ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего образования

Национальный исследовательский университет «Высшая школа экономики»

Факультет экономических наук
Образовательная программа «Экономика»

КУРСОВАЯ РАБОТА

«Методы интерпретации моделей машинного обучения»

Студентка группы БЭК171
Махнева Елизавета Александровна

Научный руководитель:
Соколов Евгений Андреевич

Москва 2020

Содержание

1 Введение

Машинное обучение может найти свое применение практически во всех сферах деятельности человека. Модели помогают обнаруживать мошеннические операции, предсказывать диагноз пациента и многое другое. Однако с развитием машинного обучения создаваемые модели становятся все более сложными. Они показывают высокое качество, однако перестают быть понятными для человека.

Данный недостаток оказывается важным с разных точек зрения. В первую очередь, мы теряем возможность объяснить, как именно модель приняла то или иное решение. Мы не понимаем модель, а значит, не можем контролировать ее обучение напрямую – лишь перестраивая ее структуру, надеясь изменить результат в нужную сторону.

2 Методы интерпретации

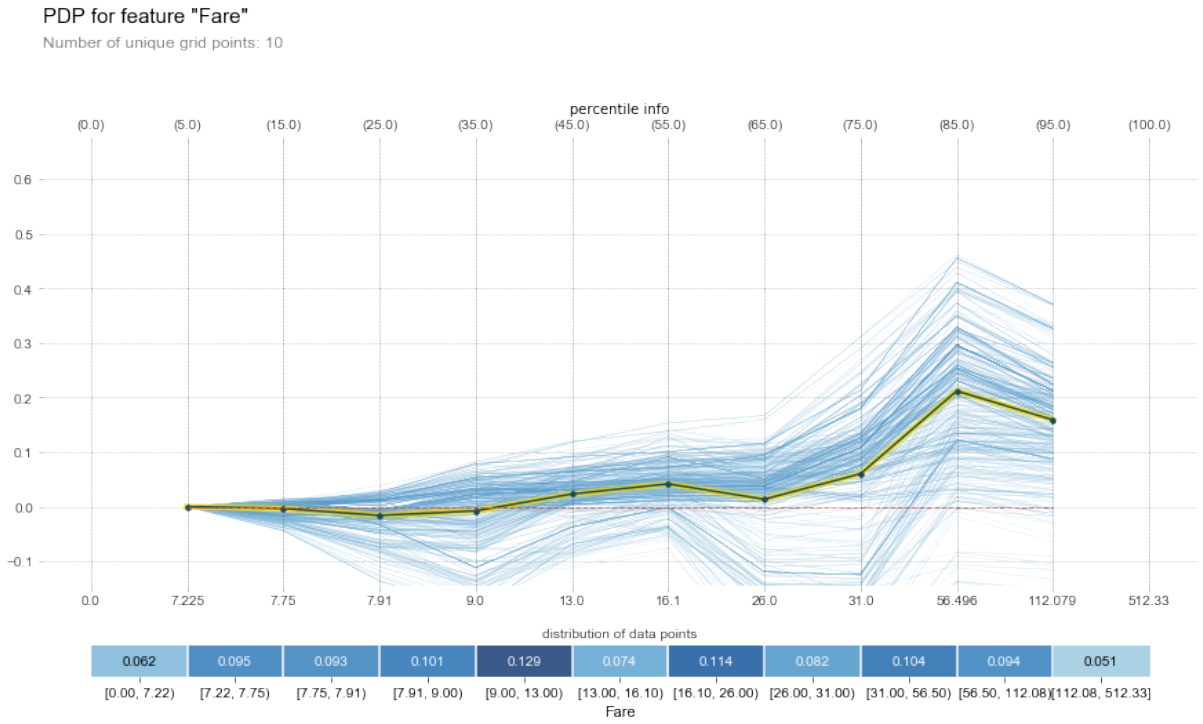
2.1 PDP

PDP (Partial Dependence Plot, график частичной зависимости) – график, который показывает зависимость прогноза модели от значения отдельного признака. С его помощью мы можем понять, как некоторый признак влияет на предсказание. Данный график можно изобразить для двух либо трех признаков из имеющихся.

2.1.1 Идея

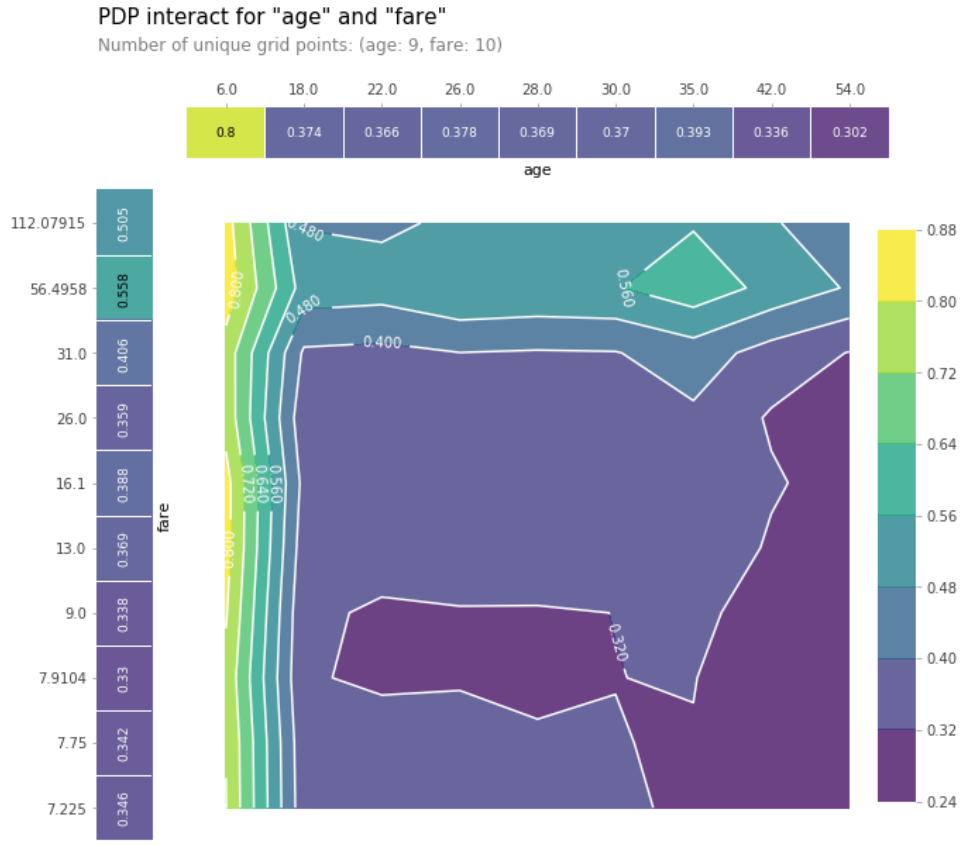
Визуализация – это отличный способ интерпретации. Если мы хотим понять, как признаки влияют на результат, можно посмотреть, как меняется прогноз от изменения одного признака при прочих равных. В идеальной ситуации мы бы построили график зависимости результата от всех признаков и меняли бы только один. Однако мы сталкиваемся с проблемой: если признаков больше двух, построить график не получится. Поэтому чтобы сохранить возможность визуализации, можно анализировать зависимость результата от одного признака без учета влияния остальных, построив график зависимости от одного признака. Аналогично можно изучать влияние одновременно двух признаков, построив трехмерный график.

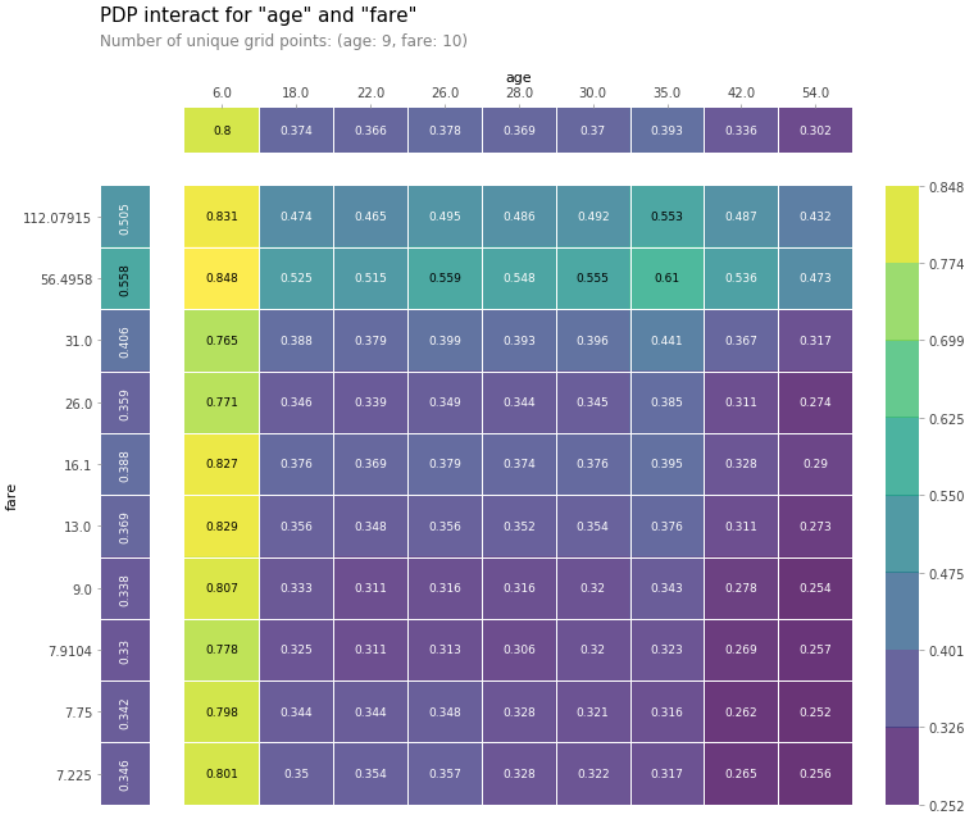
Пример PDP для одного признака



Желтая линия показывает, как в среднем выбранный признак «Fare» влияет на предсказание. Для упрощения визуализации значения признака по всем объектам были разбиты по квантилям – так график для непрерывного признака становится более наглядным. По оси ординат указан непосредственно эффект признака: значение «-0.1» указывает на то, что признак уменьшает итоговое предсказание на 0.1. То есть данный график показывает «предельный эффект» признака. В данном случае можно сделать вывод, что «Fare» незначительно влияет при маленьких значениях признака. Но при возрастании начинает вносить ощутимый вклад в предсказание – и зависимость при этом скорее положительная, но нелинейная.

Примеры PDP для двух признаков





Отличие данных графиков от предыдущего заключается в том, что он показывает не предельные эффекты, а непосредственно предсказания модели. Первый график показывает линии уровня получившейся функции двух аргументов. Второй – упрощенное представление линий уровня в виде сетки. Шкалы над и слева от графика показывают

влияние каждого признака по отдельности – сжатое представление графика для одного признака. По обоим графикам видно, что наименьшее значение предсказания достигается, когда значение «fare» наименьшее, а «age» – наибольшее. Также можно отметить экстремум функции, который достигается при «fare» ≈ 56.4958 и «age» ≈ 35 – зависимость нелинейная от обоих аргументов.

2.1.2 Принцип работы

Пусть $x = (x_1, \ldots, x_d)$ – вектор признаков объекта. У нас есть модель $f(x) : \mathbb{R}^d \to \mathbb{R}$. Мы хотим понять, как признаки x_1 и x_2 влияют на предсказание модели. Обозначим $x_r = (x_3, \ldots, x_d)$ за вектор остальных признаков.

Нам нужно получить функцию зависимости предсказания от одного-двух признаков при зафиксированных остальных: $g(x_1, x_2) = f(x_1, x_2 | \, x_r)$. Но если x_1 и/или x_2 зависимы с признаками из x_r, то возникает проблема. При изменении анализируемого признака меняется и зависимый с ним, который мы не рассматриваем – мы не сможем рассмотреть чистый предельный эффект одного признака, на него всегда будет наложен эффект другого предиктора. Поэтому одной из предпосылок метода является независимость исследуемых признаков от остальных.

Но даже с предпосылкой о независимости признаков функция $g(x_1, x_2)$ не будет показывать точный результат, так как предельные эффекты предикторов разные для разных объектов выборки. Поэтому мы рассмотрим, как влияют анализируемые признаки на среднее предсказание. То есть найдем матожидание предсказания модели при фиксированных исследуемых признаках (как констант с точки зрения матожидания):

$$\bar{g}(x_1, x_2) = \mathbb{E}(f(x_1, x_2, x_r) \, | x_r)$$

Таким образом, мы получим функцию, которая показывает предельные эффекты признаков для среднего предсказания. Но чтобы найти матожидание, мы должны знать истинные распределения признаков. Поскольку нам недоступна данная информация, можно воспользоваться методом Монте-Карло, чтобы примерно оценить искомую функцию:

$$\hat{g}(x_1, x_2) = \frac{1}{n} \sum_{i}^{n} f(x_1, x_2, X_r^{(i)}),$$

где $X_r^{(i)}$ – i строка матрицы X_r, содержащей признаки x_r для всех объектов выборки.

Результат: функция показывает, как исследуемые признаки в среднем влияют на результат работы модели. Мы можем построить ее график, чтобы более наглядно посмотреть на влияние предикторов на предсказание.

2.1.3 Реализация

Реализации данных графиков есть в разных библиотеках: `pdpbox`, `sklearn`, (`sklearn.inspection.plot_partial_dependence`, `sklearn.ensemble.partial_dependence.plot_partial_dependence`)

Библиотека `pdpbox` предоставляет более аккуратное и более наглядное представление графиков. Инструкции по установке можно найти здесь.

2.2 LIME

LIME (Local Interpretable Model-Agnostic Explanations) – метод, показывающий вклад признаков в отдельное предсказание, работающий с любой моделью.

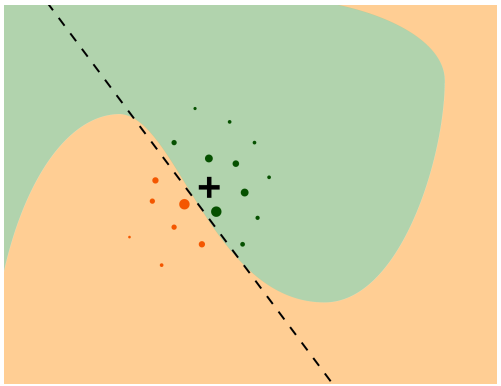
2.2.1 Идея

Результаты некоторых моделей легко интерпретировать. Например, в линейной регрессии можно посмотреть на веса. Они показывают, насколько изменится предсказание при изменении признаков. Так для каждого конкретного предсказания можно понять, почему модель выдала именно такой результат – виден непосредственный вклад каждого признака.

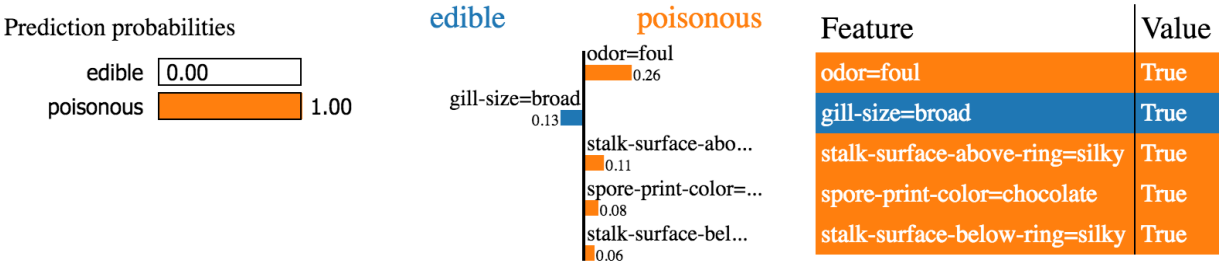
Но не все модели легко интерпретировать. Например, некоторые архитектуры нейронных сетей. Они зачастую значительно превосходят линейные модели, но при этом сама структура модели представляет собой «черный ящик» – непонятно, как именно модель сформировала предсказание, какие признаки сильнее повлияли на решение нейронной сети.

Идея состоит в том, чтобы перенести свойство интерпретируемости простых моделей на более сложные. Мы можем обучить интерпретируемую модель по выборке, где ответами являются предсказания сложной модели. В процессе обучения модель анализирует зависимости непосредственно между признаками и предсказаниями сложной модели. Тогда мы сможем интерпретировать результаты простой модели, которые являются аппроксимацией предсказаний сложной.

Возникает проблема: сложная модель выявляет зависимости, которые, например, линейная модель может не уловить. Идея данного метода заключается в том, чтобы обучать более простую модель в некоторой окрестности исследуемого объекта, предполагая что на очень локальном уровне нет сложных зависимостей. Например, дифференцируемые функции можно линеаризовать в окрестности заданной точки – можно использовать линейную модель на локальном уровне.



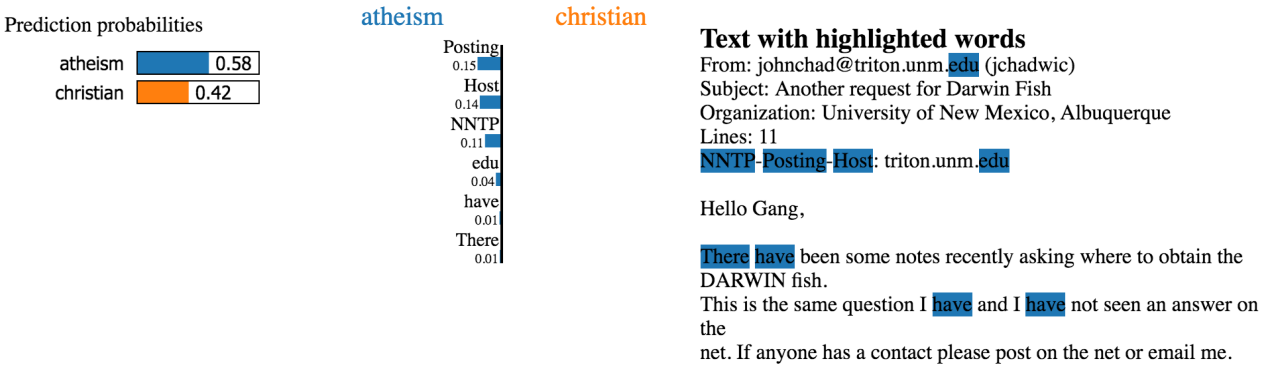
Пример LIME для табличных данных



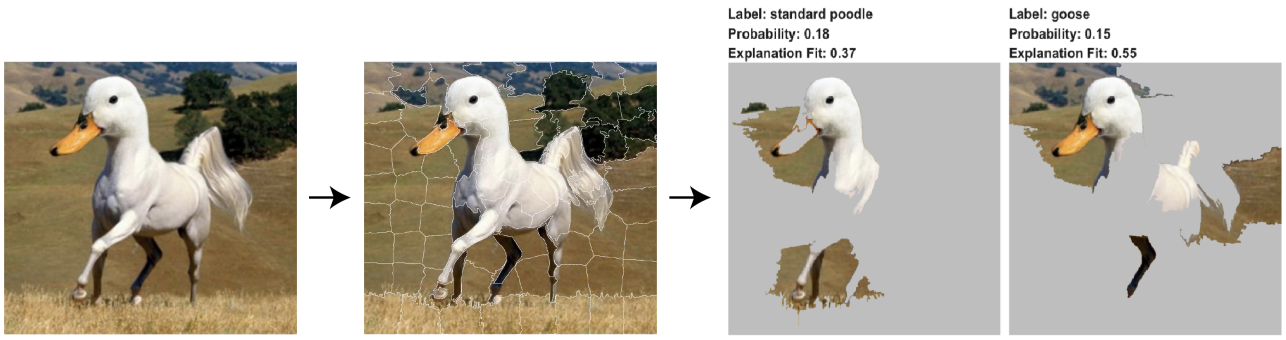
В данном случае решалась задача бинарной классификации. LIME вывел пять самых значимых признаков и показал, каким образом они повлияли на предсказание. Обученная модель показала, что исследуемый объект принадлежит к классу «poisonous». И по анализу LIME понятно почему: только один признак из выведенных увеличил вероятность принадлежности объекта к классу «edible» и только на 0.13. В то же время оставшиеся четыре признака повлияли на вероятность объекта быть «poisonous», причем суммарно увеличив вероятность на 0.51. Данная интерпретация позволяет понять, адекватна ли

модель, которую мы получили, действительно ли релевантные признаки оказались значимыми.

Пример LIME для текстов



Пример LIME для картинок



2.2.2 Принцип работы

У нас есть модель $f(x) : \mathbb{R}^d \to \mathbb{R}$ и объект $x \in \mathbb{R}^d$, предсказание для которого нужно интерпретировать. Мы хотим найти модель g из класса интерпретируемых моделей G, чтобы получить из нее объяснение результата более сложной модели (например, посмотреть на веса признаков).

Сложность моделей обычно обратно зависит от ее интерпретируемости. Например, линейную модель с 2-3 предикторами гораздо проще интерпретировать, чем модель с 10 и более предикторами. Поэтому нам нужно не просто использовать более простую модель, но и преобразовать исходное пространство признаков: $x \to x'$. Нам не подходят методы снижения размерности, которые представляют признаки в уже неинтерпретируемом виде (PCA, UMAP и пр.). В данной задаче можно уменьшить количество признаков и преобразовать их.

Чтобы уменьшить количество признаков, мы можем случайно выбирать некоторые из них. Либо комбинировать их между собой, при этом обращая внимание на интерпретируемость новых признаков. Для каждого типа данных подобное преобразование может проходить по-разному: для изображений – несколько пикселей могут объединяться в один суперпиксель, для текстов – символы объединяться в токены (например, слова – они хорошо интерпретируются).

Для преобразования признаков существует большое количество методов. Например, приведение непрерывных переменных к дискретному виду. Основная цель подобных преобразований – уменьшить множество значений признаков. Авторы алгоритма предлагают приводить предикторы к еще более упрощенному виду: использовать вместо признака

дамми-переменную его наличия. Именно поэтому наша простая модель $g(x') : \{0,1\}^{d'} \to \mathbb{R}$ работает с интерпретируемым представлением предсказания $x' \in \{0,1\}^{d'}$, где обычно $d' << d$. Дополнительно в задаче вводится мера сложности $\Omega(g)$ искомой модели как регуляризация.

Чтобы определить окрестность рядом с x, внутри которой мы можем использовать простую модель, введем меру близости $\pi_x(z)$ между объектом x и его соседом z. И наконец определим нашу функцию потерь, которую мы будем оптимизировать: $L(f, g, \pi_x)$ – разница между моделями f и g в окрестности, заданной π_x. Тогда в целом задача алгоритма выглядит следующим образом:

$$explanation(x) = \xi(x) = \underset{g \in G}{\operatorname{argmin}}(L(f, g, \pi_x) + \Omega(g))$$

Одной из особенностей алгоритма является его независимость от модели, которую необходимо интерпретировать. Поэтому мы не можем приписывать модели f никакие свойства. Вместо этого мы будем аппроксимировать ее, искусственно создавая объекты в окрестности x', переводя их в исходное пространство признаков и получая для них предсказания из f.

Понятие окрестности носит абстрактный характер, поэтому чтобы учесть расстояние между объектами на практике, мы будем использовать меру близости $\pi_x(z)$ как веса: чем ближе объект к x, тем больший вклад он вносит в функцию потерь.

Гиперпараметры в задаче:

- G – класс интерпретируемых моделей: линейные модели, решающие деревья и др.
- d' – количество признаков в новом пространстве
- $\pi_x(z)$ – мера близости: например, ядра (гауссово, логистическое и др.)
- $D(x, z)$ – расстояние между объектами, используемое при расчете меры близости: евклидова метрика, косинусное расстояние и др.
- L – функция потерь: MSE, MAE и др.

Результат: мы получаем алгоритм, который изучает работу более сложной модели и интерпретирует ее с некоторой погрешностью – можно изучать влияние признаков на отдельные предсказания.

2.2.3 Реализация

Данный метод реализован в библиотеке `lime`. Она содержит в себе методы для работы с разными типами данных: `lime.lime_tabular`, `lime.lime_text`, `lime.lime_image`. Инструкцию по установке можно найти здесь.

2.3 SHAP

SHAP (SHapley Additive exPlanations) – метод, оценивающий вклад признаков в предсказания модели на основе значений Шэпли.

2.3.1 Идея

Предсказание модели формируется на основе признаков. Если мы хотим узнать влияние отдельного признака, мы можем построить предсказание модели с ним и без него и затем

посмотреть, как меняется результат. Но модель может быть слишком сложной, чтобы мы могли оценить влияние предиктора по одному объекту, регулируя один признак при фиксированных остальных. Правильнее рассмотреть все возможные комбинации признаков: их разные значения, наличие/отсутствие, чтобы понять, как в каждом из перечисленных случаев добавление и исключение признака влияет на предсказание.

Рассматривая влияние в каждом конкретном случае, мы получаем огромное количество предельных эффектов, что тяжело интерпретируется. Поэтому можно рассмотреть, какой в среднем эффект оказывает включение признака в модель. Тогда мы получим средневзвешенную оценку вклада отдельного признака в предсказание, что позволит проинтерпретировать результат работы модели.

Однако для такого способа нужно либо обучить количество моделей, пропорциональное количеству всех комбинаций признаков – которое растет экспоненциально с увеличением количества признаков. Либо оставлять пропущенные значения вместо исключенных признаков, что может плохо отразиться на качестве работы модели. Чтобы избежать этого, мы можем несколько видоизменить наше решение, не меняя цель. Вместо обучения дополнительной модели с меньшим количеством предикторов мы оценим ее предсказание, случайно изменяя исключенные признаки и усреднив результат.

Авторы алгоритма также вносят новую идею о способе расчета вклада признаков в предсказание: данные значения можно посчитать с помощью линейной регрессии с использованием матрицы весов определенного вида.

2.3.2 Значения Шэпли (Shapley values)

Пусть S – множество не исключенных из модели признаков, в которое не входит исследуемый i-ый признак. Чтобы найти разницу предсказаний для признаков из S и из $S \cup \{i\}$, куда входит i-ый признак, нам нужно обучить две модели, f_S и $f_{S\cup\{i\}}$, соответственно. Тогда для объекта x мы получим:

$$f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S),$$

где $x_{S\cup\{i\}}$ – объект x, у которого оставили только признаки из $S\cup\{i\}$, x_S – только признаки из S.

Как отмечалось ранее, значения данного выражения могут меняться при разных комбинациях признаков, поэтому мы усредним ее для всех возможных случаев. Считая, что F – множество всех признаков, получим итоговое выражение для i-го признака:

$$\phi_i = \frac{1}{|F|} \sum_{S\subseteq F\setminus\{i\}} \binom{|F|-1}{|S|}^{-1} (f_{S\cup\{i\}}(x_{S\cup\{i\}}) - f_S(x_S)),$$

где $|F|$ – количество элементов в множестве F, $|S|$ – в множестве S.

Коэффициент:

$$\binom{|F|-1}{|S|} = \frac{(|F|-1)!}{|S|! \cdot (|F|-|S|-1)!}$$

равен количеству всех возможных комбинаций признаков без учета исследуемого. Поделив на него, а также на количество признаков получаем среднее значение вклада признака.

Данный коэффициент также можно интерпретировать как вес комбинации при расчете среднего. Биномиальный коэффициент, равный количеству комбинаций, для одного или $|F|-1$ признаков меньше, чем для любого другого числа признаков. При делении на коэффициент мы получаем большее значение, который указывает на больший вес подобных

комбинаций. Это имеет смысл, так как больше информации мы получаем, рассматривая влияние признаков по отдельности: либо оставляя только его, либо исключая все кроме него. С увеличением количества признаков в S вес комбинации убывает.

Таким образом, мы получили величину, показывающую, какой в среднем вклад вносит конкретный признак в предсказание. Данное значение пришло из теории игр и носит название значение Шэпли (Shapley value). Они показывают, какой вклад внес в общий выигрыш отдельный игрок при кооперации. В нашей задаче мы рассматриваем предсказание модели как выигрыш, а признаки как игроков.

2.3.3 Принцип работы

Метод SHAP несколько отличается от использования значений Шэпли напрямую для интерпретации вклада признаков – он модифицирует классический подход. Рассмотрим SHAP подробнее.

Вместо использования большого количества моделей мы обучим одну модель и далее будем пользоваться только ее предсказаниями. Тогда наша модель представима в виде $f : \mathbb{R}^d \to \mathbb{R}$. Вместо исключения из нее признаков (в таком случае нам пришлось бы переобучать модель) мы оставим данные признаки в виде случайных величин и посчитаем математическое ожидание предсказания при фиксированных включенных в модель признаках.

Для упрощения расчетов мы не будем рассматривать разные значения предикторов, а бинаризуем их представление, обозначив за «1» наличие предиктора и за «0» его отсутствие: $x \to x'$, $x' = \{1\}^d$. Обозначим за $z' \in \{0,1\}^d$ объект, у которого мы учитываем только признаки из S при формировании предсказания: $z_j = \begin{cases} 1, \text{если } j \in S \\ 0, \text{если } j \notin S \end{cases}$.
То есть z' – одна из возможных комбинаций предикторов.

Обученная модель работает с исходным видом признаков, поэтому нам нужно также восстанавливать значения исходных признаков по бинарному вектору. Введем для этого функцию $h_x(z') = z$, где x – исходный вектор исследуемого объекта, z' – бинарный вектор, в котором некоторые признаки заменены нулями, z – представление вектора z' в исходном пространстве признаков. Функция h_x вместо единиц восстанавливает значения из вектора x, а вместо нулей оставляет признак как некоторую переменную (случайную величину). Тогда наше ожидаемое предсказание для z представимо в виде математического ожидания:

$$\bar{f}(z) = \mathbb{E}(f(z)|\, z_S)$$

Мы знаем конкретные значения признаков z_S, так как функция h_x перенесла их из объекта x. Найдя математическое ожидание мы можем подставить данные значения, чтобы получить условное математическое ожидание, которое и будет оценкой нашего предсказания. И уже данную формулу мы можем использовать при расчете необходимых значений.

Поскольку мы не знаем истинное распределение, чтобы иметь возможность посчитать матожидание, аппроксимируем его оценкой. Для этого немного изменим функцию h_x – теперь вместо нулей она будет проставлять случайные значения соответствующих признаков. Тогда для каждого множества S мы сможем получить несколько предсказаний модели и усреднить их:

$$\hat{f}(z) = \frac{1}{k}\sum_{i=1}^{k} f(z_{F\setminus S}, z_S),$$

где $z_{F\setminus S}$ – исключенные из модели признаки, вместо которые подставлены случайные значения, z_S – включенные в модель признаки, значения которых известны из исследуемого

объекта x.

Kernel SHAP. Авторы алгоритма показали, что есть упрощенный способ найти значения Шэпли: с помощью взвешенного МНК. В нем аналогично перебираются разные комбинации z' – так формируется выборка Z для регрессии. Зависимой переменной является предсказание модели для сгенерированной выборки (переведенной в исходное пространство признкаов): $y_i = f(h_x(z'_i)) = f(z_i)$, где z'_i – строка матрицы Z. Тогда решением нашей задачи является вектор:

$$\phi = (Z^T W Z)^{-1} Z^T W y$$

Чтобы коэффициенты в регрессии соответствовали искомым значения, веса должны быть заданы диагональной матрицей W:

$$w_{ii}(z'_i) = \frac{|F| - 1}{\binom{|F|}{|S_i|} \cdot |S_i| \cdot (|F| - |S_i|)} = \frac{|F| - 1}{|F|} \cdot \frac{|S_i - 1|! \cdot (|F| - |S_i| - 1)!}{(|F| - 1)!},$$

где $|S_i|$ – количество элементов в множестве S_i, то есть количество ненулевых элементов в z'_i.

В данном случае коэффициент при константе будет показывать предсказание модели при отсутствии признаков. Коэффициенты при признаках являются соответствующими значениями Шэпли, которые можно использовать для интерпретации их вклада в предсказание.

Ранее мы уже говорили о том, что комбинации, в которых либо почти все нули, либо почти все единицы имеют больший вес при расчете значений Шэпли. Данный факт сохраняется и в линейной регрессии, поэтому формируя выборку можно отдавать предпочтение именно данным примерам – в случае, если мы ограничены в количестве наблюдений.

Интересно, что если записать задачу в более общем виде:

$$\sum_{z' \in Z} (f(h_x(z')) - g(z'))^2 \cdot \underbrace{\frac{|F| - 1}{|F|} \cdot \frac{|S - 1|! \cdot (|F| - |S| - 1)!}{(|F| - 1)!}}_{w(z') = \pi_x(z')} = L(f, g, \pi_x) + \Omega(g) \to \min_g$$

мы получим ровно ту же задачу, что решает LIME. То есть при определенных гиперпараметрах LIME и SHAP эквивалентны друг другу.

Существуют и другие модификации SHAP: TreeSHAP, Linear SHAP, Low-Order SHAP, Max SHAP, Deep SHAP и др. В отличие от классического подхода они специфицированы для отдельных классов моделей, что позволяет снизить время работы алгоритма.

2.3.4 Реализация

Данный метод реализован в библиотеке `shap`. Универсальный метод содержится в модуле: `shap.KernelExplainer`. Модицикации перечисленные ранее являются более специлизированными:

- ◇ `shap.TreeExplainer`: XGBoost/LightGBM/CatBoost/scikit-learn/pyspark models
- ◇ `shap.DeepExplainer`: TensorFlow/Keras models
- ◇ `shap.GradientExplainer`: TensorFlow/Keras/PyTorch models
- ◇ и др.

.

Инструкцию по установке можно найти здесь.

3 Анализ данных и интерпретация

Рассмотрим описанные методы на конкретной задаче. Построим относительно сложную модель (градиентный бустинг) для бинарной классификации объектов.

3.1 Данные

Для анализа был выбран датасет, содержащий информацию о клиентах разных отелей за 2015-2017 года. Описание датасета можно найти в приложении. Задача: предсказать, отменит ли клиент бронь. Датасет был предварительно обработан:

- ◇ Удалены переменные company (много пропусков), agent (ID туристических агенств – зависит от приведенной классификации), reservation status, reservation status date (данная информация обычно бывает известна уже после отмены либо отъезда гостя), arrival date year, arrival date month (данные представлены только по 2015-2017 годам, слишком мало информации для предсказаний на более дальние периоды; месяц аналогично несет мало информации – вместо данных переменных осталась переменная arrival date week number)
- ◇ Стандартизированы все числовые переменные, кроме arrival date week number, arrival date day of month (порядковые переменные), is repeated guest (бинарная переменная)
- ◇ Приведены к числовому виду порядковые переменные meal, deposit type, reserved room type, assigned room type
- ◇ Приведены к бинарному виду категориальные переменные hotel, country, market segment, distribution channel, customer type
- ◇ Удалены строки с пропущенными значениями – таких строк было немного и в них была пропущена важная информация
- ◇ Удален выброс: для одного наблюдения adr < 0, что не может быть правдой, так как adr $\in [0, 1]$

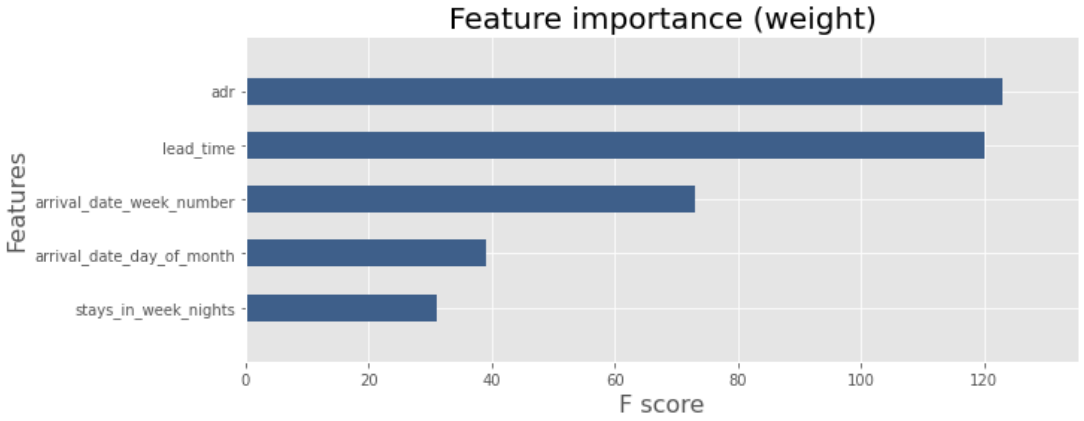
3.2 Модели

Для анализа описанных методов я выбрала модель бинарной классификации XGBoost. Подбор гиперпараметров осуществлялся с помощью кросс-валидации на тренировочных данных. Одной из проблем данных является их несбалансированность: она решалась учетом объектов положительного класса с большим весом. Посмотреть на процесс кросс-валидации и обучения модели можно здесь

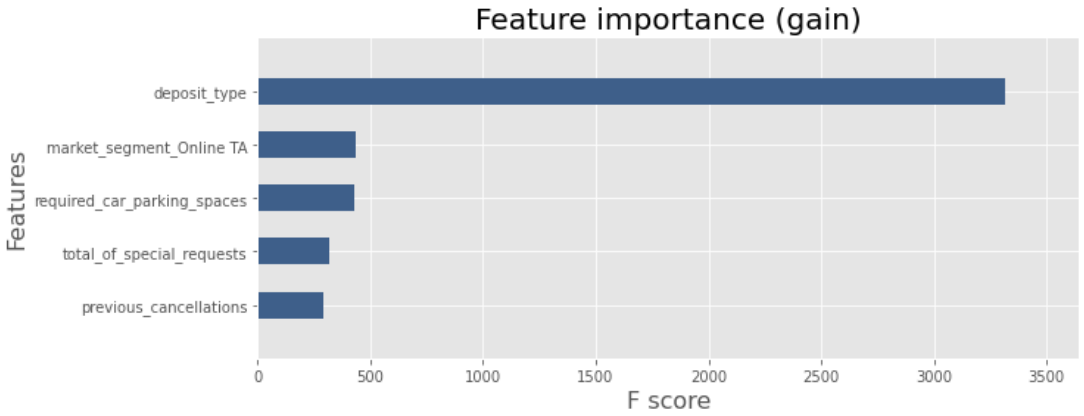
3.3 Интерпретация результатов

Модель достигла неплохого качества в задаче классификации: ROC-AUC=0.9, accuracy=0.82. Попробуем интерпретировать ее результаты.

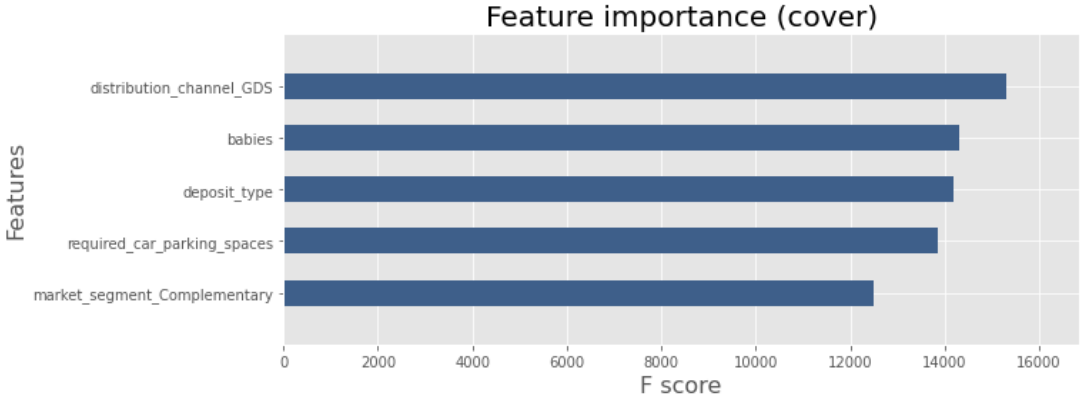
Нулевой способ это встроенный метод XGBoost, показывающий важность признаков при предсказании. Посмотрим на первые 5 самых важных признаков. Данная величина может быть расчитана тремя способами: «weight», «gain», «cover». Первый показывает, сколько раз признак появляется в дереве:



Второй – на сколько в среднем уменьшалась ошибка при использовании данного признака:



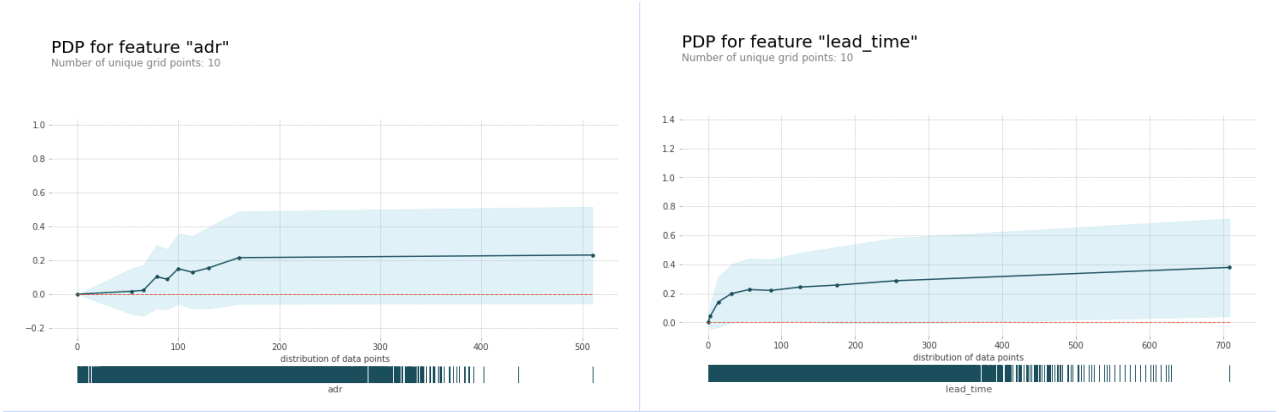
И последний – какое количество объектов выборки задействовало узлы с заданным признаком:



Теперь перейдем к описанным ранее методам. Первый – PDP. Возьмем признаки, которые сам XGBoost посчитал наиболее важными: первые два из weight (adr, lead_time), первый из gain (deposit_type) и первый из cover (distribution_channel_GDS) – они с отрывом вырываются в лидеры.

И также возьмем признаки, которые XGBoost счел самыми незначительными: последний из weight (distribution_channel_GDS, забавно – в тренировочной выборке всего 145 объектов, которым соответствует GDS. Судя по всему данный признак встречается 1-2 раза в узлах деревьев, но при этом он отсекает очень много объектов, из-за чего cover считает его важным), последние два из gain (customer_type_group, market_segment_Direct) и последний из cover (market_segment_Offline TA/TO).

Построим для них PDP.



4 Заключение

Таким образом, существующие методы интерпретации моделей машинного обучения показывают неплохие результаты. Безусловно, они имеют свои недостатки, но несмотря на это они справляются со своей задачей и с некоторой погрешностью объясняют работу сложных моделей.

Список литературы

[1] Interpretable Machine Learning | Christoph Molnar | Christoph Molnar | 2020 | all pages

[2] “Why Should I Trust You?” Explaining the Predictions of Any Classifier | ...

5 Приложения

Приложение 1. Описание данных: