

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	3
2 Методы интерпретации	5
2.1 PDP	5
2.1.1 Идея	5
2.1.2 Принцип работы	7
2.1.3 Реализация	7
2.2 LIME	8
2.2.1 Идея	8
2.2.2 Принцип работы	9
2.2.3 Реализация	10
2.3 SHAP	11
2.3.1 Идея	11
2.3.2 Значения Шэпли (Shapley values)	12
2.3.3 Принцип работы	12
2.3.4 Реализация	14
3 Пример применения	15
3.1 PDP	16
3.2 LIME	18
3.3 SHAP	18
4 Анализ данных и интерпретация	21
4.1 Данные	21
4.2 Модель	21
4.3 Интерпретация результатов	22
5 Заключение	31

1 Введение

Машинное обучение находит свое применение во многих 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, но 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 структуру, надеясь изменить результат в нужную сторону. Высокая метрика качества, как и несколько метрик одновременно, также не говорят о корректности модели. Во-первых, они могут противоречить друг другу, из-за сложнее сделать корректный вывод. Во-вторых, по-прежнему есть вероятность, что модель выучит неверную закономерность, которую мы не обнаружим, пока в нашу тестовую выборку не попадет подходящий объект.

Помимо этого, отсутствие интерпретируемости может служить препятствием на пути внедрения моделей в различные области. Если мы не знаем, почему алгоритм, управляющий беспилотным автомобилем, принял решение увеличить скорость перед пешеходным переходом, мы вряд ли начнем массово выпускать данную технику. В подобных областях использование моделей накладывает большие риски – поэтому важно осознавать, как именно алгоритмы принимают решения. Препятствием также является непонимание со стороны людей, незнакомых с машинным обучением. Человек, незнакомый с направлением, не сможет сразу понять, как работает тот или иной алгоритм. Поэтому создание методов интерпретации ускоряет распространение машинного обучения также за счет снижения порога необходимых для работы с ним знаний.

Модели используются во многих областях, где выгода от них перевешивает риски, которые они несут. Тем не менее, даже если нам не страшны несколько ошибок, совершенные машиной, мы можем упускать важные детали, которые позволяют улучшить алгоритм и избежать увеличения количества ошибок в будущем. Например, в случае смещенности выборки модель может дискриминировать некоторые типы объектов: при скрининге резюме – отказывать женщинам, при выдаче кредита – учитывать расу клиента и т.д. Закономерность, заложенная в алгоритм, некорректна. Практически невозможно выявить данную проблему без понимания того, как модель создает свое предсказание.

Интерпретация не только дает возможность проверить корректность модели, но и в паре с машинным обучением может помочь извлечь дополнительную информацию из имеющихся данных. Если мы обучим модель, а затем интерпретируем результаты ее работы, то сможем посмотреть на наши данные с другой точки зрения: какие признаки оказались важны для предсказания, какие закономерности присутствуют в выборке и т.д. Данный бонус дает некоторое понимание устройства мира. Причем его можно использовать и с целью получения новых знаний: извлечение существующих эмпирических закономерностей в природе и обществе.

Безусловно, интерпретация имеет и некоторые недостатки. Два основных – невозможность достичь абсолютной точности интерпретации и необходимость жертвовать качеством модели. Первый интуитивно понятен: мы никогда не можем быть уверены, что корректно интерпретируем «черный ящик» модели. Даже прозрачные алгоритмы интерпретируются с некоторой неточностью: например, в линейной регрессии веса не всегда могут соответствовать вкладу признака – данное правило нарушается при наличии зависимости между признаками. Второй минус: зачастую методы интерпретации применяются

к сложным, непрозрачным моделям. Поскольку сами по себе они неинтерпретуемы, приходится применять методы, которые некоторым образом воздействуют на них и зачастую снижают их качество, упрощают их.

Поэтому важно понимать, какая стоит задача в исследовании: получить высокое качество или корректную модель. В первом случае можно обойтись без интерпретации. Во втором случае интерпретация – полезный инструмент, проверяющий адекватность модель.

Как раз для задач, в которых важно понимание, существует огромное количество методов интерпретации. В данной работе мы рассмотрим три из них, что, безусловно, не является исчерпывающим списком.

2 Методы интерпретации

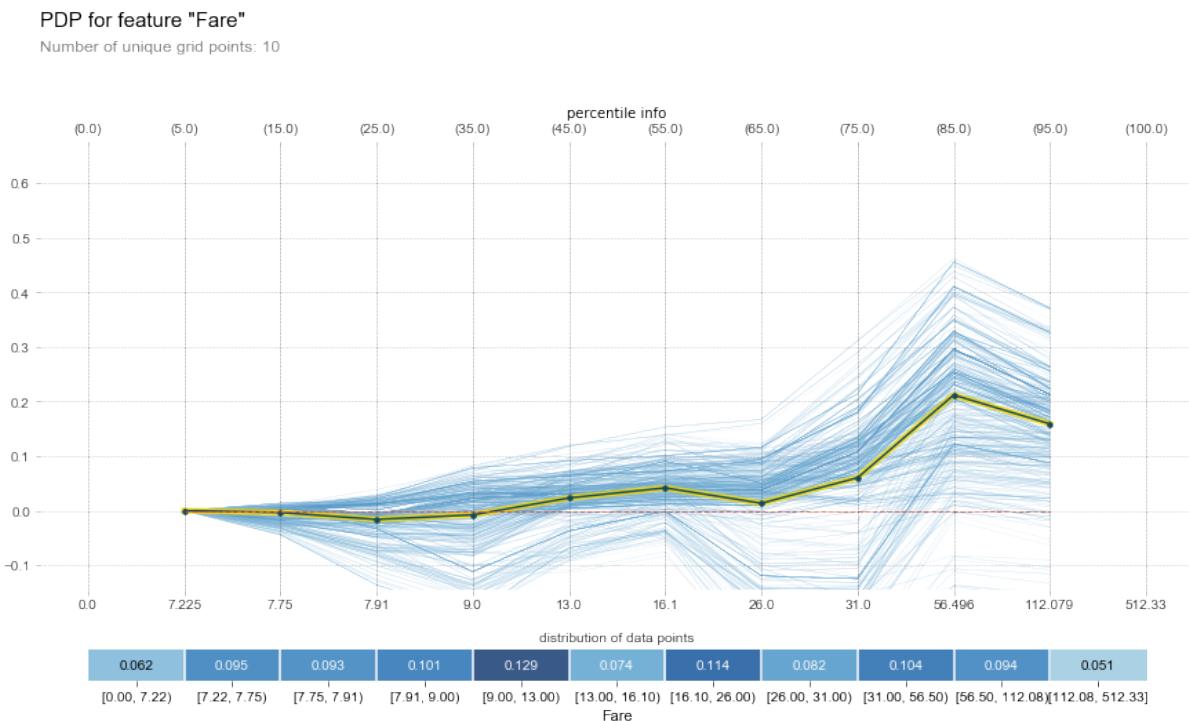
2.1 PDP

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 [2]

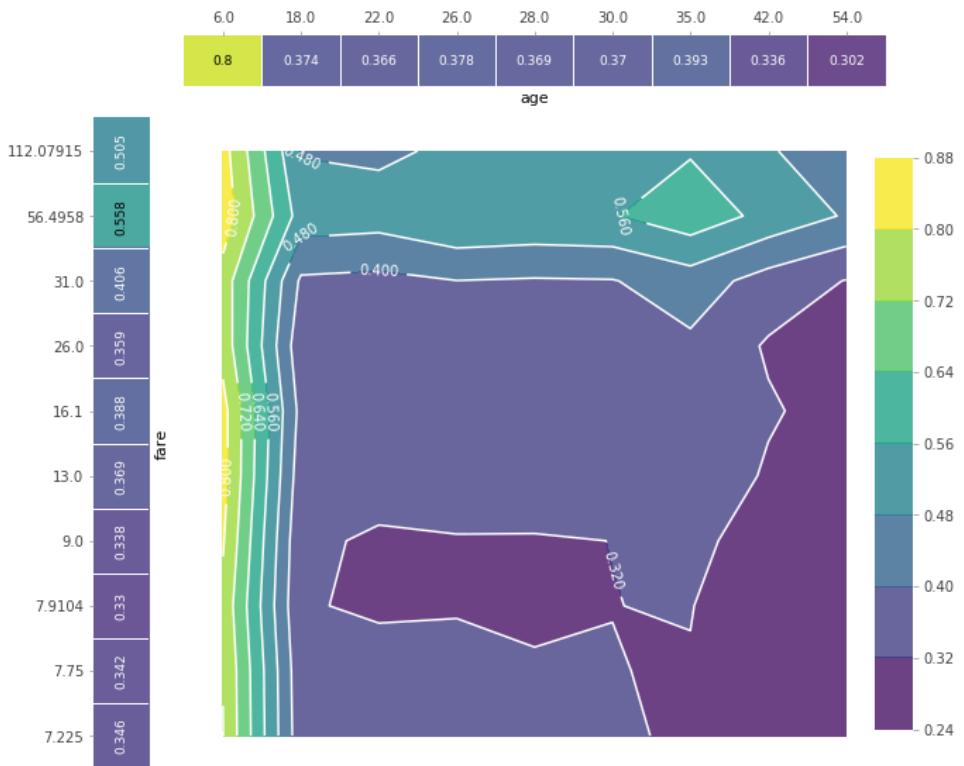


Желтая линия показывает, как в среднем выбранный признак «Fare» влияет на предсказание. Для упрощения визуализации значения признака по всем объектам были разбиты по квантилям – так график для непрерывного признака становится более наглядным. По оси ординат указан непосредственно эффект признака: значение «-0.1» указывает на то, что признак уменьшает итоговое предсказание на 0.1. То есть данный график показывает «пределенный 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 [2]

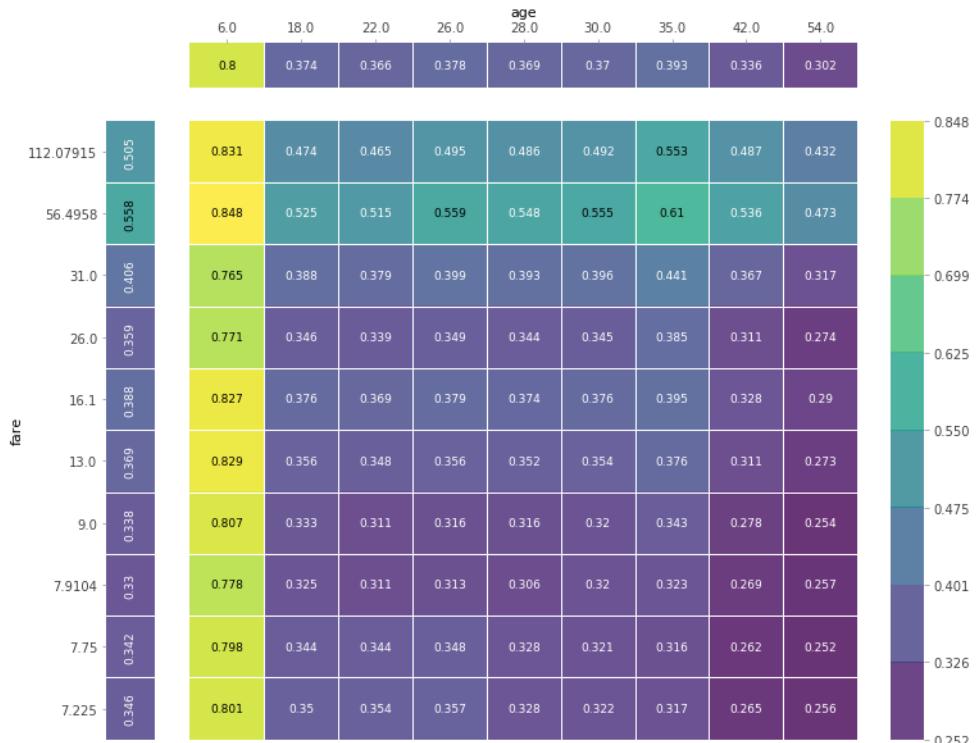
PDP interact for "age" and "fare"

Number of unique grid points: (age: 9, fare: 10)



PDP interact for "age" and "fare"

Number of unique grid points: (age: 9, fare: 10)



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 $x = (x_1, \dots, x_d)$ – вектор признаков объекта. У нас есть модель $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}$. Мы хотим понять, как признаки x_1 и x_2 влияют на предсказание модели. Обозначим $x_r = (x_3, \dots, x_d)$ 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 зависят 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 [3].

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) [3]:

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

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 [1]:

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

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

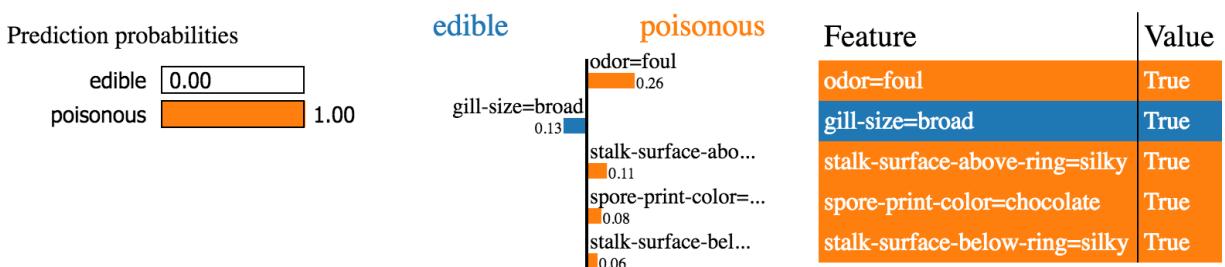
Результаты некоторых моделей легко интерпретировать. Например, в линейной регрессии можно посмотреть на веса. Они показывают, насколько изменится предсказание при изменении признаков. Так для каждого конкретного предсказания можно понять, почему модель выдала именно такой результат – виден непосредственный 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 [4]. Например, дифференцируемые функции можно линеаризовать в окрестности заданной точки – можно использовать линейную модель на локальном уровне.

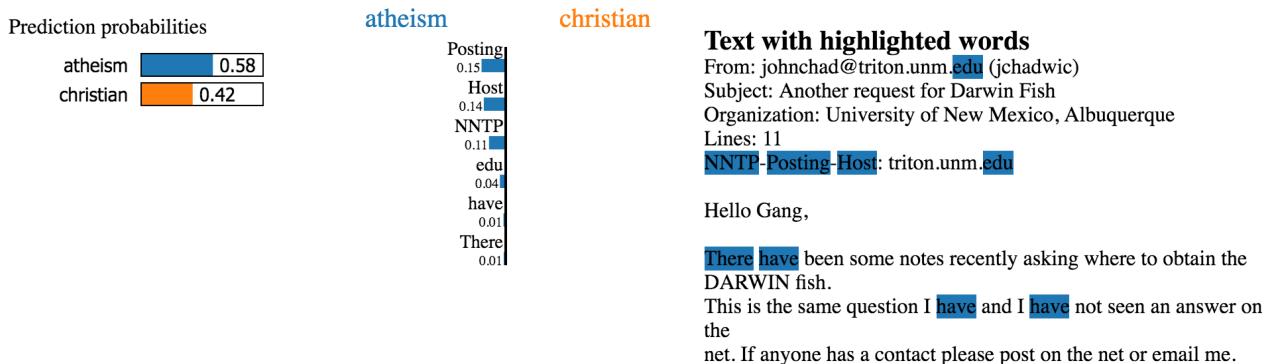
Пример LIME для табличных данных [5]



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

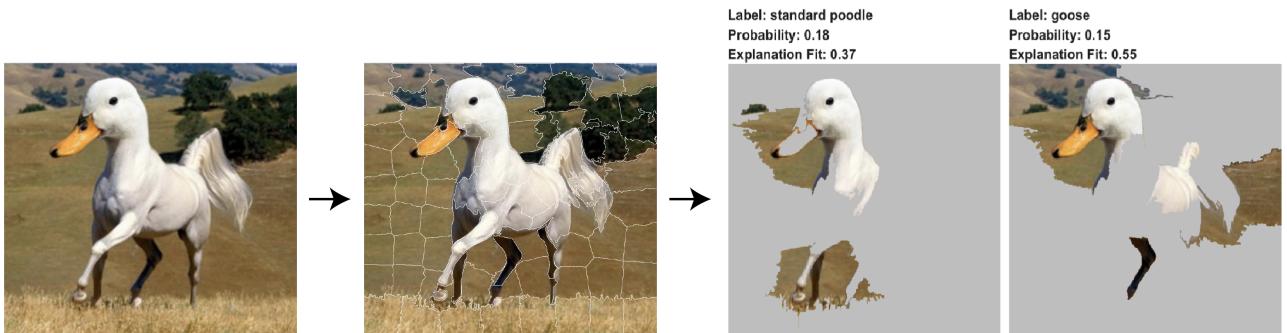
модель, которую мы получили, действительно ли релевантные признаки оказались значимыми.

Пример LIME для текстов [5]



Вновь решалась задача бинарной классификации: определить, к какой теме (христианство/атеизм) относится документ. LIME выделил отдельные слова, которые, по его «мнению», оказались наиболее важны для предсказания. Можно заметить, что слово «Posting» оказалось довольно-таки значимым, поскольку увеличило вероятность принадлежности текста к тематике атеизма на 0.15. Однако это несколько континтуитивно, поскольку «posting» может относиться одинаково вероятно к обоим темам. Мы выявили недостаток в модели, к результатам ее работы стоит относиться аккуратно.

Пример LIME для картинок [4]



Интерпретация картинок с LIME наиболее понятная и интересная. Изображение было разбито на несколько частей (супер-пиксели). И далее алгоритм выбрал наиболее важные для получения того или иного ответа. На второй картинке LIME корректно выбрал фрагменты гуся, которые модель смогла распознать. Но на первой картинке модель ошиблась и выявила пуделя.

2.2.2 Принцип работы

У нас есть модель $f(x) : \mathbb{R}^d \rightarrow \mathbb{R}$ 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) [6].

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: $x \rightarrow x'$. 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 [6].

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 объединяются в токены (например, слова – они хорошо интерпретируются) [1].

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 $g(x') : \{0, 1\}^{d'} \rightarrow \mathbb{R}$ работает с интерпретируемым представлением предсказания $x' \in \{0, 1\}^{d'}$, где обычно $d' \ll d$. Дополнительно в задаче вводится мера сложности $\Omega(g)$ искомой модели как регуляризация [4].

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 [6]:

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

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 [6].

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 [6].

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

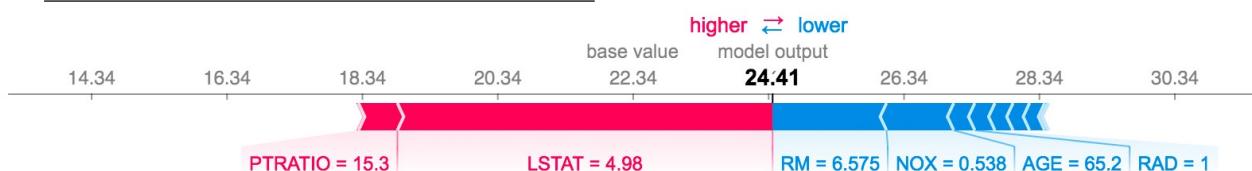
Предсказание модели формируется на основе признаков. Если мы хотим узнать влияние отдельного признака, мы можем построить предсказание модели с ним и без него и затем посмотреть, как меняется результат. Но модель может быть слишком сложной, чтобы мы могли оценить влияние предиктора по одному объекту, 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 признаков для разных объектов, мы получаем очень много предельных эффектов, что тяжело интерпретируется. Поэтому можно рассмотреть, какой в среднем эффект оказывает включение признака в модель. Тогда мы получим средневзвешенную оценку вклада отдельного предиктора в предсказание и сможем интерпретировать 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.

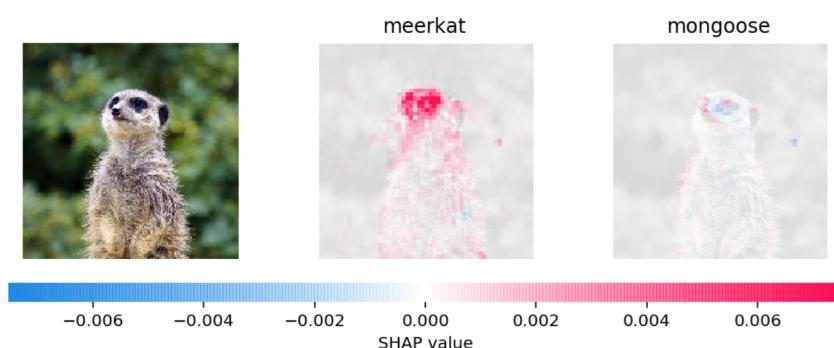
Авторы алгоритма также вносят новую идею о способе расчета вклада признаков в предсказание: данные значения можно посчитать с помощью линейной регрессии с использованием матрицы весов определенного вида [7].

Пример SHAP для табличных данных [8]



В данном примере решалась задача регрессии, предсказание модели: 24.41. Отталкиваясь от base value как от исходного значения под воздействием влияния разных признаков (показаны стрелочками), данное наблюдение получило именно такой ответ, по мнению SHAP. Приведенные предикторы оказали наибольшее воздействие на результат работы модели. В аналогичном виде представляется интерпретация предсказаний для текстов.

Пример SHAP для картинок [8]



В данном случае SHAP, в отличие от LIME, работал с более мелкими признаками – обычными пикселями. Значимость каждого признака показана с помощью цвета: чем насыщеннее цвет, тем сильнее он повлиял на результат. Красный цвет показывает положительное воздействие, синий цвет – отрицательное. Однако SHAP может работать и с супер-пикселями, аналогично LIME.

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 [1, 7]:

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

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 [1, 7].

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 [1].

2.3.3 Принцип работы

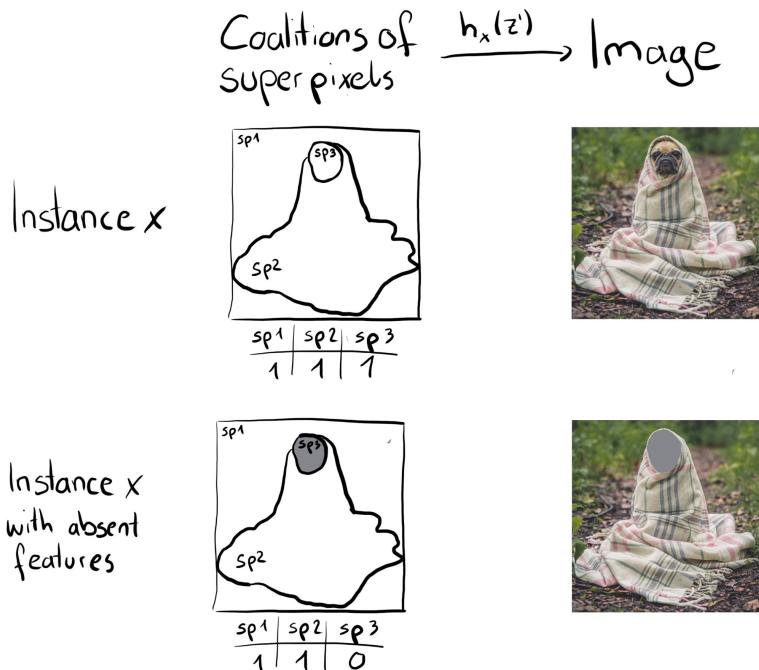
Метод SHAP несколько отличается от использования значений Шэпли напрямую для интерпретации вклада признаков – он модифицирует классический подход. Рассмотрим SHAP подробнее.

Вместо использования большого количества моделей мы обучим одну модель и далее будем пользоваться только ее предсказаниями. Тогда наша модель представима в виде

$f : \mathbb{R}^d \rightarrow \mathbb{R}$. 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 [1, 7].

Для упрощения расчетов мы не будем рассматривать разные значения предикторов, а бинаризуем их представление, обозначив за «1» наличие предиктора и за «0» его отсутствие: $x \rightarrow x'$, $x' = \{1\}^d$. Обозначим за $z' \in \{0, 1\}^d$ объект, у которого мы учтываем 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 [1, 7].

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 [1].



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. Данную формулу мы можем использовать при расчете необходимых значений [1].

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 [1]:

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

где $z_{F \setminus S}$ – исключенные из модели признаки, вместо которых 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 признаков): $y_i = f(h_x(z'_i)) = f(z_i)$, где z'_i – строка матрицы Z . Тогда решением нашей задачи является вектор [1]:

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

Чтобы коэффициенты в регрессии соответствовали искомым значениям, веса должны быть заданы диагональной матрицей W [7]:

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

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 [1].

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 [1].

Интересно, что если записать задачу в более общем виде:

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

мы получим ровно ту же задачу, что решает LIME. То есть при определенных гиперпараметрах LIME и SHAP эквивалентны друг другу [7].

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 [1, 7].

2.3.4 Реализация

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

- ◊ `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](#).

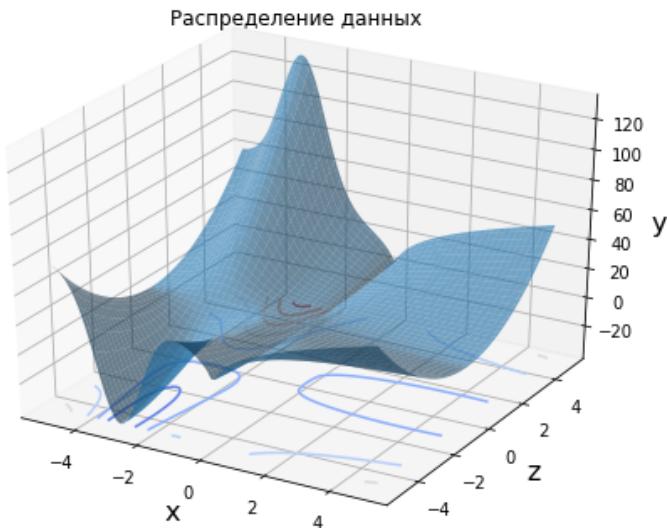
3 Пример применения

Возьмем искусственный датасет с двумя признаками и одной зависимой переменной, чтобы испытать описанные методы. Выбрано именно два признака, чтобы была возможность визуально посмотреть на набор данных и увидеть закономерности в них, а также усложнить задачу алгоритмам, используя трехмерное пространство вместо двумерного.

Зависимая переменная задается следующей формулой:

$$y = \frac{e^{-x^2-6x+5}}{70000} \cdot z - 3 \sin 0.3x + \sqrt{|x|} \cdot z^2$$

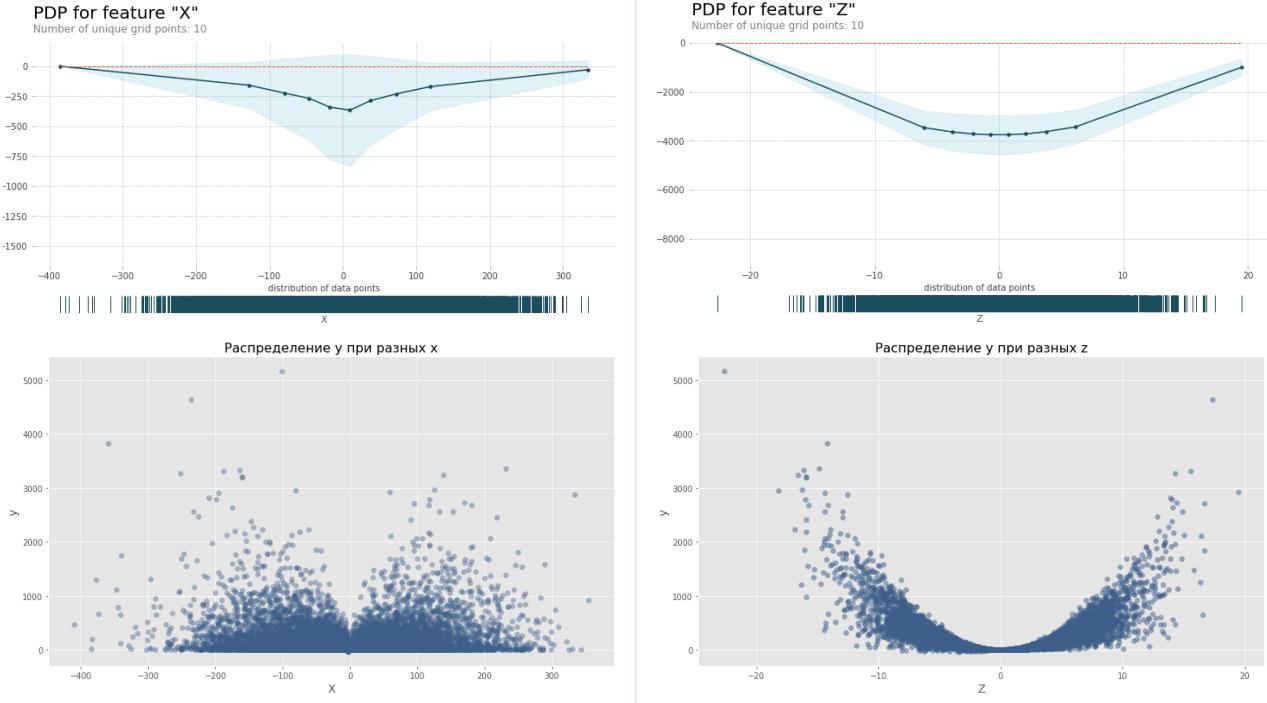
График данной функции:



Переменные x и z сгенерированы как независимые нормальные случайные величины. Для задач был выбран стандартный градиентный бустинг реализации `sklearn` (`GradientBoostingRegressor`). Гиперпараметры подобраны кросс-валидацией. Подробнее об обучении и кросс-валидации [здесь](#).

3.1 PDP

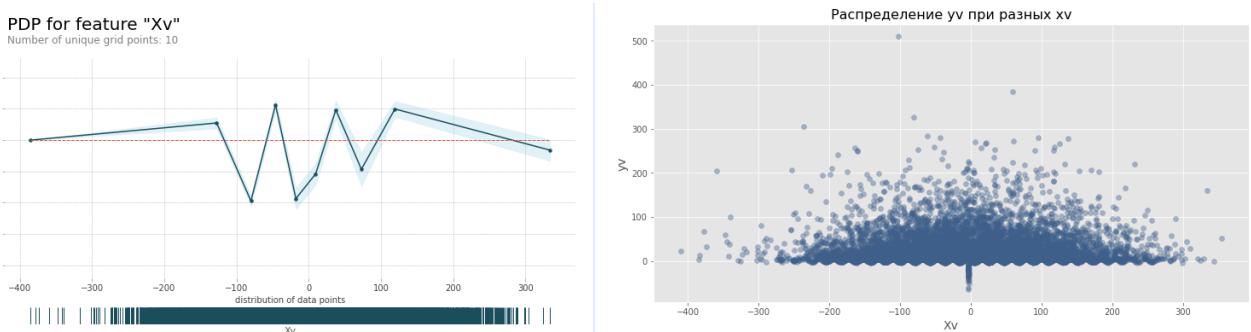
В библиотеке `pdpbox` представлены два типа графиков: для одного признака и для двух. Посмотрим на каждый:



По графикам видно, что PDP действительно улавливает верную закономерность. Для признака X : PDP показывает углубление рядом с нулевым значением, которое видно на данных. Аналогично, для признака Z : PDP показывает параболу, которая также хорошо заметна в выборке. Однако неточно отображен масштаб, эффект признака X оказался занижен, признака Z – завышен. Это видно именно по PDP – линии, показывающей эффект для среднего предсказания. Например, для Z эффект -4000 не наблюдается практически ни для одного объекта. В целом PDP верно описывает закономерность в данных без учета масштаба.

Однако так получается не со всеми функциями. Рассмотрим небольшой контрпример:

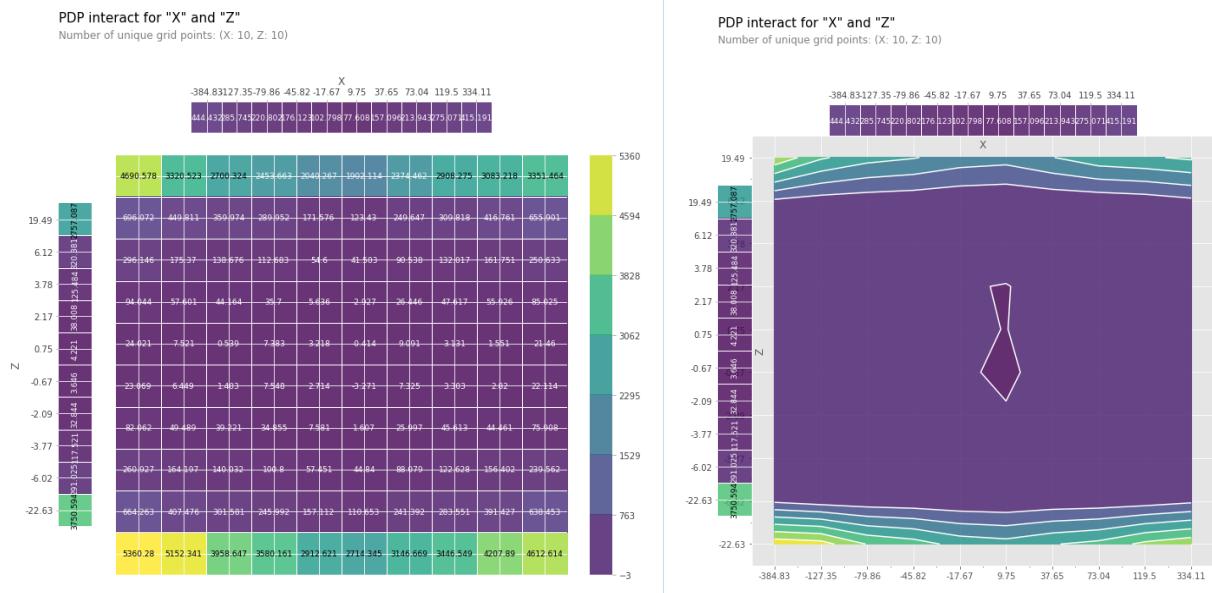
$$y = \frac{e^{-x^2-6x+5}}{70000} \cdot z - 3 \sin 0.3x + z^2$$



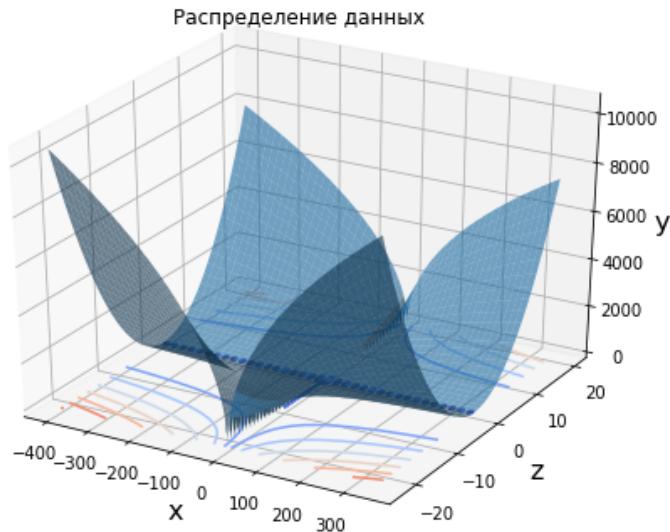
PDP показывает некоторые колебания эффекта переменной Xv в интервале от -100 до 100 . И это может быть правдой в масштабе левого графика. Такие колебания можно заметить и на правом графике – небольшие волны около нуля. Однако мы смотрим на выборку и видим более глобальную картину: чем ближе к нулю, тем больше эффект признака, хоть он и возрастает незначительно. Интуиция в данном случае конфликтует с результатом PDP – стоит рассмотреть также другие точки зрения для принятия окончательного решения.

тельных решений.

Вернемся к исходному примеру и взглянем на оба признака одновременно:



Получилась не очень показательная картинка. По ней кажется, что признак X не оказывает особого влияния на результат. Данный график показывает не эффект признаков, а значение зависимой переменной. Поэтому вновь посмотрим на данные (возьмем масштаб побольше), чтобы сравнить PDP и истинную зависимость:

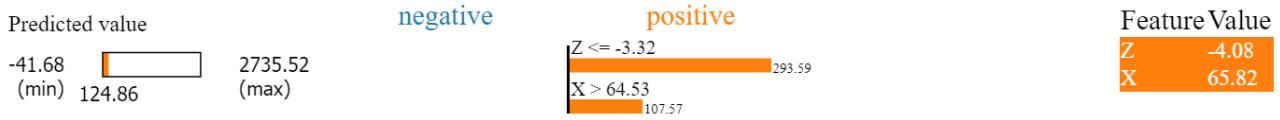


Получилось в самом деле похоже на истинную зависимость – видно, по линиям уровня на 3D-графике.

PDP оказался довольно-таки хороши: он показал извлеченные из данных и модели зависимости, что полезно при анализе выборки и результатов работы модели. Однако стоит быть аккуратными – контрпример показал, что не всегда PDP отображает то, что мы хотели бы получить.

3.2 LIME

Перейдем к следующему методу. Выберем объект из выборки для анализа:



Стоит отметить, что данная иллюстрация показывает, как сформировалось предсказание под влиянием признаков, если исходно отталкиваться от некоторого среднего значения (среднего предсказания).

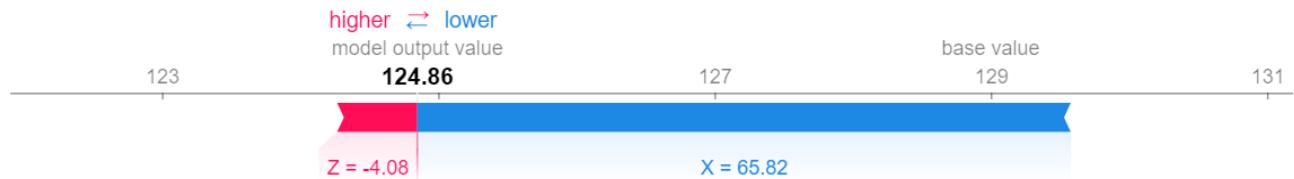
LIME отмечает, что оба признака внесли положительный вклад в результат работы модели: значение предсказания выросло примерно на 400 единиц. При этом важно, что признак Z принял значение меньше -3.32 , а признак X – значение больше 64.53 . Данные пороги как раз объясняются дискретизацией признаков в алгоритме LIME.

Вернемся к исходному виду зависимости, чтобы сравнить полученный результат с истинной зависимостью. В самом деле, при каждом $X > 64.53$ график функции $y(z)$ выглядит как парабола с ветвями вверх с вершиной в $Z = 0$, причем чем больше X , тем быстрее возрастают ветви параболы. Соответственно, при любых значениях Z , отличных от нуля функция возрастает. Значит, LIME представил правдивый результат: оба признака действительно вносят положительный вклад в значение зависимой переменной, что верно выявила наша модель.

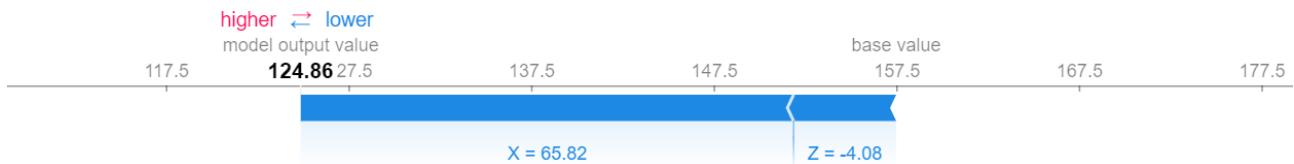
3.3 SHAP

Последний рассмотренный метод – SHAP. Ранее мы детально описали KernelSHAP, однако также упомянули TreeSHAP, который лучше подходит для моделей, в основе которых лежат решающие деревья. Оставим тот же самый объект, что мы рассматривали в LIME, и используем оба типа SHAP, чтобы сравнить качество и скорость.

KernelSHAP



TreeSHAP



Мнения расходятся: KernelSHAP считает, что Z положительно влияет на предсказание, TreeSHAP – наоборот. Но оба сходятся на том, что X оказывает большое отрицательное влияние, чем они отличаются от LIME. Однако стоит отметить, что LIME и SHAP по-разному рассчитывают некоторое исходное значение (base value), от которого отталкиваются при объяснении.

Тем не менее, результаты SHAP оказываются менее интуитивными. Значение $Z = -4.08$ должно увеличивать значение y , так как это лежит далеко от вершины параболы,

причем за значением 25% квартиля. Аналогично, X – чем больше X , тем сильнее наклон ветвей параболы – в нашем случае X лежит на границе 75% квартиля, что уже почти выходит за пределы основной выборки. Все этом кажется говорит о том, что оба признака должны вносить положительный вклад в предсказание. Однако SHAP показывает совсем не такие результаты.

Для данной задачи рассматривалась небольшая выборка. Для нее KernelSHAP оказался быстрее TreeSHAP. Однако на большой выборке TreeSHAP выходит в лидеры (см. след. главу).

Далее рассмотрим только KernelSHAP, так как он предоставляет более точные результаты. Взглянем также на другие графики, которые предоставляет библиотека:

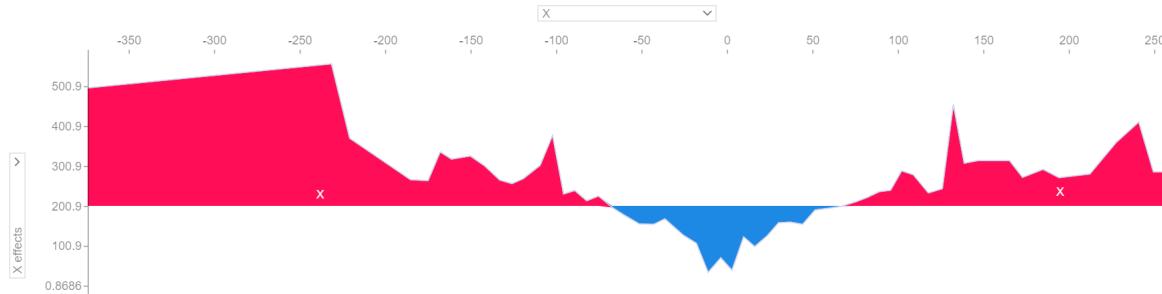


Рис. 1: Эффект признака X в зависимости от его значения

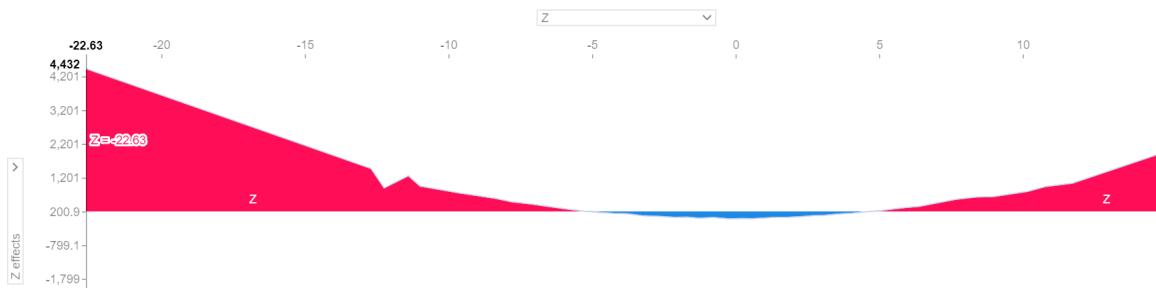


Рис. 2: Эффект признака Z в зависимости от его значения

Оба графика являются аналогами PDP, которые мы строили ранее. Они отображают похожую картину, в случае признака X даже более детализированную.

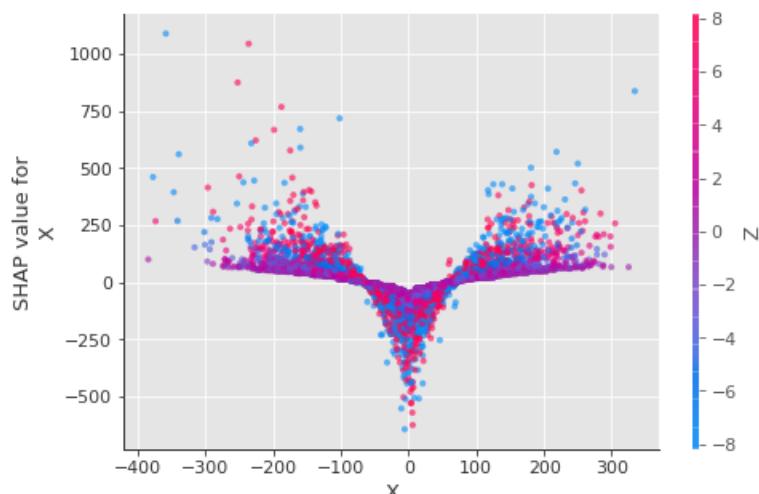


Рис. 3: Зависимость SHAP value признака X от его значения

Еще одна альтернатива PDP от SHAP. Данный график показывает распределение признака X и его SHAP value для разных объектов, а также его связь с признаком Z . По

графику можно сделать следующий вывод: чем ближе значение X к нулю, тем больше по модулю отрицательный вклад признака (его SHAP value); чем больше по модулю значение X , тем больше его положительный вклад. И есть две точки перегиба, в который вклад практически незаметен.

Цвет точек на графике показывает взаимосвязь с признаком Z – его шкала нарисована справа. Видно, что для низких и высоких значений Z сохраняется одинаковая зависимость y от X . Однако при Z , близком к нулю, значимость X снижается и колеблется также около нуля. Данную зависимость действительно можно увидеть в исходных данных, но 3D-графику истинной функции y .

Получили, что SHAP не всегда дает точные результаты, к методу стоит относиться аккуратно. Тем не менее он предоставляет широкий функционал по анализу выборки, выполняет свои основные функции интерпретации и вылавливает зависимости, которые обнаружила модель в данных, хоть и с некоторой погрешностью.

Таким образом, все три метода решают поставленную задачу – интерпретируют результат работы модели. Однако также все содержат определенную погрешность, из-за чего к объяснениям, полученным с помощью PDP, LIME и SHAP, стоит относиться осторожно, рассматривая разные точки зрения и обращая внимание на сами данные, смысл каждой переменной.

4 Анализ данных и интерпретация

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

4.1 Данные

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

- ◊ Удалены переменные company (много пропусков), agent (ID туристических агентств – 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 $adr > 0$

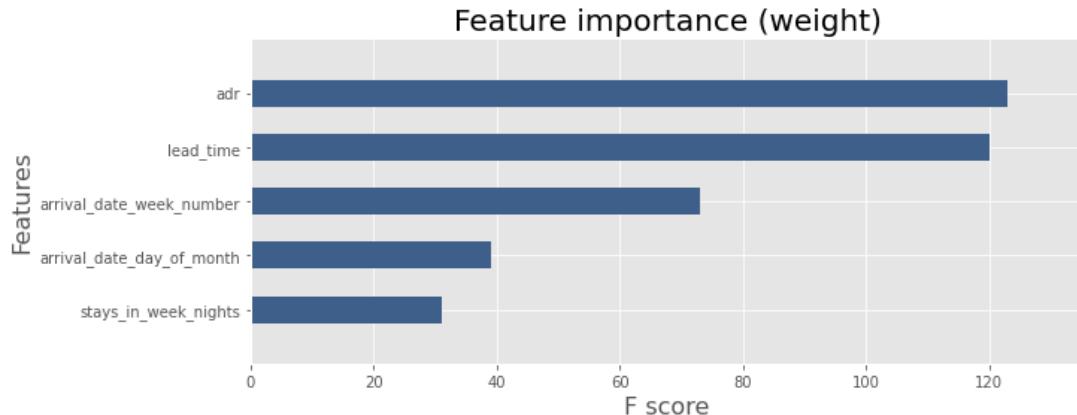
4.2 Модель

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](#)

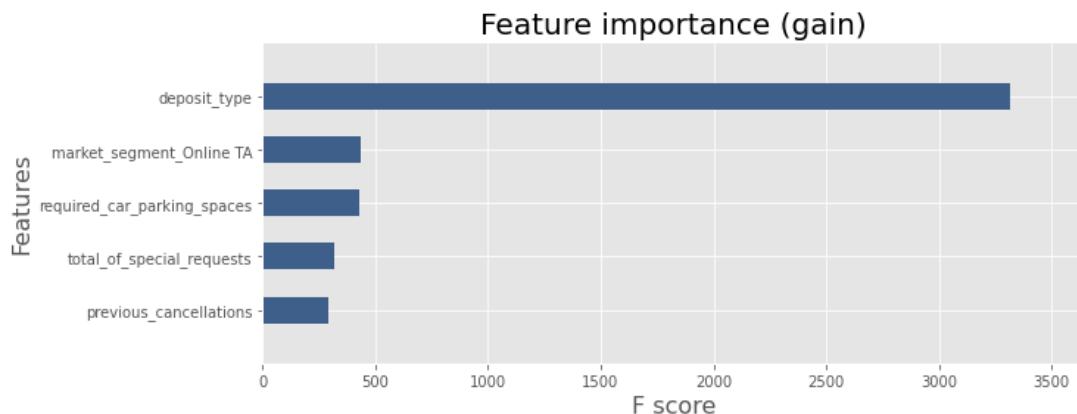
4.3 Интерпретация результатов

Модель достигла неплохого качества в задаче классификации: ROC-AUC=0.905, accuracy=0.824. Попробуем интерпретировать ее результаты.

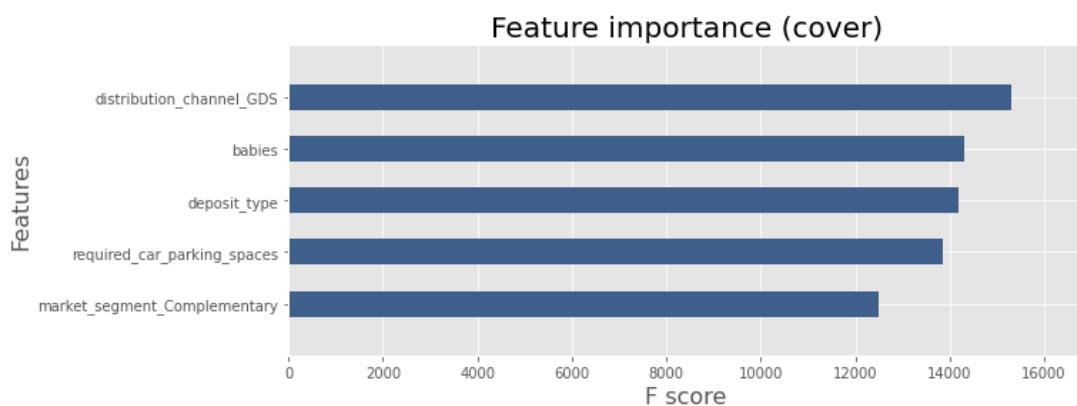
Нулевой способ это встроенный метод XGBoost, показывающий важность признаков при предсказании. Посмотрим на первые 5 самых важных признаков. Данная величина может быть рассчитана тремя способами: «weight», «gain», «cover». Первый показывает, сколько раз признак появляется в дереве:



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



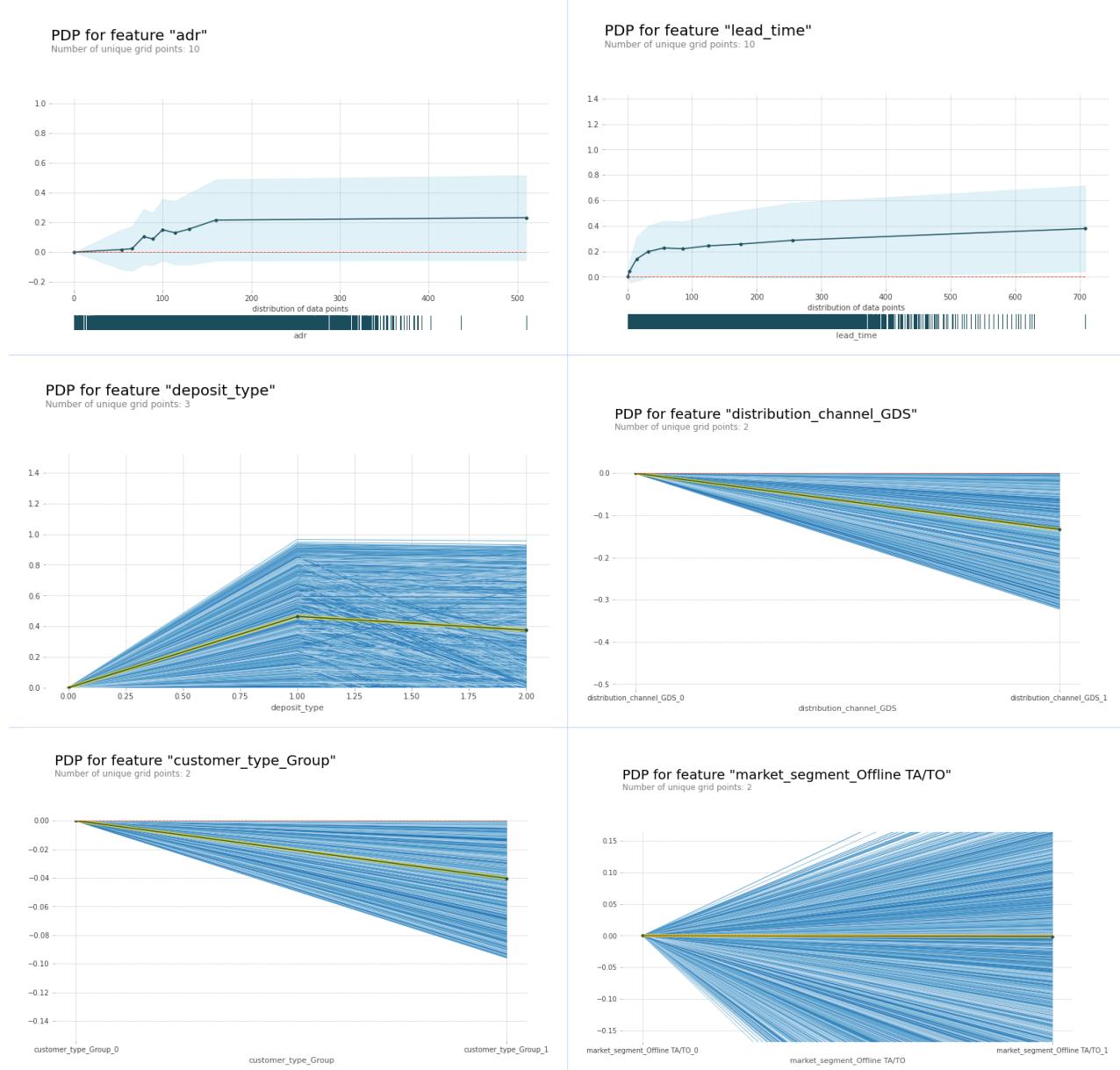
И последний – какое количество объектов выборки задействовало узлы с заданным признаком:



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 выходят 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), последний из gain (customer_type_group) и последний из cover (market_segment_Offline TA/TO).

Построим для них PDP.



Первый признак является непрерывным, второй – дискретным, третий – порядковым, остальные – бинарными. Для первых двух видно распределение значений признака (штриховка под графиком) и коридор, показывающий как данный признак влиял на разные объекты в выборке. Для остальных: выборка была кластеризована, и для каждого кластера из 1000 была построена усредненная по подвыборке кривая.

По графикам видно, что:

- ◊ признак adr оказался важным с точки зрения PDP – для больших значений (> 50) он вносит положительный вклад в прогноз, увеличивая вероятность отмены брони. Но видно, что для некоторых объектов в выборке он оказывал также и отрицательное влияние – коридор задевает область отрицательных значений.

Данный признак показывает, сколько в среднем гость тратит на проживание и связанные с ним расходы. Исходя из графика, можно сделать вывод, что чем больше предстоящие расходы, тем выше вероятность отмены брони – звучит логично, клиент вероятнее отменит бронь, если для него эта поездка окажется слишком дорогой. Причем вклад данного признака стабилизируется с ростом затрат и составляет $+0.2$ к вероятности отмены в среднем

- ◊ Аналогичный график у lead time (время от открытия брони до приезда).

Здесь ситуация также интуитивно понятна: если гость очень заранее забронировал номер, то за время до приезда его планы могут поменяться. Поэтому клиент вероятнее отменит бронь в данном случае – $+0.2-0.4$ к вероятности отмены

- ◊ deposit type также оказался важным признаком: в среднем он оказывает положительное воздействие на предсказание, которое доходит вплоть до полного влияния в виде $+0.9$ к вероятности. Ни для одного кластера признак не оказывает отрицательное воздействие, однако возможно для отдельных объектов это неверно – важно аккуратно интерпретировать результаты.

Здесь результат несколько континуативен. Если у клиента есть полный предоплаченный депозит, который не возвращается, то вероятность отмены брони стремится к единице, что нелогично – внесенный залог должен мотивировать гостей приезжать. Далее мы видим, что для большинства кластеров при переходе к возвращаемому частичному депозиту вероятность практически не меняется и остается около единицы – это уже более логично, однако также спорно: в случае отмены придется тратить время на бюрократию, связанную с возвратом средств. Но для части кластеров при переходе к возвращаемому депозиту вероятность отмены даже падает, что вызывает сомнения в корректности использования данного признака. Возможно, стоило выбрать другую форму для данного признака: сделать его бинарным, а не порядковым – то есть, возможно, данная ситуация сложилась из-за неправильного представления категорий депозита.

- ◊ distribution channel GDS отрицательно влияет на предсказание, причем довольно-таки сильно: при переходе от 0 к 1 вероятность отмены брони снижается в среднем на 0.1, максимально по кластерам на 0.3

Данный признак показывает то, что гость воспользовался глобальной системой бронирования, то есть вероятно самостоятельно организовал себе поездку. Снижение вероятности в данном случае не очень логично: человеку проще отменить поездку, когда он ее организовал сам. Также в таком случае выше шанс возникновения проблем в поездке (по сравнению с организацией, предоставляемой туристическими агентствами), из-за чего бронь также может быть отменена

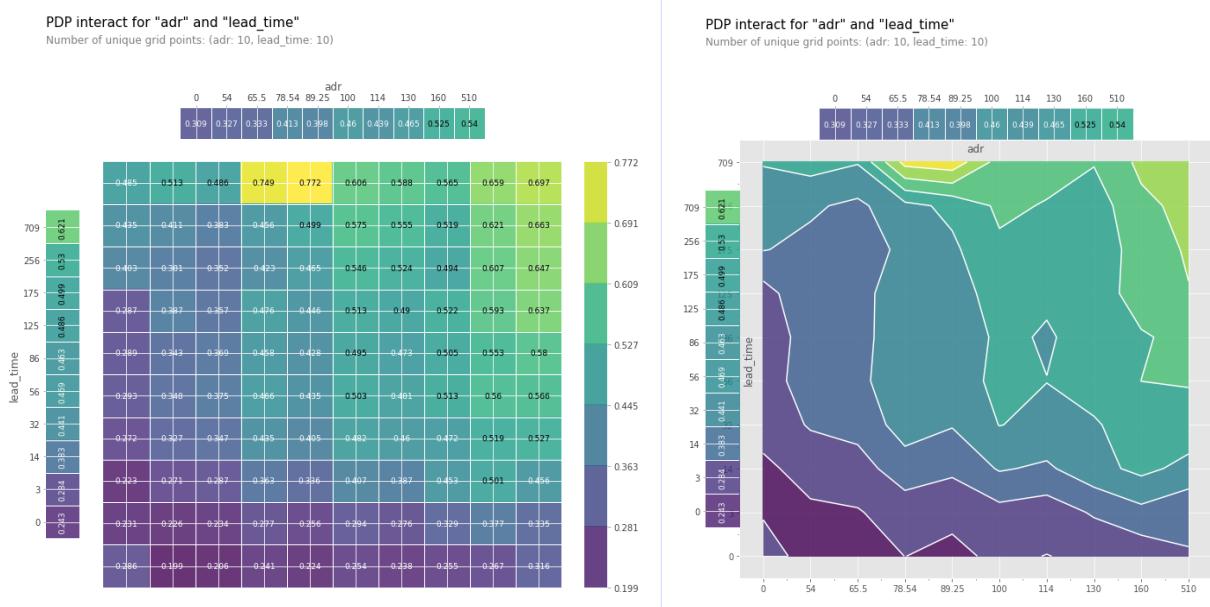
- ◊ у customer type group похожий график, однако влияние существенно ниже. Вероятность снижается на 0.04 в среднем, максимально по кластерам всего лишь на 0.1

График показывает, что при организации групповой поездки ниже шанс, что она отменится. Исходно не очень понятно: если бронь на группу, то она вероятнее отме-

нится, потому что сложно организовать путешествие на целую группу, или она вероятнее не отменится из-за, например, обязательства каждого перед другими (чтобы неставить людей из своей группы в неудобное положение). То есть данный график приносит некоторое дополнение к нашим данным. Теперь мы знаем, что если поездка организуется для группы, то бронь скорее не будет отменена – учитываем данное дополнение с осторожностью, учитывая неточность метода

- ◊ признак market segment offline TA/TO оказался незначительным, в среднем он вносит нулевой вклад в предсказание. Однако по кластерам видно, что разброс существенный. То есть данный признак оказывает влияние на каждый отдельный объект – PDP в виде усредненной кривой не совсем корректный выбор для интерпретации в данном случае, так как он не учитывает разброс

Построим также графики взаимодействия признаков (PDP для двух признаков)¹. Посмотрим на совместное влияние признаков, которые XGBoost по критерию weight считал важными: adr и lead_time.

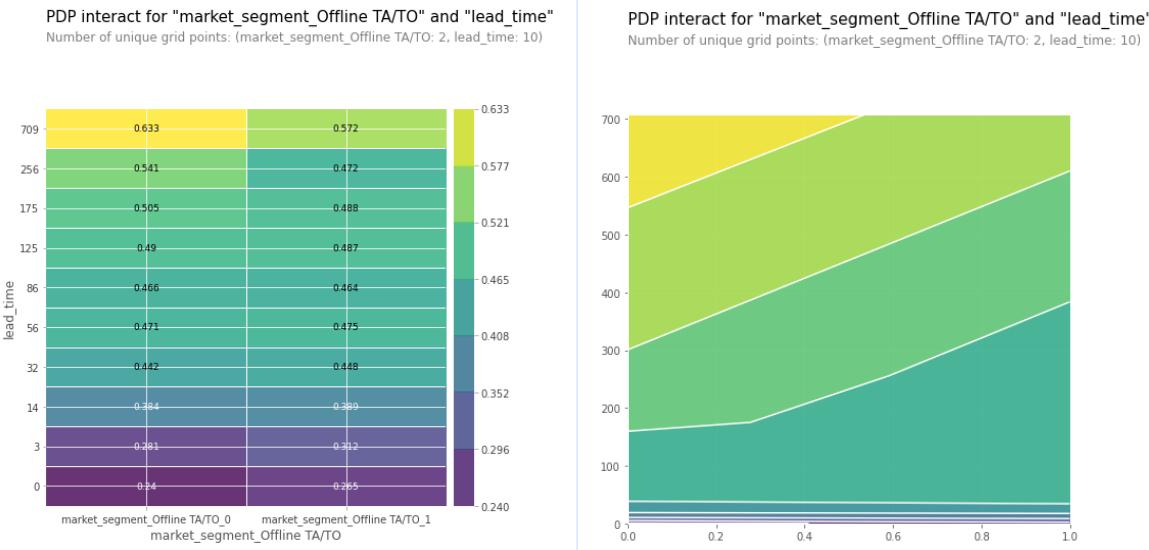


Данный график показывает не влияние предикторов, а непосредственно предсказание. Действительно, взаимодействие данных признаков меняет картину (то есть предсказание не линейно зависит от них). Разброс значений наблюдается с 0.199 до 0.772, разница в 0.573. Особо выделяется пик, который сложился именно из взаимодействия – желтый островок со значениями 0.7+

В целом мы видим, что при росте времени до приезда и затрат вероятность растет. И выделяется особый случай, когда затраты не очень большие, а время до приезда велико – в таком случае вероятность отмены брони максимальна.

¹При построении второго типа графика (contour), который показывает линии уровня может возникнуть ошибка. Это связано с несоответствием версии библиотеки matplotlib

Также интересно узнать, станет ли market segment offline TA/TO более значимым в сочетании с другим, например, с lead time.

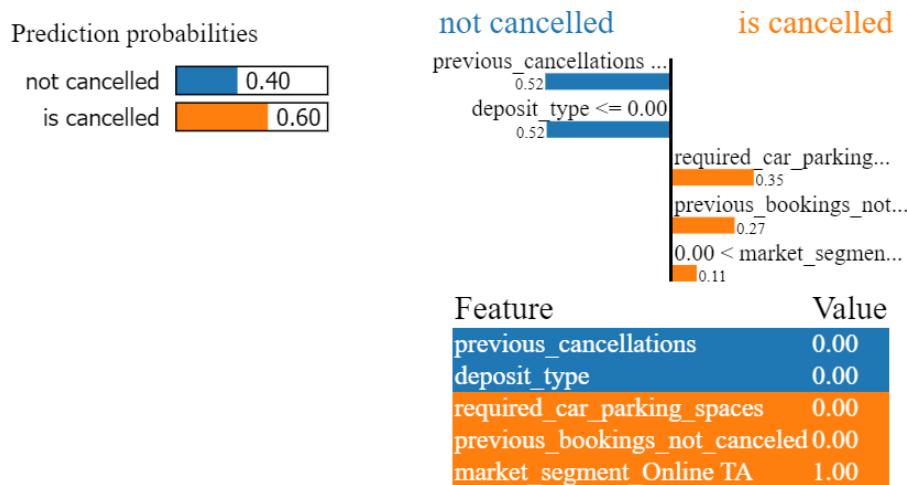


Рассмотрение двух признаков привело к появлению влияния market segment offline TA/TO – при больших значениях lead time он становится более значимым, его предельный эффект при $\text{lead time} = 709$ в среднем составляет $0.572 - 0.633 = -0.06$, что по модулю больше нуля. При малых значениях lead time он перестает влиять на предсказание. Это отличается от результатов, полученных ранее.

Получили добавку к интерпретации market segment offline TA/TO – если поездка организована туроператором, то она менее вероятно отменится, что интуитивно понятно.

С помощью PDP мы убедились в корректности восприятия моделью некоторых признаков. Для других – в континуитивности результатов, а также нашли возможность для пересмотра формата признаков с целью улучшения модели. Стоит отметить, что в PDP есть очевидный недостаток – если признак равен 0, то его влияние также равно 0, что не всегда правда. Важно обращать на это внимание при интерпретации.

Перейдем к следующему методу – LIME. Рассмотрим конкретную поездку (первая из тренировочной выборки), чтобы увидеть, какие признаки оказались наиболее важными для предсказания отмены брони. Истинное значение для данного объекта: 0, то есть бронь не была отменена.



Модель предсказала, что бронь будет отменена, и LIME с некоторой погрешностью показывает нам, что на это повлияло. Мы видим, что отсутствие отмен ранее снизило

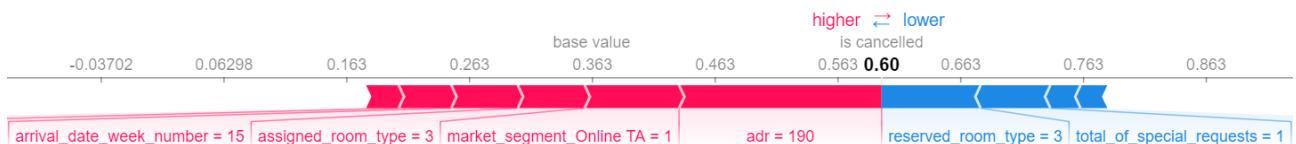
вероятность текущей отмены, что корректно. Снова столкнулись с признаком deposit type – отсутствие депозита также снизило вероятность отмены – в целом не противоречит ничему, однако отсутствие депозита позволяет без каких-либо потерь отменить бронь, то есть все же скорее увеличивает вероятность отмены с точки зрения интуиции. Два данных признака сильнее всего повлияли на снижение вероятности отмены: -0.52 от каждого.

Клиент не запросил парковочное место, что увеличило вероятность отмены причем довольно-таки сильно (+0.35) – неочевидное влияние. Отсутствие не отмененных ранее бронирований также повысило вероятность отмены, что звучит логично. Судя по всему это новый для отеля клиент. Он заказывал номер через онлайн-туроператора, и это повысило вероятность отмены на 0.11 – интуитивно понятно, по интернету проще отменить бронь.

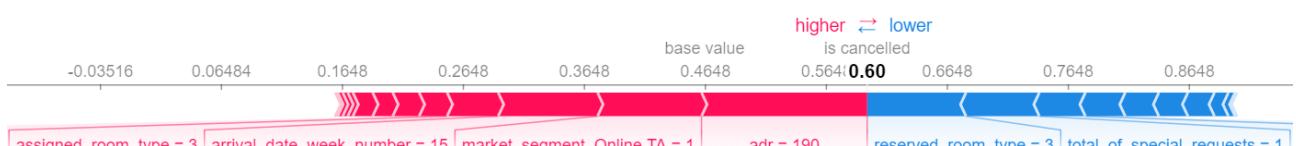
LIME вывел первые пять признаков, которые сильнее всего повлияли на результат. Остальные признаки довели предсказание до того, что получила модель. Она ошиблась с ответом, и мы по полученному объяснению можем понять почему. Вопросы вызывают признаки deposit type и required car parking spaces – они оказывают неочевидное воздействие на предсказание, возможно, именно из-за них модель ошиблась. Также стоит обратить внимание на силу влияния признаков: не очень понятно, почему предикторы, снижающие вероятность отмены имеют такое большое значение. То же можно сказать и про required car parking spaces.

Таким образом, LIME позволяет выявить недостатки в понимании смысла признаков у модели. Благодаря этому мы можем предотвратить использование некорректной модели или попытаться улучшить ее.

Посмотрим на то же самое предсказание с помощью последнего рассматриваемого метода – SHAP. Ранее мы рассматривали принцип работы KernelSHAP и выявили его сходство с LIME. Но по умолчанию они используют разные линейные модели и веса для расчетов значений признаков. Поэтому признаки, которые они обозначают за наиболее важные, могут отличаться. Для ускорения расчетов выборка была кластеризована.



Все признаки, кроме market segment online TA, не совпадают с теми, что выявил LIME. Так как мы используем градиентный бустинг, имеет смысл попробовать TreeSHAP, так как он должен работать быстрее для нашей модели (но с меньшей точностью) – он действительно рассчитал значения гораздо быстрее в сравнении с KernelSHAP:



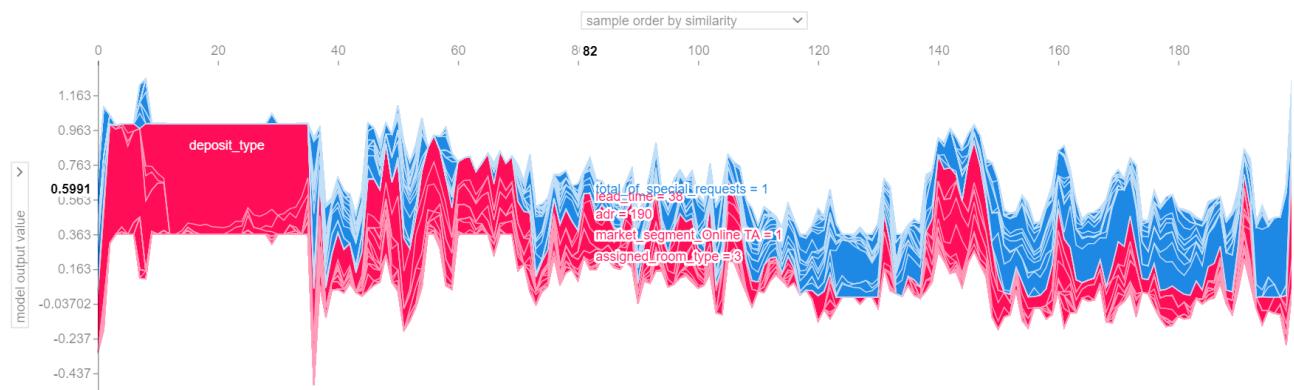
Результат практически не поменялся: изменилась сила влияния некоторых признаков, assigned room type и arrival date week number поменялись местами – TreeSHAP довольно-таки качественная замена KernelSHAP для моделей, основанных на деревьях. Остальные графики также практически не поменялись.

По обоим графикам можно сделать примерно одинаковые выводы. С точки зрения SHAP тип зарезервированной комнаты и дополнительные требования от клиента уменьшили вероятность того, что он отменит бронь. Тип комнаты представлен в зашифрованном виде, поэтому мы не можем сказать, корректно ли модель учитывает признак. Однако

наличие дополнительных требований интерпретируется логично, так как предоставление необходимых клиенту услуг безусловно вызывает положительный отклик с его стороны.

В то же время такие признаки, как расходы, связанные с проживанием, бронь через онлайн-туроператора, неделя прибытия и предоставляемый тип комнаты (который будет предоставлен клиенту по факту – может отличаться по техническим причинам или по желанию клиента). Первые два обсуждались ранее и интуитивно понятны. Последний также можно опустить, поскольку мы не знаем, как расшифровать признак. Стоит отметить, что забронированный и назначенный тип комнаты совпали – требование клиента было выполнено. Соответственно, это должно снижать вероятность отмены. Но модель этого не учитывает, так как не знает взаимосвязи между признаками. В частности поэтому предикторы, связанные с типами комнаты, могли некорректно влиять на предсказание.

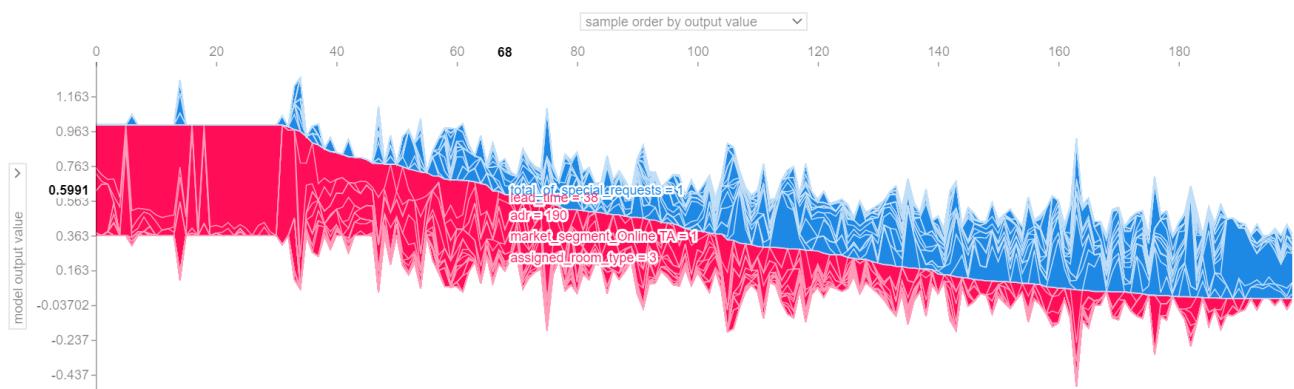
Мы рассмотрели отдельное предсказание. Попробуем также построить другие типы графиков, чтобы увидеть более общую картину – возьмем подвыборку из 200 наблюдений.



Forceplot для подвыборки: показывает, как получено предсказание для разных объектов

Данный график соединяет в себе индивидуальные графики для объектов выборки. По оси абсцисс отложены объекты из выборки, то есть мы можем одновременно смотреть на то, как сформировалось предсказание для разных наблюдений. Например, на графике отмечен объект, который мы уже рассматривали ранее, а также наиболее важные для его предсказания признаки.

Интерактивный интерфейс позволяет посмотреть различные комбинации. На графике выше показана зависимость предсказания модели от объекта, причем объекты были упорядочены по схожести между собой. Отсортируем объекты по значению предсказания:



Forceplot: объекты отсортированы по значению предсказания

Наш объект переместился к началу координат. Также можно заметить наблюдения, для которых модель, «по мнению SHAP», сделала уверенное предсказание – вероятность

примерно равна 1.



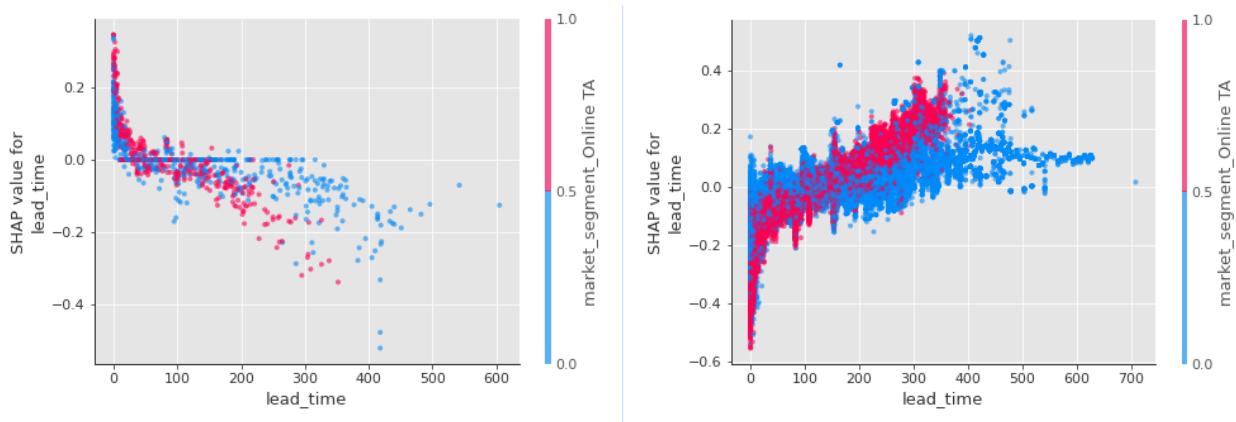
Forceplot: объекты отсортированы по возрастанию признака lead_time

Данный график является аналогом PDP, построенным по методу SHAP – он показывает, как в среднем меняется предсказание в зависимости от значения признака lead time. Но поскольку влияние предикторов в PDP и SHAP рассчитывается разными способами, графики оказались не слишком похожи.



Forceplot: зависимость эффектов признака adr от признака lead_time

Также можно посмотреть, как меняется эффект от выбранного признака при разных значениях другого – некоторый аналог PDP для двух признаков. Например, при adr = 81 lead time увеличивает вероятность отмены



KernelSHAP

TreeSHAP

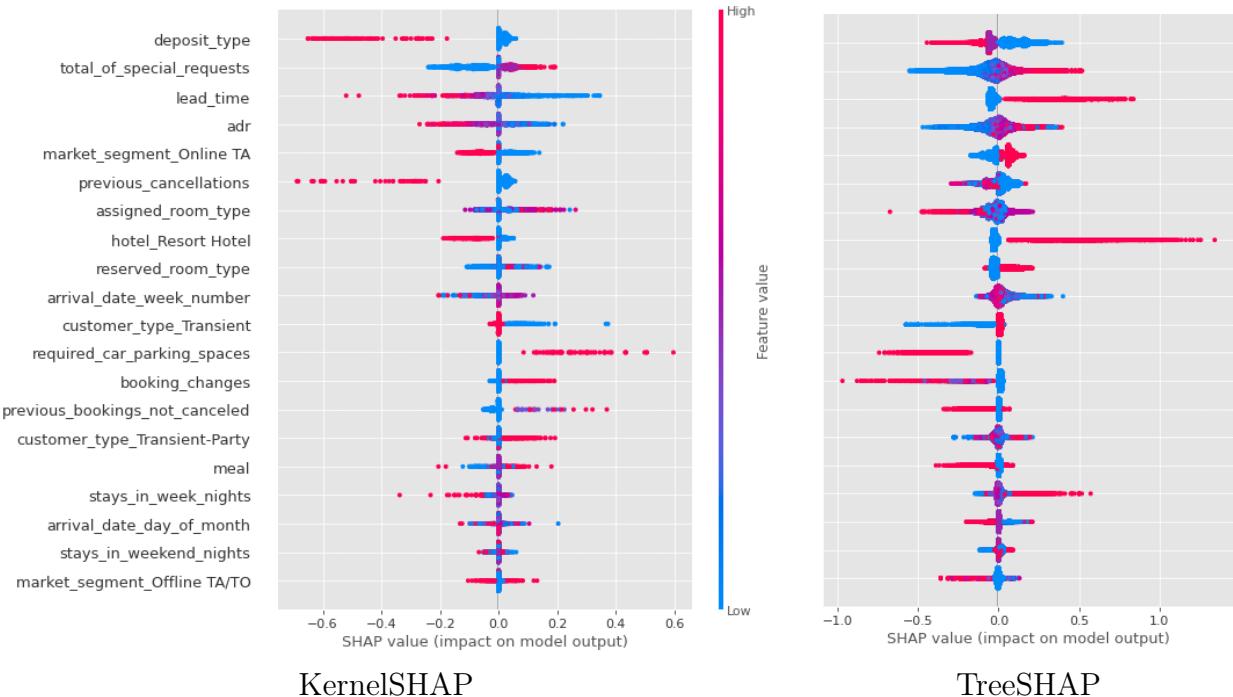
Dependence plot для признака: зависимость SHAP value признака от его значения

Данный тип графика значительно отличается для выбранных методов. Для KernelSHAP график был построен по подвыборке из 200 объектов, для TreeSHAP по всей выборке.

По левому графику можно сделать вывод, что чем меньше значение lead time, тем более значим признак для предсказания. Интересно, что, например, при lead time = 100 SHAP value, то есть вклад признака в предсказание, может быть как положительным, так

и отрицательным. Цвет точек на графике показывает значение другого признака, в данном случае market segment online TA – он выбирается автоматически, как признак с которым выбранный взаимодействует сильнее всего. Зависимость, показанная на графике, может быть разной для разных значений признака market segment online TA – для красных точек зависимость более сильная, чем для голубых.

Для правого графика получаются противоположные выводы, так как зависимость стала ровно обратной. Данная ситуация может быть связана с погрешностью метода TreeSHAP.



Summary plot для всех признаков и объектов: показывает выборочное распределение признаков, их SHAP values

Исходя из названия графика: он показывает некоторую сводку по предикторам и их важности для модели. По вертикали расположены разные признаки, по горизонтали их SHAP value, цвет точек определяет значение признака. Графики для разных методов SHAP отличаются.

Некоторые выводы:

- ◊ влияние total special requests равномерно распределяется: при высоких значениях – большой отрицательный вклад, при низких значениях – большой положительный вклад. Но для многих объектов он оказался незначительным
- ◊ для KernelSHAP: required car parking spaces не вносит вклад при маленьких значениях (0), и вносит большой положительный вклад в предсказание при больших значениях. Для TreeSHAP: наоборот

Сложилась странная ситуация, в которой два метода SHAP противоречат друг другу. Это может быть связано с тем, что для KernelSHAP мы использовали подвыборку вместо всей выборки, с погрешностью TreeSHAP при расчете значений Шэпли и т.д. Это еще раз подтверждает мысль о том, что необходимо с осторожностью относиться к данным методам интерпретации

SHAP дает более детальную картину. Мы можем посмотреть как на отдельное предсказание, так и на глобальное влияние признака. Он сочетает в себе достоинства PDP и LIME. Однако к его результатам стоит относиться аккуратно, так же, как это было с другими методами. При интерпретации стоит рассматривать ситуацию с разных сторон, строить больше графиков, чтобы убедиться в некоторой зависимости.

5 Заключение

Мы рассмотрели три метода интерпретации моделей машинного обучения: PDP, LIME, SHAP. Они имеют строгое математическое обоснование: простые, но красивые идеи, которые нашли свое применение в машинном обучении. Важным общим свойством методов является использование графиков и изображений – наиболее понятная, хорошо интерпретируемая человеком форма информации.

Существует огромное количество других методов, помимо приведенных, которые также хорошо интерпретируют работу модели. В последнее время данная область развивается особенно активно, так как распространение использования машинного обучения вызвало рост спроса на объяснение и интерпретацию результатов разных техник.

Описанные методы позволяют интерпретировать результат работы любой модели. Это полезное средство для самых разных исследований. Как мы уже заметили, они имеют свои недостатки, не всегда точно показывают влияние признаков на предсказание. Тем не менее они выполняют основную задачу – представляют результат работы модели в понятном человеку виде.

Список литературы

- [1] Interpretable Machine Learning | Christoph Molnar | Christoph Molnar | 2020 | ch. 1, 2, 5
- [2] PDPbox | Li Jiangchun (SauceCat) | GitHub
- [3] Greedy Function Approximation: a Gradient Boosting Machine | Jerome H. Friedman | 1999 | p. 1219-1220
- [4] Explanatory Model Analysis | Przemyslaw Biecek, Tomasz Burzykowski | 2020 | ch. 10
- [5] LIME | Marco Tulio Ribeiro (marcotcr) | GitHub
- [6] “Why Should I Trust You?” Explaining the Predictions of Any Classifier | Marco Tulio Ribeiro, Sameer Singh, Carlos Guestrin | 2016 | p. 3-6
- [7] A Unified Approach to Interpreting Model Predictions | Scott M. Lundberg, Su-In Lee | 2017 | p. 3-7
- [8] SHAP | Scott Lundberg (slundberg) | GitHub
- [9] Hotel booking demand | Kaggle
- [10] PDPbox tutorial
- [11] LIME tutorial
- [12] SHAP tutorial
- [13] Interpretable Machine Learning | Parul Pandey | Medium | 2019
- [14] Consistent Individualized Feature Attribution for Tree Ensembles | Scott M. Lundberg, Gabriel G. Erion, and Su-In Lee | 2019 | p. 1-4