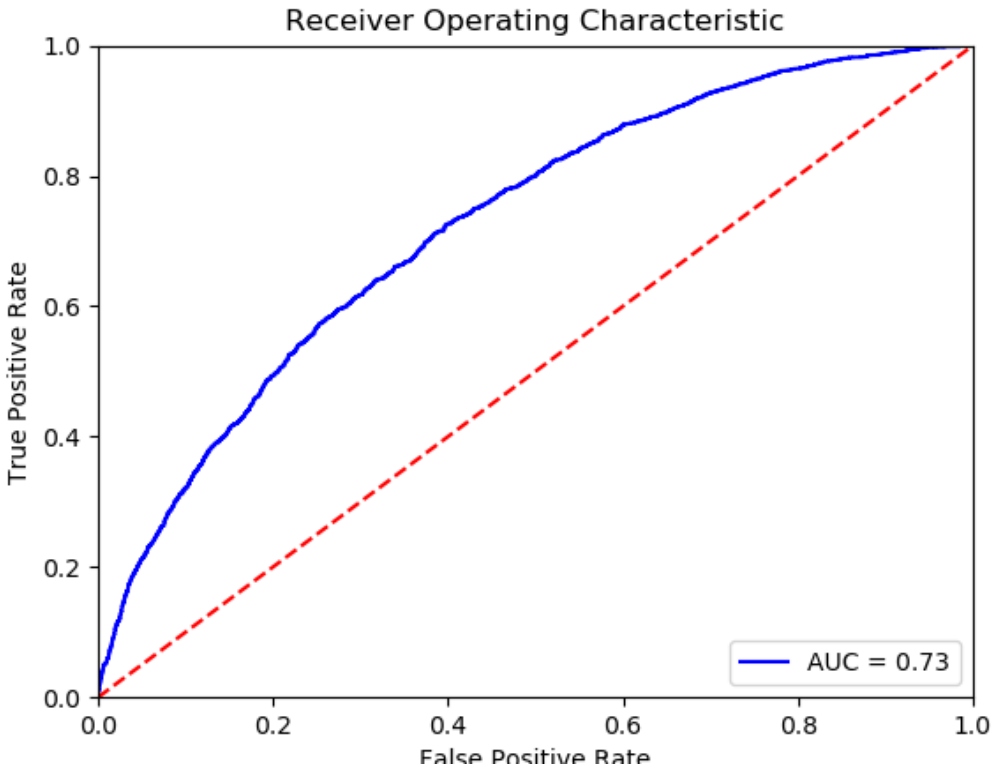
	precision	recall	F1-score
0.0	0.69	0.62	0.65
1.0	0.65	0.72	0.68
Accuracy			0.67

ТОЧНОСТЬ: 67%

Матрица ошибок

	Positive	negative
positive	True positive - 2103	False positive - 1055
negative	False negative - 1099	True negative - 2081

Рандомный лес обладает лучшим показателем точности, с выбранным мной списком признаков.



4)XGBOOST FOREST

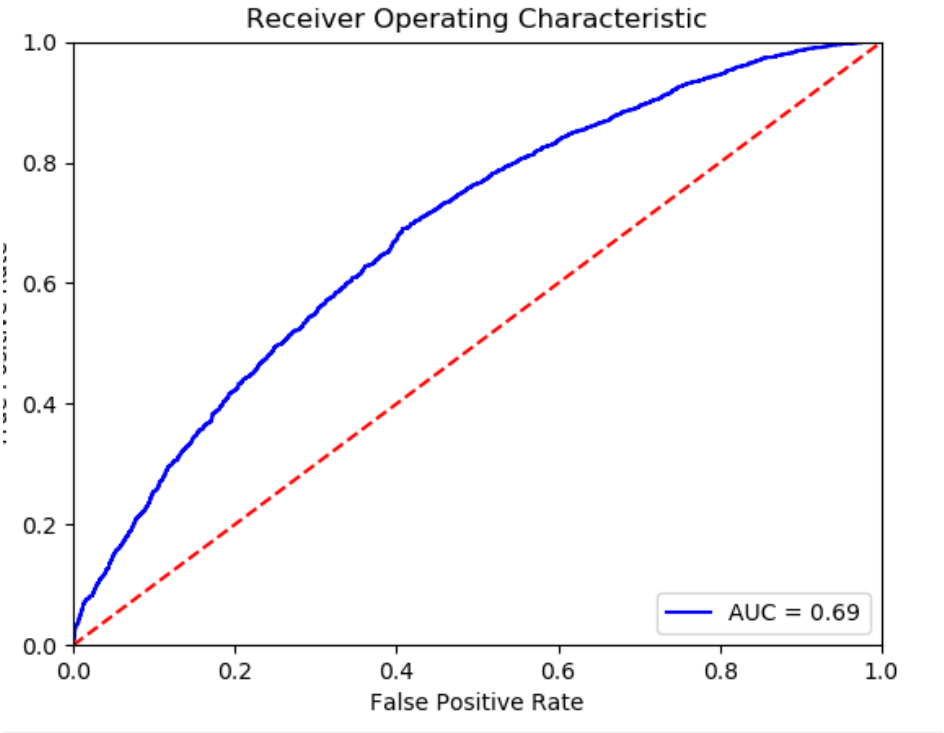
	precision	recall	F1-score
0.0	0.63	0.61	0.62
1.0	0.62	0.64	0.63
Accuracy			0.63

ТОЧНОСТЬ: 63%

Матрица ошибок

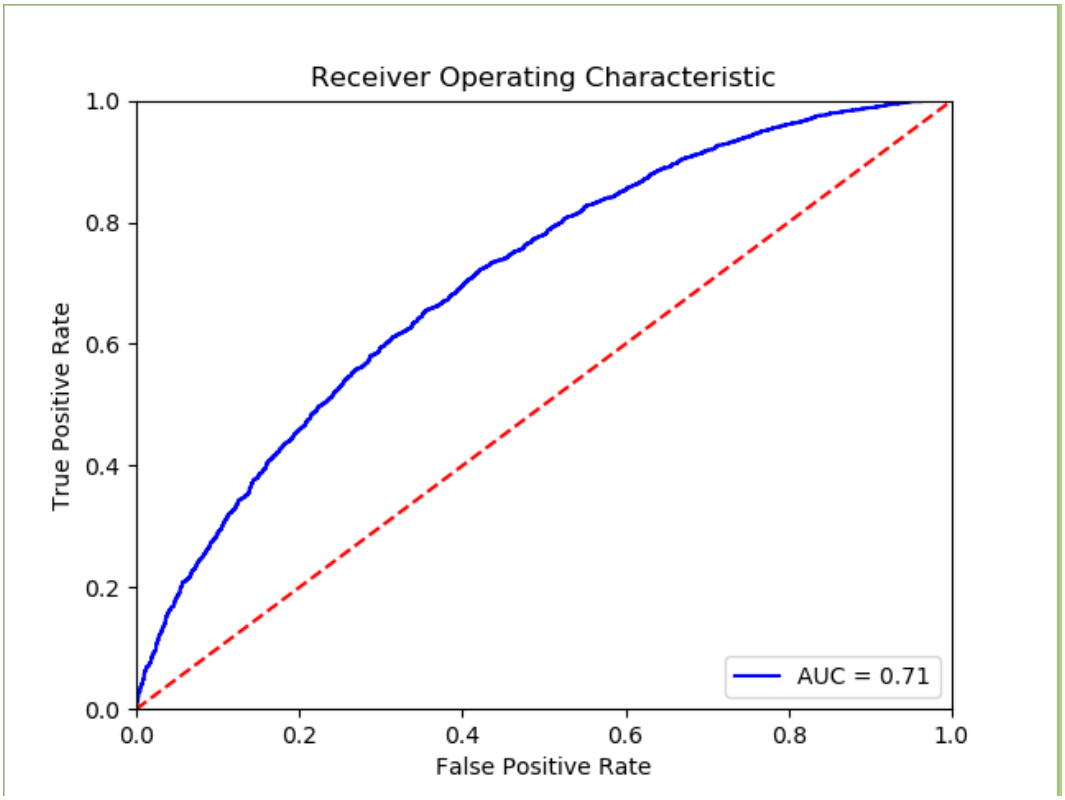
	Positive	negative
positive	True positive - 1890	False positive - 1221
negative	False negative - 1166	True negative - 2061

Данный лес обладает худшим показателем точности из всех лесов.



5)LGBM FOREST

	precision	recall	F1-score
0.0	0.64	0.65	0.65
1.0	0.66	0.65	0.65
Accuracy			0.65

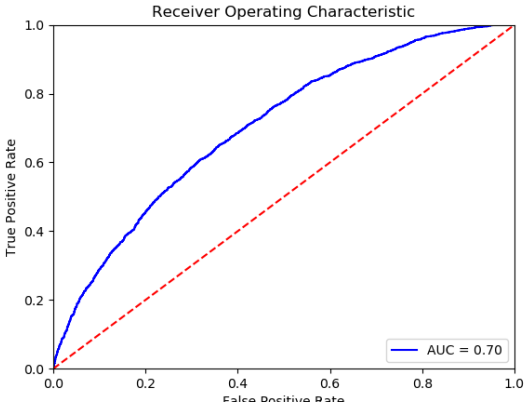


Матрица ошибок

	Positive	negative
positive	True positive - 2000	False positive – 1165
negative	False negative - 1059	True negative - 2114

6)CATBOOST

	precision	recall	F1-score
0.0	0.66	0.63	0.64
1.0	0.64	0.67	0.65
Accuracy			0.65



Матрица ошибок

	Positive	negative
positive	True positive - 1980	False positive – 1210
negative	False negative - 1019	True negative - 2129

6 и 5 обладают почти одинаковыми показателями точности, что значит данные методы могут друг друга взаимозаменять.

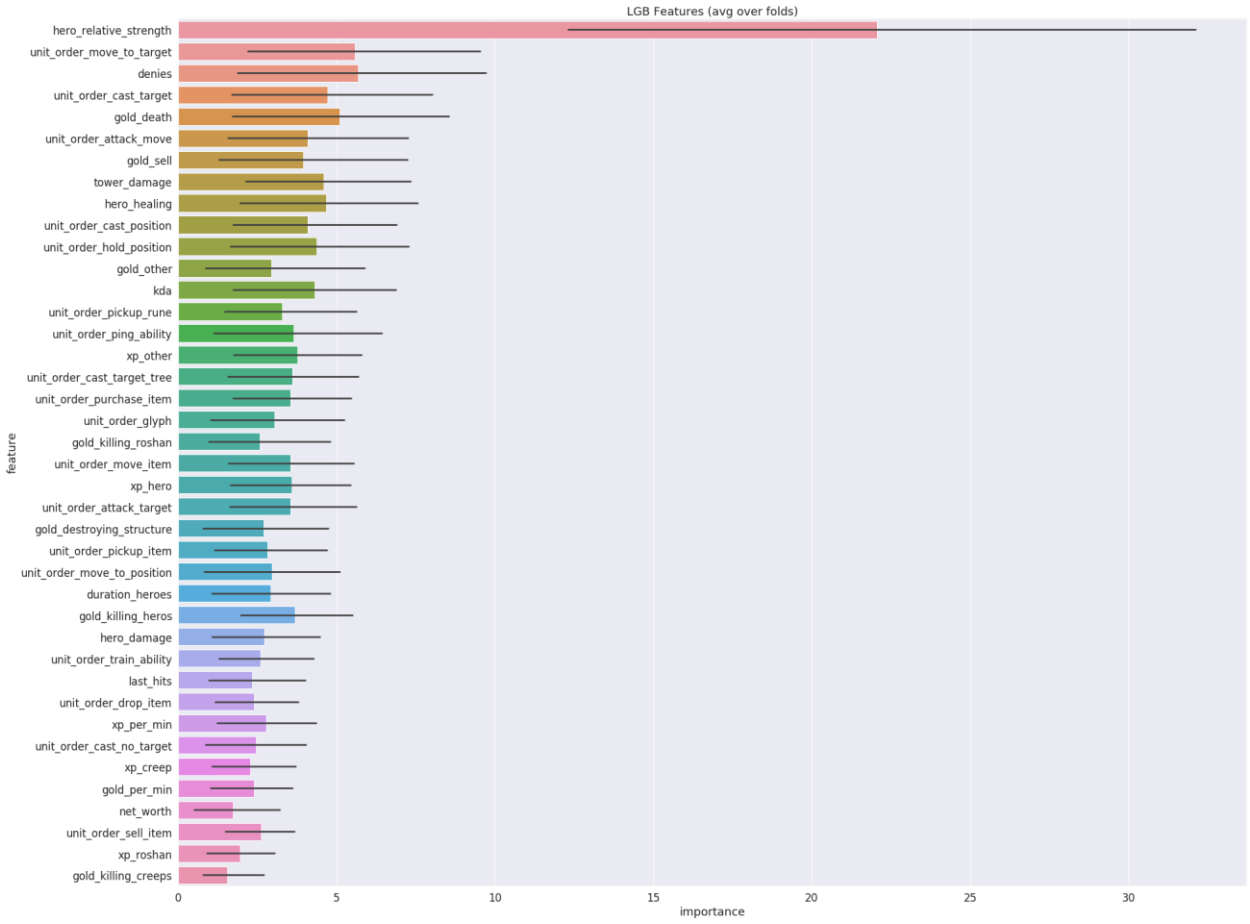


Рисунок 10 – Диаграмма полезности признаков

С помощью случайного леса мы можем построить диаграмму, на которой признаки отсортированы по критерию полезности. С ее помощью нам становится понятно, какие признаки использовать, а какие нет.

С помощью данной диаграммы мы можем заметить что большую полезность мы можем видеть на тех признаках, которые выражают сыгранность команды, чем те признаки которые привязаны к самому герою. Например как:

Пример командных признаков:

1)Hero relative strength - показывает характеристику успеха одних героев относительно других, а именно у каких пар или троек героев самый большой процент побед. Данный признак является наиболее полезным среди всего нашего списка.

2)KDA – показывают командную сыгранность игроков.

Пример личных признаков:

1)Net worth – Признак, который показывает общую ценность персонажа в конце матча

2)Gold_killing_creeps- Признак, который показывает сколько денег получил герой за убийство крипов(крипы – разновидность существ в игре Dota 2, за убийство которых игрок получает фиксированное количество золота)

Заключение

В ходе работы над проектом было проведено полноценное исследование вопроса применимости методов машинного обучения к предсказанию исхода командного киберспортивного соревнования.

Исследование демонстрирует пользу систематического подхода к подбору признаков для предсказания, а также сравнивает несколько основных моделей машинного обучения. Модели, применяемые в случае линейно разделимых наборов данных (линейная и логистическая регрессия) показывают не самые лучшие результаты, что означает, что два рассматриваемых класса линейно неразделимы. В качестве нелинейных классификаторов были выбраны вариации алгоритма решающего дерева и случайного леса из-за описанных ранее преимуществ. Эти алгоритмы машинного обучения показывают результаты, превосходящие результаты линейных классификаторов, достаточные для того, чтобы сказать, что эта категория моделей хорошо подходит для решения задач на структурных наборах данных. Все результаты были получены с помощью стандартных методов расчета метрик в задачах машинного обучения.

Так как предмет изучения – это командное соревнование, с точки зрения предметной области, исследование можно улучшить, добавив больше признаков, описывающих сыгранность команды. Кроме того, еще одним направлением дальнейшего исследования является решение задачи регрессии – например, предсказания количества нанесенного урона или очков у двух команд и принятие решения на основе этого. Наконец, возможно проверить работу реализованных алгоритмов на больших наборах данных с неофициальной статистикой матчей.

Список использованных источников:

1. http://statistica.ru/theory/osnovy-lineynoy-regressii/
2. http://baguzin.ru/wp/prostaya-linejnaya-regressiya/
3. https://habr.com/ru/post/343752/#1-mnogomernaya-lineynaya-regressiya
4. https://habr.com/ru/company/io/blog/265007/
5. https://dyakonov.org/2016/11/14/%D1%81%D0%BB%D1%83%D1%87%D0%B0%D0%B9%D0%BD%D1%8B%D0%B9-%D0%BB%D0%B5%D1%81-random-forest/