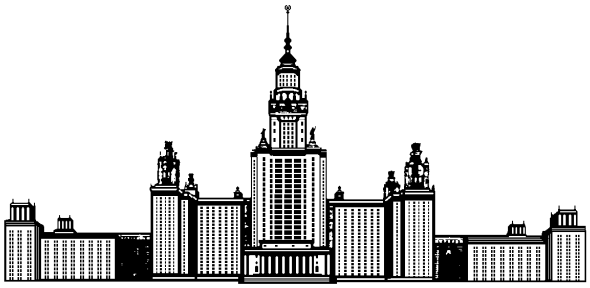


Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра математических методов прогнозирования

Кузнецов Михаил Константинович

Оценка важности признаков

Feature importance estimation

КУРСОВАЯ РАБОТА

Научный руководитель:
д.ф.-м.н., профессор
А. Г. Дьяконов

Москва, 2021

Содержание

§1. Введение

Как показывает практика, для решения сложных задач необходимы сложные модели. Иногда нам важно не только как хорошо решена задача с точки зрения качества, но и умение объяснить полученные ответы. Например, в следующих ситуациях «хорошие» объяснения играют большую роль: вынос приговора в суде, задача скоринга в банках, назначение лечения врачом. Для данных примеров уместно применение *локальных методов* определения оценки важности признаков [1], [8], [12]–[14], [16], [18]: для конкретного объекта находится насколько один признак дает больший вклад в предсказание, в отличие от другого. Простое обобщение локальных методов на все данные, например усреднение, часто приводит к «плохим» результатам (см. раздел 5.1.2, Lime). *Глобальные методы* [2], [4]–[7], [17], [19], напротив, стараются найти важность для всех данных. Такая информация бывает полезна в диагностировании систем. Представим, что мы анализируем важности признаков на протяжении какого-то времени. Большой скачок важности признака может стать сигналом об изменениях в данных. В случае поступления некорректных новых данных, время, когда произошел скачок, поможет найти причины сбоя.

§2. Обзор литературы

Существует три наиболее известные категории методов оценки *важности признаков* (feature importance): filter, embedded, wrapper.

Фильтр-методы (filter) не учитывают модель, например, коэффициенты корреляции, взаимная информация. Работают достаточно быстро.

Встроенные методы (embedded) используют внутреннее представление модели. Примерами могут служить веса модели, information gain в деревьях. Существенным недостатком является ограниченность их применения, выигрышем — более конкретное представление о степени взаимодействия признаков.

Оберточные методы (wrapper) — наиболее общий способ определения важности, так как он зависит не от устройства модели (model-agnostic), а только от ее ответов. Shapley values находят значимость исходя из индивидуального вклада признака, входящего в подмножество исходных.

Смешанные методы (mixed) — смесь вышеперечисленных. В основном, нейросетевые.

§3. Постановка задачи

Пусть $(\mathbf{X}, \mathbf{Y}) = (\mathbf{X}_1, \mathbf{X}_2, ..., \mathbf{X}_p, \mathbf{Y})$ — случайный вектор, $(x, y) = (x_1, x_2, ..., x_p, y)$ — его реализация, $\text{IndP} = \{1, 2, ..., p\}$ — индексы переменных, 2^{IndP} — множество всех подмножеств IndP. Совокупность упорядоченных x_i формирует эмпирический аналог признака — X_i, а упорядоченных y — Y. Выборка обозначается, как $(x^{(i)}, y^{(i)})_{i=1}^n$.Множество значений переменной Z равно $rng(Z)$. $\mathcal{X} = rng(X_1) \times rng(X_2) \times \ldots \times rng(X_p)$, $\mathcal{Y} = rng(Y)$. Допустим мы обучаем модель $\mathbf{a}$. Тройка $(\mathbf{X}, \mathbf{Y}, \mathbf{a})$ формирует конкретную ситуацию. Обозначим за $\mathcal{S}$ набор из всевозможных таких троек.

Тогда задача в общем случае выглядит следующим образом: необходимо задать функции $\phi_i : \mathcal{S} \to \mathbb{R}$, $i = \overline{1, p}$. Назовём ее значение на конкретной тройке *важностью i-ого признака.* На практике мы не знаем распределения признаков, поэтому тройка $(\mathbf{X}, \mathbf{Y}, \mathbf{a})$ заменяется на $(X, Y, \mathbf{a})$.

Естественно, интерес заключается в нахождении важности, которая:

- имеет большие по модулю значения на более релевантных признаках,

- быстро считается,
- дает дополнительную информацию об устройстве модели,
- устойчива к шуму в данных.

§4. Методы

4.1. Filter методы

Широко известный пример — *линейный коэффициент корреляции Пирсона*:

$$\rho_{\mathbf{X}_i,\mathbf{Y}} = \frac{\mathbb{E}[\mathbf{X}_i\mathbf{Y}] - \mathbb{E}[\mathbf{X}_i]\mathbb{E}[\mathbf{Y}]}{\sqrt{\mathbb{E}\left[\mathbf{X}_i^2\right] - (\mathbb{E}[\mathbf{X}_i])^2}\sqrt{\mathbb{E}\left[\mathbf{Y}^2\right] - (\mathbb{E}[\mathbf{Y}])^2}} = \phi_i.$$

Если две переменные сильно коррелируют с $\mathbf{Y}$, но $\mathbf{Y}$ в действительности зависит от одной, стоит воспользоваться ранговым аналогом: вместо значений $\mathbf{X}_i, \mathbf{Y}$ берутся их позиции в упорядоченной версии $\mathbf{X}_i^{sorted}, \mathbf{Y}^{sorted}$. Такие коэффициенты измеряют степень только линейной (неранговые) или монотонной (ранговые) зависимости.

Другой подход, позволяющий уловить нелинейную связь — *взаимная информация*:

$$\mathrm{I}(\mathrm{X};\mathrm{Y}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}} \mathbb{P}_{X,Y}(x,y)\log_2\frac{\mathbb{P}_X(x)\mathbb{P}_Y(y)}{\mathbb{P}_{X,Y}(x,y)},$$

$$\mathrm{I}(\mathrm{X};\mathrm{Y}\mid\mathrm{Z}) = -\sum_{x\in\mathcal{X}}\sum_{y\in\mathcal{Y}}\sum_{z\in\mathcal{Z}} \mathbb{P}_{X,Y,Z}(x,y,z)\log_2\frac{\mathbb{P}_{X|Z}(x\mid z)\mathbb{P}_{Y|Z}(y\mid z)}{\mathbb{P}_{X,Y|Z}(x,y\mid z)},$$

где $\mathbb{P}_{X,Y}(x,y)$ — вероятность встретить объект (x,y) в выборке. Взаимная информация имеет несколько полезных свойств:

- $\mathrm{I}(\mathrm{X};\mathrm{Y}) \geq 0$,
- $\mathrm{I}(\mathrm{X};\mathrm{Y}\mid\mathrm{Z}) = \mathrm{I}(\mathrm{Y};\mathrm{X}\mid\mathrm{Z})$,
- $\mathrm{I}(\mathbf{X};\mathbf{Y}) = 0$ тогда и только тогда, когда $\mathbf{X}$ и $\mathbf{Y}$ независимые случайные величины,
- $\mathrm{I}(\mathrm{X},\mathrm{Z};\mathrm{Y}\mid\mathrm{W}) = \mathrm{I}(\mathrm{X};\mathrm{Y}\mid\mathrm{W}) + \mathrm{I}(\mathrm{Z};\mathrm{Y}\mid\mathrm{W},\mathrm{X})$.

Хотя это является огромными плюсами для метода, всё же он не всегда хорош. Например, когда ошибка MSE распределена по Стьюденту, алгоритм выбирает нелучшие с точки зрения MSE подмножества признаков [3].

4.2. Embedded методы

L_1 *регуляризация* является достаточно простым способом выявления «хороших» признаков. Однако с увеличением уверенности, что какой-то признак важный, уменьшается сложность модели.

В статье [7] авторы не прибегают к подобным «трюкам», а используют *среднее увеличение количества информации* (Mean Decrease Impurity):

$$\phi_m = \frac{1}{N_T}\sum_T \sum_{t\in T: v(s_t)=X_m} p(t)\Delta i\left(s_t,t\right),$$

где T — дерево, N_T — количество деревьев, $p(t)$ — доля объектов дошедших до узла t, $v\left(s_t\right)$ — признак по которому произошло разделение s_t в узле t, $\Delta i\left(s_t,t\right)$ — изменение количества информации в узле t с разбиением s_t. Основные результаты представлены для *полностью случайных и до конца построенных* (totally randomized and fully developed) деревьев.

В частности, при бесконечно большой выборке категориальных данных справедливо:

$$\phi_m = \sum_{k=0}^{p-1} \frac{1}{C_p^k} \frac{1}{p-k} \sum_{B \in \mathcal{P}_k(V^{-m})} I\left(X_m; Y \mid B\right), \tag{1}$$

$$\sum_{m=1}^{p} \phi_m = I\left(X_1, \ldots, X_p; Y\right), \tag{2}$$

где $\mathcal{P}_k\left(V^{-m}\right)$ — множество подмножеств мощности k, не включающих признак m. Формула (1) даёт нам полноценное представление о зависимости признака и целевой переменной. Здесь присутствует разложение как по мощности множества взаимодействия с другими признаками (сумма по k), так и по её степени (сумма по подмножествам). Оказывается, если ограничить глубину деревьев до $q \leq p$, важность будет равна первым q слагаемым из первой суммы в (1). Разбиение наглядно визуализируется на Рис. 1. Можно заметить, что некоторые признаки становятся неважными в присутствии других. В случае случайного леса признаки X_2, X_5 находились вверху дерева. Это привело к «скрытым эффектам»: часть признаков вносят свой вклад только при обуславливании с X_2, X_5.

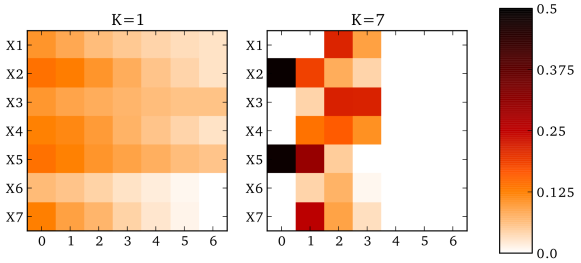


Рис. 1: Важность признаков в зависимости от мощности множества, на котором она обуславливается. Значение в клетке (i, j) — j-ое слагаемое из первой суммы в (1) для X_i. На картинке слева используется обычное дерево, справа — случайный лес [7]

Это также было замечено в [15], где авторы предложили использовать имплементированный ими метод построения *условных деревьев* (conditional trees [ctree]). Для каждого узла выбор переменной X_j для расщепления осуществляется исходя из значений p-критериев независимости условного выхода $Y|X_i, i = \overline{1,p}$, посчитанного на основе данных, дошедших до узла. Такой подход в отличии от gini importance более устойчив к технике сэмплирования с возвратом в случайном лесе. Еще одним решением может стать *перестановочная важность* (permutation importance). О ней пойдёт речь в секции 4.3.

4.3. Wrapper методы

Наиболее простым и эффективным методом является *перестановочная важность*. Для её вычисления необходимо сравнить выход модели на двух выборках: исходной и перемешанной по интересующему признаку. Такой подход сохраняет маргинальное распределение $\mathbf{X}_i$ и прост в вычислении, однако при сильнокоррелированных признаках он склонен занижать значимость, в частности, в случае аддитивной регрессионной модели [6]. Данный подход можно развить и на группу признаков, как это сделано в статье [5]:

$$\phi_J = \mathbb{E}\left[\left(\mathbf{Y} - \mathbf{a}\left(\mathbf{X}_{(J)}\right)\right)^2\right] - \mathbb{E}\left[\left(\mathbf{Y} - \mathbf{a}(\mathbf{X})\right)^2\right] = \mathrm{R}(\mathbf{a}, \mathbf{X}_{(J)}) - \mathrm{R}(\mathbf{a}, \mathbf{X}),$$

$$\hat{\phi_J} = \frac{1}{N_T} \sum_T \left[\hat{\mathrm{R}}\left(T, \mathrm{oob}(\hat{\mathbf{X}}_{(J)}, T)\right) - \hat{\mathrm{R}}\left(T, \mathrm{oob}(\hat{\mathbf{X}}, T)\right)\right].$$

где $\hat{}$ обозначает эмпирический аналог, $\mathbf{X}_{(J)}$ — случайный вектор, полученный заменой в $\mathbf{X}$ случайных признаков $\mathbf{X}_J$ на их независимую от $\mathbf{Y}$ и оставшихся признаков копию, $\mathrm{oob}(\hat{\mathbf{Z}}, T)$ — те объекты $\hat{\mathbf{Z}}$, которые не попали в *тренировочную выборку* дерева T. В случае аддитивной регрессионной модели ϕ_J пропорционален дисперсии ответов на соответствующем подмножестве признаков. С помощью *графика частичной зависимости* (PDP) это можно наглядно увидеть.

Частичная зависимость (partial dependence) может и сама быть инструментом для подсчета важности:

$$\mathrm{PD}^{\mathrm{IndSet}}(x) = \mathbb{E}\left[\mathbf{a}(z) \mid z_{\mathrm{IndSet}} = x_{\mathrm{IndSet}}\right], \quad \mathrm{IndSet} \subseteq \mathrm{IndP}.$$

В pyBreakDown [14] авторы хотят найти такие признаки, чтобы при их небольшом возмущении прогноз модели существенно изменился. Это они добиваются следующим образом. Начальное множество $\mathrm{IndSet} = \varnothing/\mathrm{IndP}$. На каждой итерации алгоритма в/из IndSet добавляется/удаляется один признак. В первом варианте он максимизирует $\left|\mathrm{PD}^{\mathrm{IndSet}}(x) - \mathrm{PD}(x)\right|$ (*step-up*), во втором минимизирует (*step-down*). В итоге получается последовательность признаков с убывающей/увеличивающейся важностью, так как она определяется, как разность между соответствующими $\mathrm{PD}(x)$ на соседних IndSet.

Можно сделать предположение, что важность признака больше, если он сильнее взаимодействует с другими. Это можно формально выразить через показатель [9]:

$$\mathrm{H}_{\mathrm{j}}^2 = \sum_{i=1}^{n}\left[\mathbf{a}\left(x^{(i)}\right) - \mathrm{PD}^{\{j\}}\left(x^{(i)}\right) - \mathrm{PD}^{\{-j\}}\left(x^{(i)}\right)\right]^2 / \sum_{i=1}^{n}\mathbf{a}^2\left(x^{(i)}\right),$$

где $\mathrm{PD}^{\{-j\}} = \mathrm{PD}^{\mathrm{IndSet}}$, $\mathrm{IndSet} = \mathrm{IndP}\setminus\{j\}$. $\mathrm{H}_{\mathrm{j}}^2$ положительная и равна 0 тогда и только тогда, когда признак не взаимодействует с другими. Однако, как показано в разделе 5.1.2, это не очень подходящее определение значимости.

Рассмотрим один из самых популярных методов построения важности. Пусть у нас есть некоторая характеристическая функция $v : 2^{\mathrm{IndP}} \to \mathbb{R}$ такая, что $v(\varnothing) = 0$. Будем искать определение важности, удовлетворяющее следующим свойствам:

- *сумма в конечный ответ* (efficiency): $\sum_{i\in\mathrm{IndP}}\phi_i(v) = v(\mathrm{IndP})$,
- *аддитивность* (additivity): $\forall v, w : \phi(v+w) = \phi(v) + \phi(w)$, где $(v+w)(S) = v(S) + w(S)$ $\forall S \subseteq \mathrm{IndP}$,
- *симметрия* (symmetry): Если $v(S\cup\{i\}) = v(S\cup\{j\})$ $\forall S$, где $S \subset \mathrm{IndP}$ и $i, j \notin S$, тогда $\phi_i(v) = \phi_j(v)$,
- *корректность* (dummy): Если $v(S\cup\{i\}) = v(S)$ $\forall S$, где $S \subset \mathrm{IndP}$ и $i \notin S$, тогда $\phi_i(v) = 0$.

Оказывается аддитивность и симметричность вместе эквивалентна *согласованности* (consistency) [8]: Если $\forall v, w; \forall S : i \notin S$ выполнено $v(S\cup\{i\}) - v(S) \geq w(S\cup\{i\}) - w(S)$, тогда $\phi_i(v) \geq \phi_i(w)$. Существует единственная важность, удовлетворяющая данным требования. Это *Shapley values*:

$$\mathrm{Sh_i(v)} = \sum_{S\subseteq\mathrm{IndP}\setminus\{i\}}\frac{1}{p}\frac{1}{\binom{p-1}{|S|}}(v(S\cup\{i\}) - v(S)), \quad i = \overline{1,p}.$$

В данном случае учитывается вклад i-ого признака во всевозможные подмножества других. В [16] рассматривается другой подход задать такие значения, помогающий избежать экспоненциальной сложности:

$$\phi_i = \frac{1}{p!\cdot|\mathcal{X}|}\sum_{O\in\pi(p)}\sum_{y\in\mathcal{X}}\left[\mathbf{a}\left(\tau\left(x, y, \mathrm{Pre}^i(O)\cup\{i\}\right)\right) - \mathbf{a}\left(\tau\left(x, y, \mathrm{Pre}^i(O)\right)\right)\right],$$

$$\tau(x, y, S) = (z_1, z_2, \ldots, z_p), \quad z_i = \begin{cases} x_i & ; \quad i \in S \\ y_i & ; \quad i \notin S \end{cases}$$

где $\pi(p)$ — множество упорядоченных перестановок длины p, $\mathrm{Pre}^i(O)$ — множество индексов, которые стоят перед i в $O \in \pi(p)$. Авторы считают $65000\cdot p$ итераций совместного сэмплирования перестановки и элемента выборки достаточным для аппроксимации с ошибкой 0.01 для 99% переменных.

Рассмотрим некоторые нейросетевые определения значимости признаков. Метод DeepLift [13] основан на разделении отрицательного и положительного вклада в целевую переменную. Таким образом возможно избежать некоторые проблемы, связанные с обнулением градиентов и отсутствием изменения ответов модели при перестановке входов. Пусть есть начальное значение x_{m0}, y_0. Положим $\Delta y = y - y_0, \Delta x_m = x_m - x_{m0}$, тогда:

$$\phi_m = m_{\Delta x_m \Delta y} \Delta x_m,$$

где $m_{\Delta x_m \Delta y} = const$. Выполняются свойства:

- *Сумма в дельту* (summation to delta):

$$\sum_{i=1}^{p} \phi_i = \Delta y.$$

- *Цепное правило* (chain rule):

$$m_{\Delta x_i \Delta y} = \sum_j m_{\Delta x_i \Delta z_j} m_{\Delta z_j \Delta y}.$$

Внутренние состояния пересчитываются через цепное правило и специальное определение мультипликаторов, которое учитывает появление «знаковых Δ» в присутсвии или отсутствии Δ другого знака. Данный подход решил часть проблем, но некоторые всё же остаются. Например, трансформация коэффициентов при проходе через max_pool слой.

Если посчитать для части модели коэффициенты $m_{\Delta x_m \Delta y}$, используя Shapley values, получим DeepShap [8].

Shapley values можно аппроксимировать с помощью линейной регрессии, если взять MSE лосс с определенным ядром. Тогда мы получим так называемый Kernel SHAP [8]. Он сходится гораздо быстрее в отличии от простого сэмплирования Shapley values.

Другой подход связан с построением так называемой *объясняющей модели* (explanation model). В CXplain [12] используется Granger's определение причинности взаимосвязи между признаками и целевой переменной, в котором:

- все признаки релевантные,
- признак временно предшествует метке, то есть для того, чтобы получить метку, нужна информация о признаке.

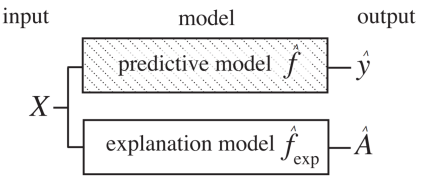


Рис. 2: Важность в CXPlain. Объясняющая модель $\hat{f}_{\text{exp}}$ обучается выдавать важность признаков $\hat{A}$ для исходной модели $\hat{f}$ [12]

Важность определяется как нормированная разница ошибок объясняемой модели на маскированных данных и исходных. У объясняющей модели:

- цель — предсказать важность признаков,
- вход — элемент из выборки,
- лосс — расстояние Кульбака — Лейблера между истинным и предсказанным распределениями важности признаков.

Заметим, что данная модель нужна когда нет истинных меток объектов. Маскирование X_i подразумевает замену значений в X_i на другие так, чтобы связь с оставшимися признаками пропала. Замену значений можно осуществить несколькими способами: усреднить значения признака, заполнить константой, перемешать. Для устойчивости авторы обучают ансамбль

обучающих моделей на сэмплированных выборках. Итоговая важность — медиана предсказаний ансамбля, а точность — интерквартильный размах. В таком подходе точность оценки важности коррелирует с ошибкой ранжирования важности признаков. Даже при небольшой мощности ансамбля хорошо оценивается точность. Сильной стороной данного метода является быстрота, что как мы помним, оказалась краеугольным камнем в Shapley values.

4.4. Mix методы

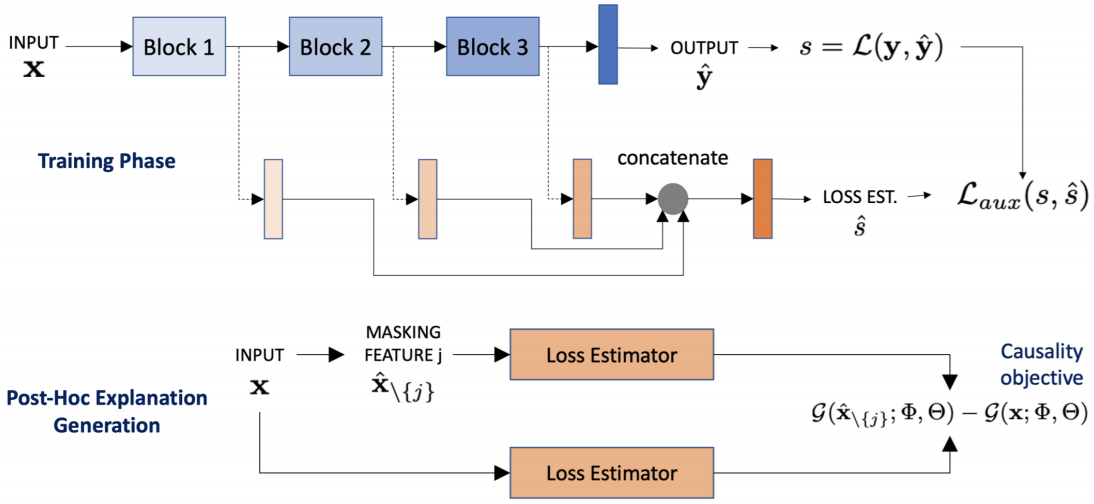


Рис. 3: Схема работы алгоритма получения важности в PRoFILE. Синим цветом отмечена исходная модель, а коричневым — объясняющая [18]

Использование не только выходов исходной модели, а также её внутреннего представления и знания о лоссе дают возможность построить «хорошую» объясняющую модель. В PRoFILE (см. рис. 3) у нее:

- цель — научиться предсказывать лосс основной сети,
- вход — латентные представления после некоторых слоёв основной сети, за каждым из которых следует линейный слой,
- лосс — $\sum_{(i,j)} \max\left(0, -\mathbb{I}\left(s_i, s_j\right) \cdot \left(\hat{s}_i - \hat{s}_j\right) + \gamma\right)$,

  где $\mathbb{I}\left(s_i, s_j\right) = \begin{cases} 1 & \text{, если } s_i > s_j \\ 0 & \text{, иначе} \end{cases}$ $\quad s_* = \mathcal{L}_{mainnet}(y_*, \mathbf{a}(x_*)),$ $\quad \hat{s}_* = \mathcal{L}_{expnet}(s_*, \exp\_\text{model}(x_*)).$

Стоит заметить, что градиент от ошибки объясняющей модели влияет на слои в основной. Таким образом, мы тренируем модели совместно. В отличии от Shap, CXplain и Lime данный подход устойчив к «помехам» в данных: на датасетах Cifar10-C, MNIST-USPS PRoFILE оказался лучше по показателю качества:

$$\Delta \text{ log-odds } = \text{ log-odds } \left(p_{\text{ref}}\right) - \text{ log-odds } \left(p_{\text{masked}}\right),$$

где $\text{log-odds } (\text{p}) = \log\left(\frac{\text{p}}{1-\text{p}}\right)$, p_{ref} — вероятность, полученная на оригинальных данных, а p_{masked} — на маскированных.

Рассмотрим другой подход. Допустим мы знаем заранее важность скольких признаков хотим найти, например, s штук. Тогда логичным способом получения таких ϕ_i может стать обучение нейронной сети, которая выдает нам набор переменных, входящих в интересующее множество. В одном из методов ранжирования оценок важности признаков (FIR) (см. рис. 4) обучение происходит поочередно. Оно сочетает в себе два этапа, где маски для признаков — бинарные вектора (1 - берем признак, 0 - нет) длины d:

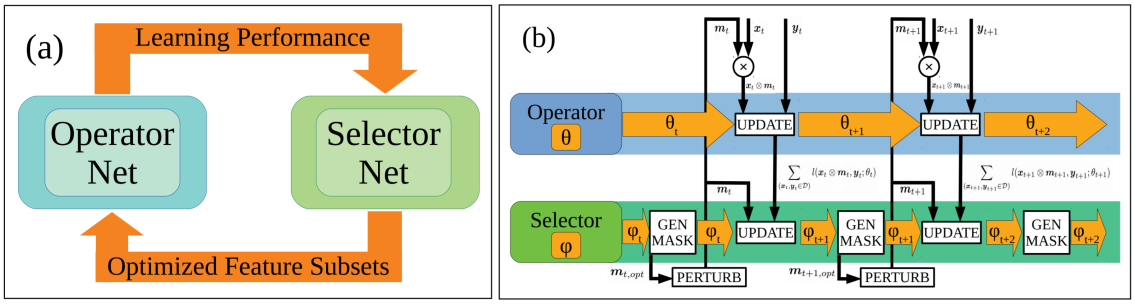


Рис. 4: Схема работы алгоритма получения важности в FIR [19]

1. operator net генерирует обучающую выборку для selector net: пары масок и соответствующий лосс на них
2. selector net передаёт operator net'у следующие «хорошие» маски:
   (a) лучшая маска с предыдущей итерации,
   (b) маска, получаемая результатом следующего алгоритма:
      i. стартуем с маски $\boldsymbol{m}_0 = \left(\frac{1}{2}, \cdots, \frac{1}{2}\right)$, выбираем топ s компонент градиента selector net'а. Валидируем полученную «оптимальную» маску:
         A. берем топ s компонент градиента только теперь уже в точке, равной полученной маске. Таким образом получаем две маски: $\boldsymbol{m}_{\text{opt}}$ — содержит s единиц, $\overline{\boldsymbol{m}}_{\text{opt}}$ — содержит $d - s$ единиц.
         B. заменяем компоненту маски $\boldsymbol{m}_{\text{opt}}$ с отрицательным градиентом на компоненту с наибольшим градиентов в маске $\overline{\boldsymbol{m}}_{\text{opt}}$
         C. проверяем условие $\text{selector\_net}\left(\boldsymbol{m}_{opt}\right) \leq \text{selector\_net}\left(\boldsymbol{m}'_{opt}\right)$, где $\boldsymbol{m}'_{\text{opt}}$ получена заменой компоненты маски $\boldsymbol{m}_{opt}$ с наименьшим градиентом на компоненту маски $\overline{\boldsymbol{m}}_{\text{opt}}$ с наибольшим. Если это условие не выполнено повторяем A.-B.
   (c) полученная $\boldsymbol{m}_{\text{opt}}$ на самом деле может быть неоптимальной, поэтому добавляем небольшую случайность: случайно выберем $s_{rand} < s$ компонент в $\boldsymbol{m}_{\text{opt}}$ /$\overline{\boldsymbol{m}}_{\text{opt}}$, инвертируем их и поменяем значения местами с другой маской. Сделаем так несколько раз.

В итоге у operator net:

- цель — обучение с учителем конкретной задачи,
- вход — x и маска признаков,
- лосс — соответствующий задаче.

А у selector net:

- цель — предсказать loss operator net,
- вход — маска признаков,
- лосс — MSE с лоссом, переданным от operator net.

Важность признака — соответствующая компонента градиента лосса selector net'а в точке оптимального набора признаков. Процесс построения оптимального набора очень долгий, но как показали результаты экспериментов, точность среди DFS, RF, RFE оказалась лучшая, как и качество выбранных признаков исходя из задачи. Однако время работы несравнимо больше: x440 дольше RF и x2 дольше Lime.

§5. Эксперименты

Во всех экспериментах участвуют следующие модели:

- **бустинг (LightGBM)** со стандартными параметрами,
- **метод опорных векторов (SVM)** с линейным ядром,
- **полносвязная нейронная сеть (MLP)** с параметрами:
  - 2 слоя,
  - 64 нейронов на слой,
  - dropout = 0.2 (с такой вероятностью выход нейрона случайно зануляется),
  - batch_size = 8,
  - num_epoch = 100,
  - early_stopping_patience = 15.

В качестве методов для оценки важности признаков выступают:

- **rfpimp (permutation)** — перестановочная важность с f1 показателем качества,
- **eli5 (gain)** — среднее увеличение количества информации для gini importance,
- **pyBreakDown (up)** — описано в разделе 4.3,
- **shap (Tree/Kernel)** — для бустинга использовался вариант Tree, так как он позволяет посчитать истинные Shapley values. Однако сложность вычисления все равно достаточно большая для используемых данных, поэтому сэмплировалось приблизительно 100–200 объектов из выборки для подсчета важности X_i. Если мы работаем с SVM, то используется Kernel версия. Как отмечено раннее, она также приближает Shapley values, однако скорость сходимости и точность гораздо меньше по сравнению с Tree версией,
- **lime** — метод взят из открытого источника [10], разработанного авторами данного метода [11],
- **CXplain [exp_m (p, mlp), exp_m (p, lgbm)]** — CXplain с перемешивающим маскированием и MLP/LightGBM в качестве объясняющей модели. Версия авторов метода в открытом доступе показала неинформативные результаты (все признаки одинаково важны). Поэтому была разработана аналогичная версия, где использовалась MSE/NLL ошибка в задаче регрессии/классификации,
- **permutation (nll)** — перестановочная важность с nll ошибкой. Эта оценка важности является целевой переменой для объясняющей модели в CXplain (nll).

Для таких методов, как pyBreakDown, shap, lime, CXplain важность X_i получалась усреднением важности X_i для всех объектов из *валидационной выборки*. Код всех экспериментов можно найти по ссылке на github репозиторий.

5.1. Коррелированные признаки

5.1.1 Данные

Рассмотрим задачу классификации. Признаки состоят из пяти групп (gr_1, ..., gr_5) и одного шумового (rand). Каждая группа состоит из трёх признаков: gr_{ij}, j = 1, 2, 3. Группы независимы от друг друга и сэмплированы из нормального распределения $\mathcal{N}_3(0, C_i)$, где

$$C_i = \begin{pmatrix} 1 & \rho_i & \rho_i \\ \rho_i & 1 & \rho_i \\ \rho_i & \rho_i & 1 \end{pmatrix}, \quad \rho_i = \frac{i}{6}.$$

Шум был взят из распределения $\mathcal{N}_3(0, 1)$. Пусть $\boldsymbol{\xi}_i$ имеет биномиальное распределение $Bi(1, 0.5)$.

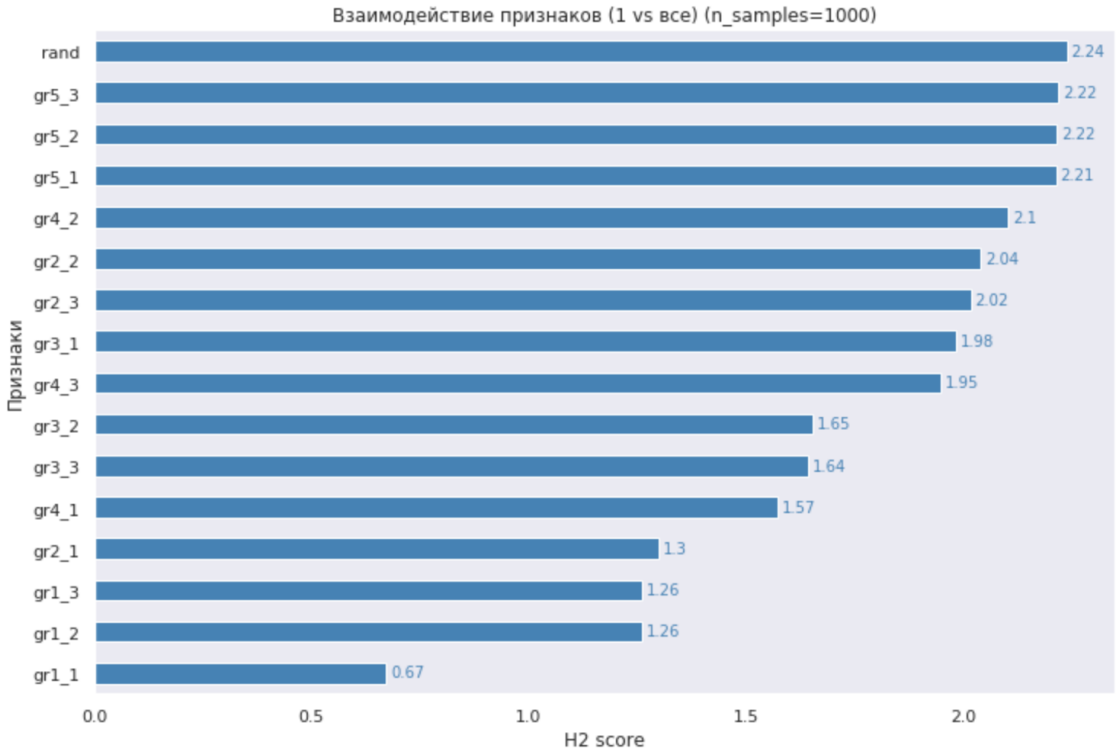
Тогда целевая переменная Y равна

$$\mathbb{I}[Z \geq \text{median}(Z)], \quad \text{где}$$
$$Z = \text{sigmoid}(\xi_1 \, \text{gr}_{11} + ... + \xi_5 \, \text{gr}_{51}).$$

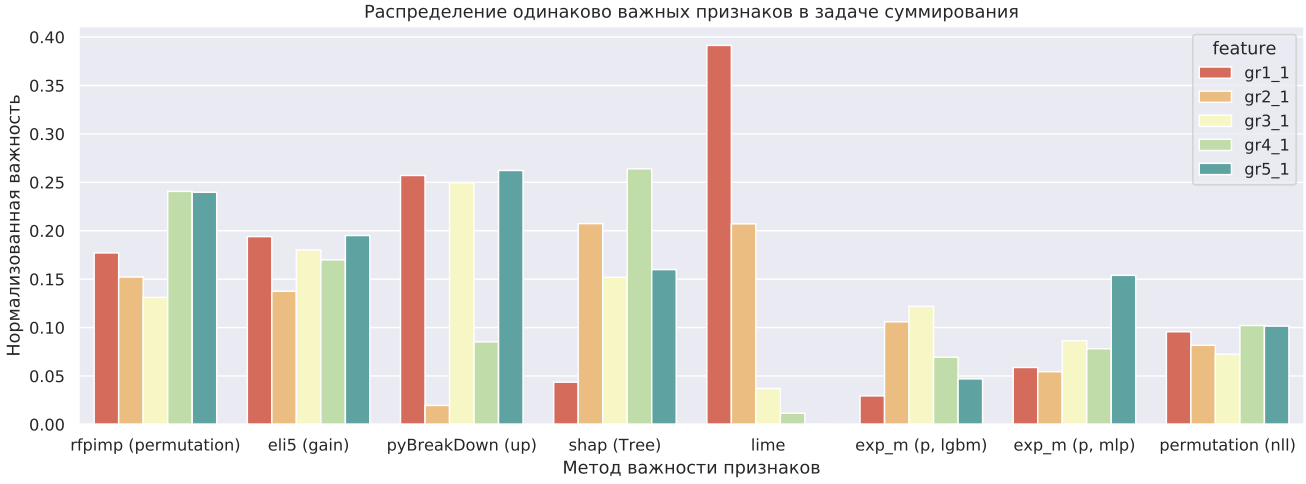
В данной задаче признаки $\text{gr}_{11}, ..., \text{gr}_{51}$ одинаково важны для предсказания Y. Количество сэмплированных объектов равно 5000.

5.1.2 Важность

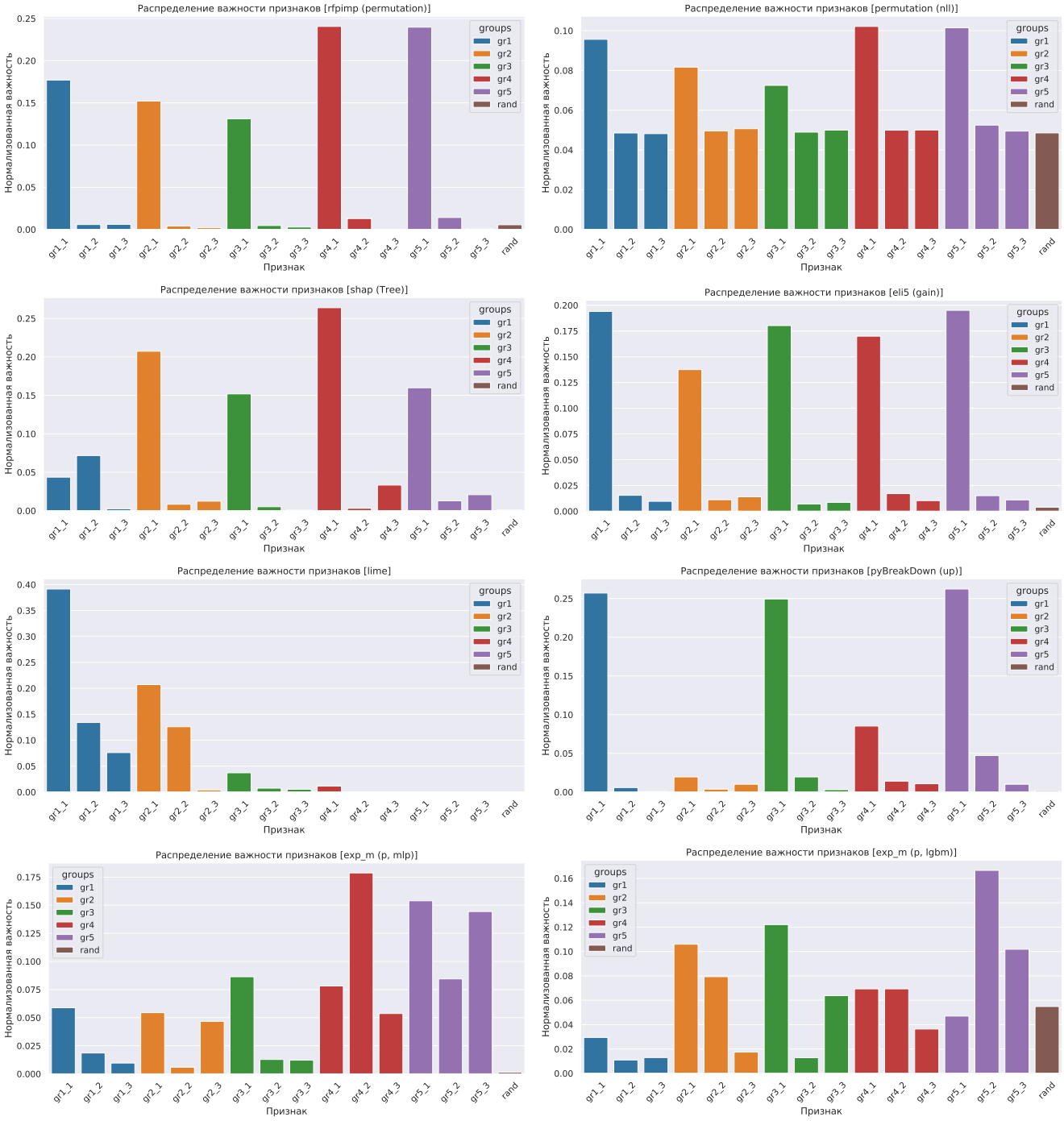
Посчитаем H_2 score для наших данных.



Видно, что случайный признак оказался на первом месте. Чем больше групповые корреляции, тем больше взаимодействия получают признаки.



Наиболее равномерно распределены перестановочная, gain, Shapley, CXplain важности. Lime оставляет 2 наиболее важных признака, исходя из их независимости. Рассмотрим оценки важности признаков подробнее.



Выбор оценочной функции для перестановочной функции важен. В данном случае f1 качество оказалась более подходящим. Shap устойчив к средним и большим корреляциям, но не к малым (gr_{12} самый важный в группе gr_1, а не gr_{11}). Стоит заметить практически нулевую важность случайного признака. Eli5 выдает одинаковую важность для групп с разной корреляцией, что говорит об его устойчивости к ней. Lime в силу своего построения отдает предпочтение малому количеству признаков, причем с характерным экспоненциальным убыванием важности. PyBreakDown склонен выделять с большой уверенностью небольшое количество важных признаков. Если выделить в каждой группе gr_i самый важный признак, то exp_m (p, lgbm) и exp_m (p, mlp) похожи. Однако есть несколько отличий. В exp_m (p, mlp) чем больше групповая корреляция, тем больше суммарной важности в группе. Кроме того, для Lgbm важности признаков кажутся более хаотичными, что может свидетельствовать о переобучении. Относительно большая важность случайного признака также это подтверждает.

5.1.3 Удаление признаков

Рассмотрим задачу рекурсивного удаления признаков (RFE). На каждой итерации удаляется наименее важный признак и замеряется качество на *тестовой выборке*.

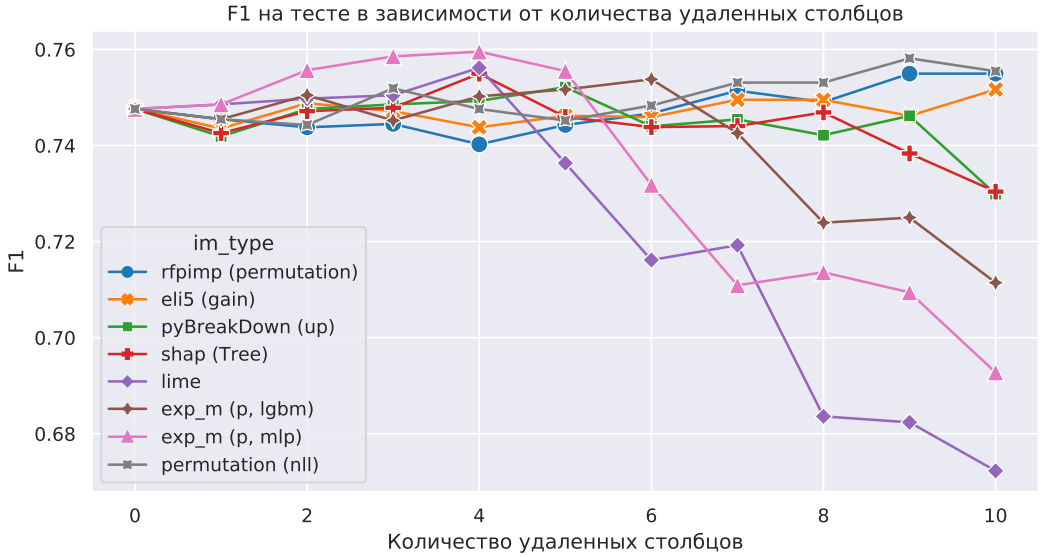


Рис. 5: Один запуск эксперимента RFE. Конечное количество признаков = 30% от исходного. Для вычисления shap сэмплировались 200 объектов из выборки.

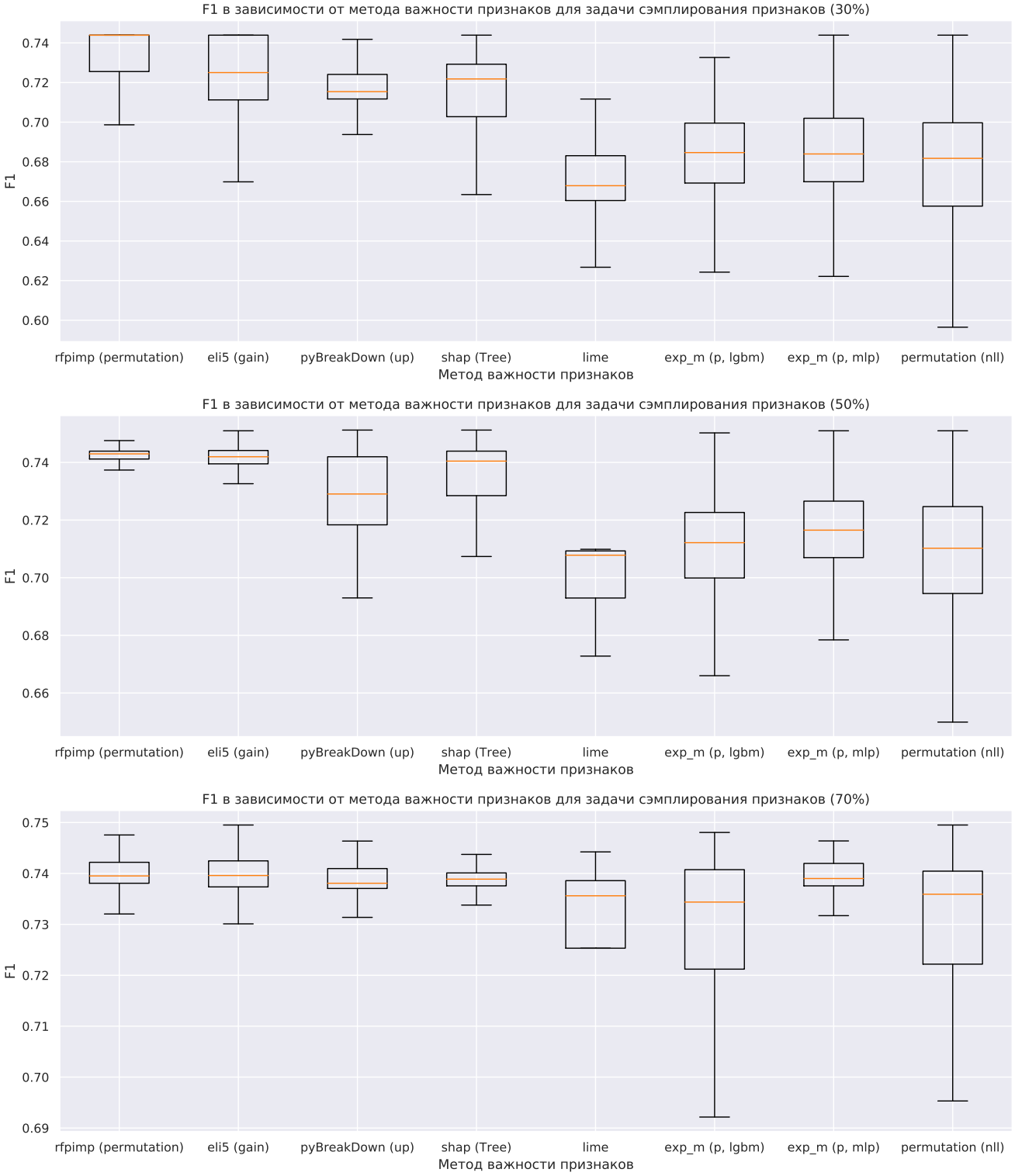
Перестановочная и gain важность лучше всего отбирают признаки «на будущее». Для перестановочной важности ошибка nll (серая линия) оказалась все время не ниже ошибки f1 (синяя линия). Исходя из распределения оценок важности признаков для nll можно сделать вывод, что коррелированные переменные служат хорошей заменой друг другу. Дерево действительно инвариантно к сдвигу или масштабированию признака.

Lime не учитывает зависимости второго и более порядков. Это может стать причиной выбрасывания относительно «хороших» признаков на ранних стадиях.

В идеальном варианте exp_m оценки важностей должны совпадать с permutation (nll). Этого не происходит по двум вариантам: объясняющая модель слишком простая или мало данных. Учитывая, что обучение объясняющей модели происходило на валидационной выборке второй вариант вероятнее.

5.1.4 Сэмплирование признаков

Посмотрим, как важность хороша с точки зрения дальнейшего обучения м одели. Для каждого метода мы взяли нормализованный модуль важности и просэмплировали 1000 раз определенное количество признаков (30%, 50%, 70% от исходного количества). После чего замерили качество (f1-качество) на *тестовой выборке*.



При большом количестве признаков перестановочная, gain, pyBreakDown, shap, exp_m(p, mlp) работают примерно одинаково. Однако при меньшем количестве признаков качество у pyBreakDown резко падает из–за итеративного алгоритма работы. Перестановочная важность (rfpimp) имеет лучшие результаты в целом.

5.1.5 Копия признака

Выберем самый важный признак. Добавим его копию. На каждой итерации будем удалять наименее важный признак, за исключением выбранного и его копии. Обучать заново модель и повторять процедуру по достижению определенного количества оставшихся признаков

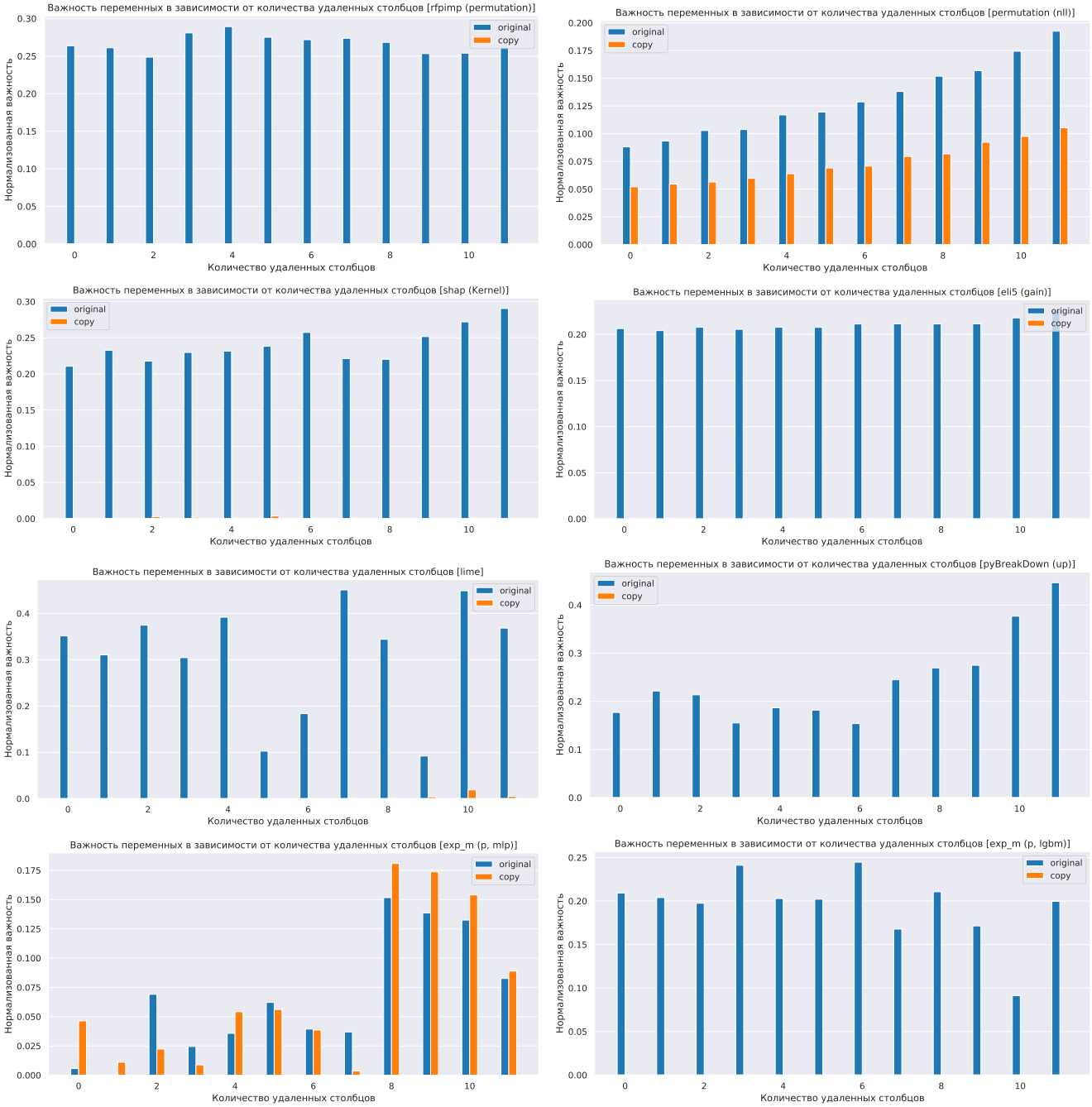


Рис. 6: Исходная модель LGBM. Конечное количество признаков= 30% от исходного. Для сэмплирования shap использовалось 100 объектов.

В большинстве случаев деревья не берут копию в качестве признака для разделения данных. Как следствие, значение копии равняется нулю. Однако с точки зрения (permutation (nll) вероятности положительного класса меняются.

Возьмём SVM в качестве модели.

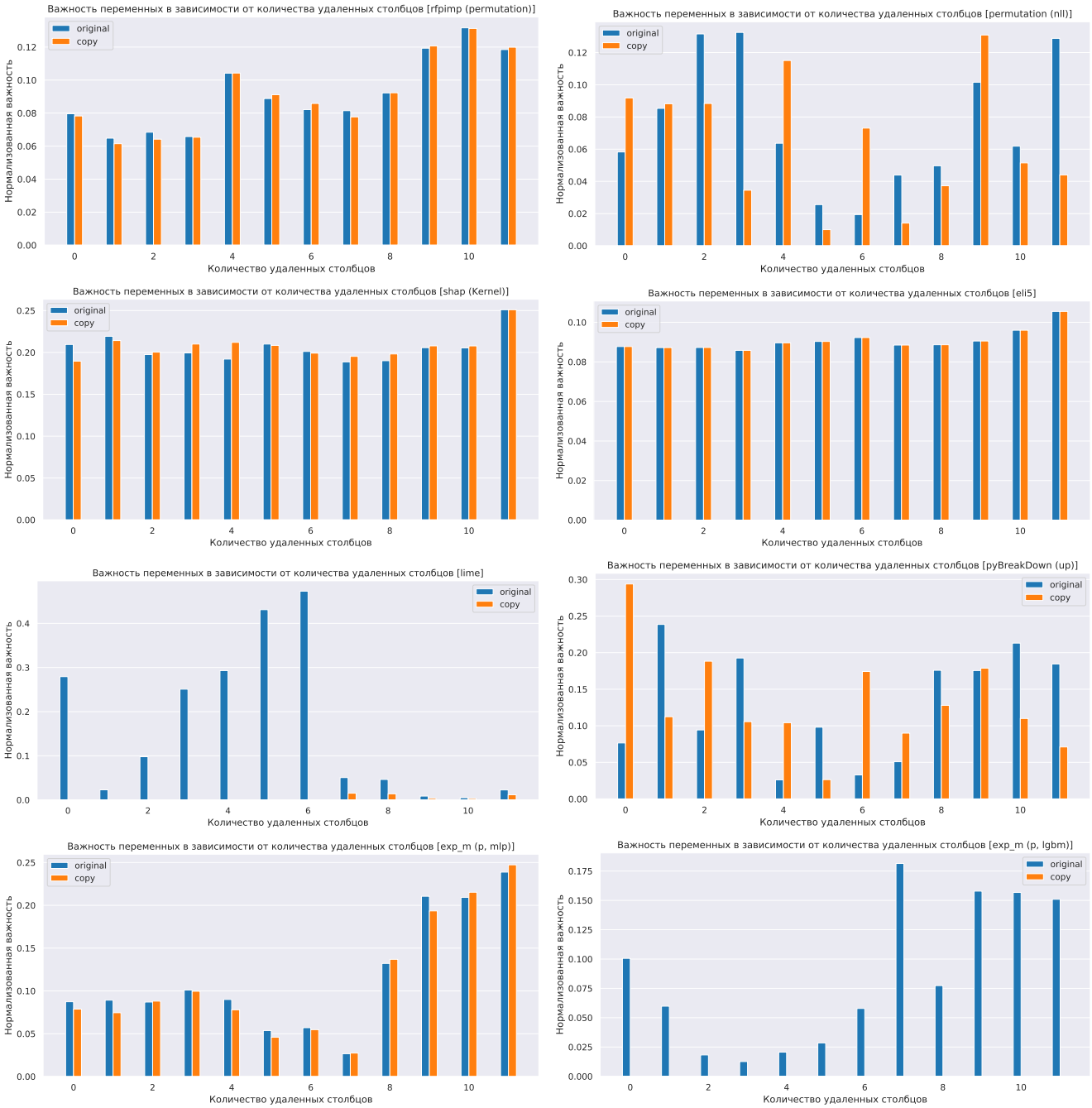


Рис. 7: Исходная модель SVM. Конечное количество признаков= 30% от исходного. Для сэмплирования shap использовалось 100 объектов.

Симметричные важности дают rfpimp, eli5, shap, exp_m (p, mlp). Если выбирать данные методы, как способ фильтрации нужных переменных для предсказания, стоит обращать внимание на признаки, похожие друг на друг. Так как деление важности среди группы очень близких по значениям признаков скроет истинную оценку важности. Решением проблемы может стать выполнение иерархической кластеризации на основе ранговой корреляции Спирмена, выбор порога и сохранение одного признака из каждого кластера.

Использование линейного слоя нейронной сети в качестве объясняющей модели помогло избавиться от асимметричного распределения важности.

5.2. Функции

Рассмотрим некоторые простые функции от многих переменных.

5.2.1 Данные

Признаки $X_1, ..., X_6, eps$ связаны с Y следующим образом:

$$Y_{reg} = X_1X_2X_3 + X_4 + X_5 + X_6 + eps, \quad \text{где } esp \sim N\left(0, 0.1^2\right), X_i \sim N(0,1)$$
$$Y_{clf} = \mathbb{I}[Z \geq \text{median}(Z)], \quad \text{где } Z = \sigma\left(Y_{reg} - \text{mean}\left(Y_{reg}\right)\right)$$

Количество сэмплированных объектов равно 1000. В качестве модели использовался бустинг (LightGBM).

5.2.2 Стандартная классификация/регрессия

Рис. 8: $y = y_{clf}$

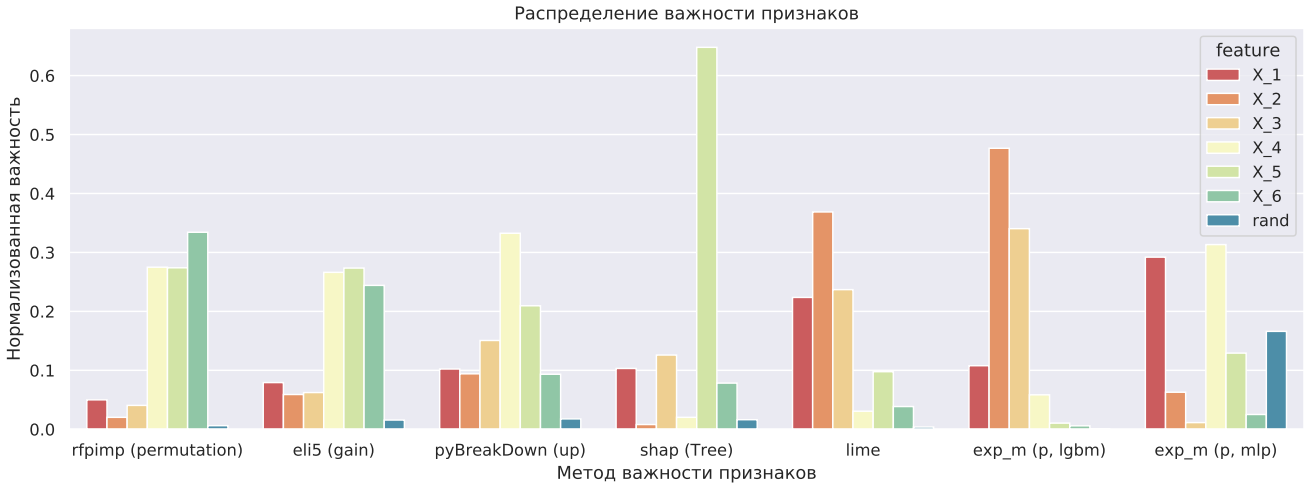


Рис. 9: $y = y_{reg}$

Наименьшую важность случайному признаку дает rfpimp, lime. Перестановочная важность отдает предпочтение более независимым признакам: X_4, X_5, X_6. Более того, сумма важностей X_1, X_2, X_3 не равняется важности X_4, или X_5, или X_6. Это говорит о том, что признаки участвующие в сложных взаимосвязях могут получить маленькую оценку важности.

5.2.3 Квадратичная функция

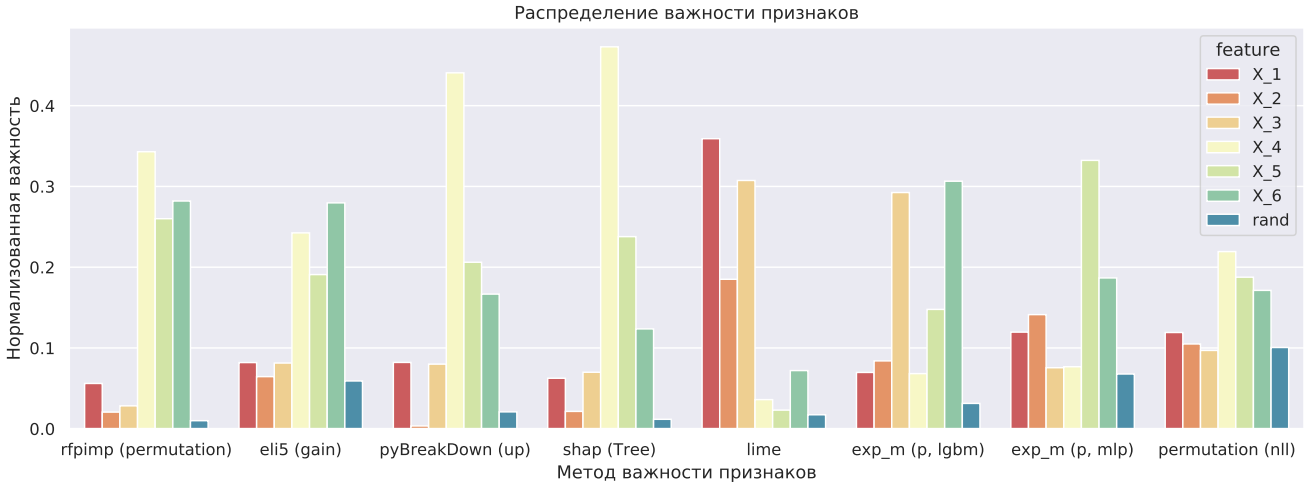


Рис. 10: $y = y_{clf}\left(y^2_{\text{reg}}\right)$

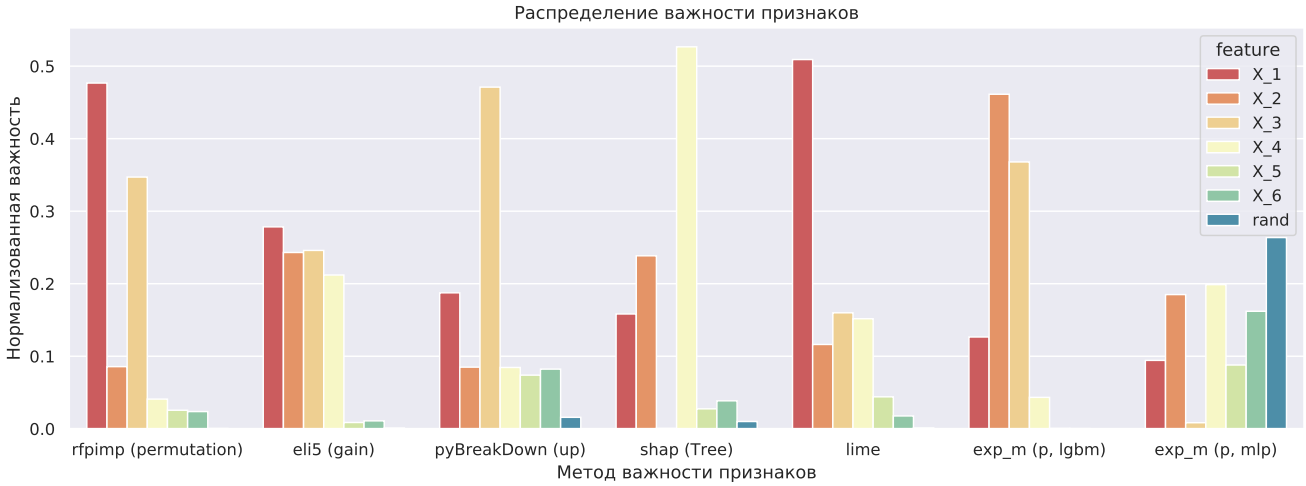


Рис. 11: $y = y^2_{reg}$

Признаки X_1, X_2, X_3 сохранили свою важность для $y = y_{clf}\left(y^2_{\text{reg}}\right)$ из-за поведения сигмоиды на бесконечности: при относительно больших/малых z, $\text{sigmoid}(z)$ не сильно меняется. Для $y = y^2_{reg}$ же сигмоида не применялась.

5.2.4 Степенная функция

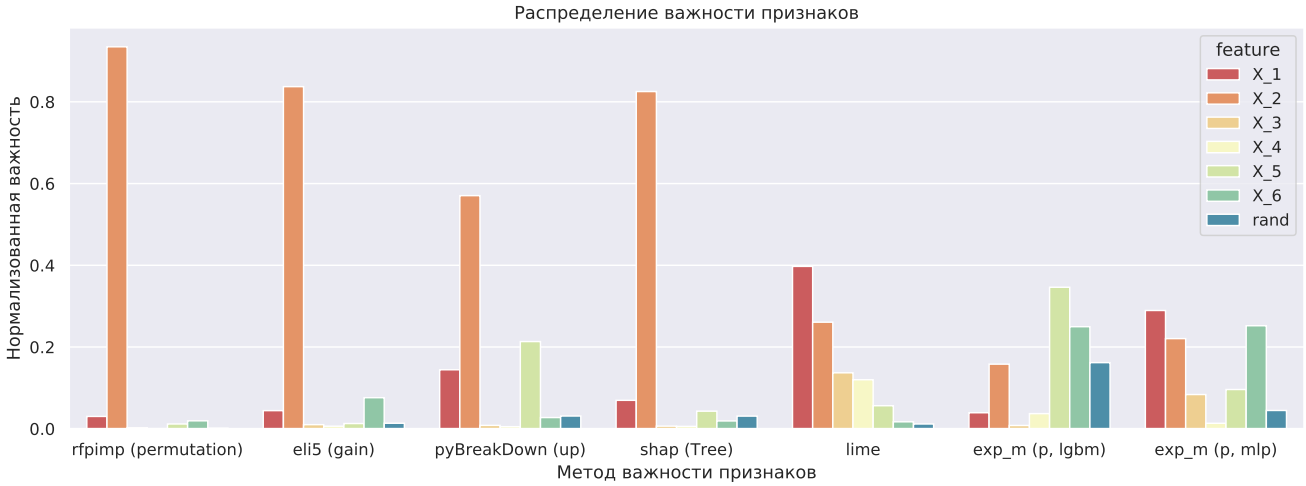


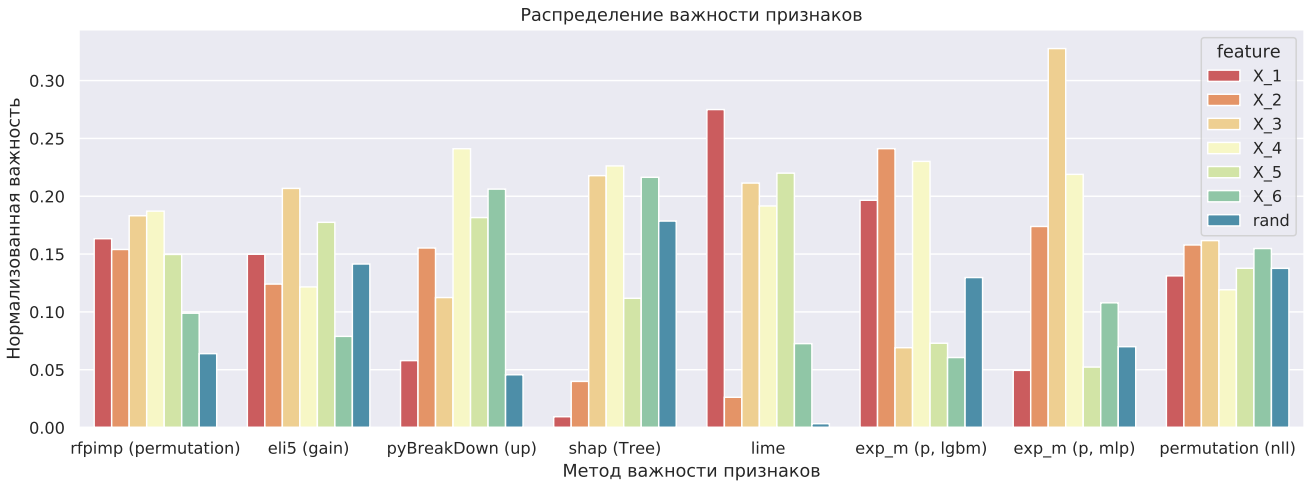
Рис. 12: $y = |X_1 + 10|^{(X_2+10)}$

Lime плохо обобщает из–за предположения независимости признаков. В большинстве случаев признак при степени более важен. Важность с использованием объясняющей модели показала плохие результаты. Большую роль играет сложность объясняющей модели. В данном примере ее оказалось недостаточно.

5.2.5 Сумма по модулю 2

Здесь все признаки одинаково важны и задаются следующим образом:

$$X_1, \ldots, X_6, X_7 \sim Bi(1, 0.5)$$
$$y = \text{Xor}\,(X_1, \ldots, X_6)$$



Shap дает большую важность случайному признаку, так как в большом количестве подмножеств признаков является важным. Это может свидетельствовать о переобучении модели. Аналогичные рассуждения применимы и для permutation (nll).

Если сравнивать оценки важностей признаков с оценкой для случайного признака, то перестановочная важность (f1) показала лучшие результаты.

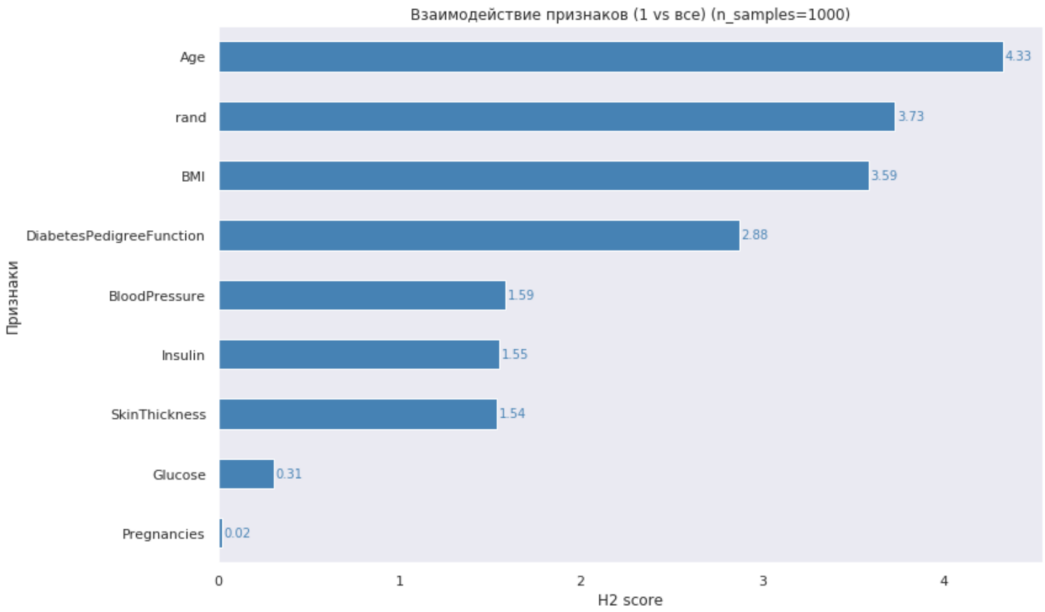
5.3. Прогноз диабета

5.3.1 Данные

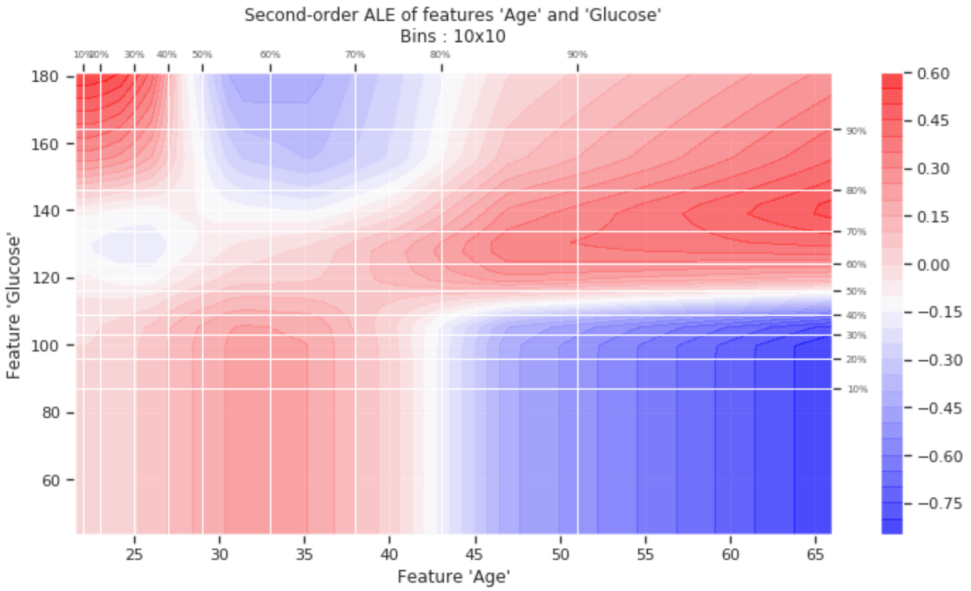
Датасет был взят с сайта kaggle. Признаки состоят из наиболее связанных с целевой меткой параметров: уровень глюкозы (Glucose), уровень инсулина (Insulin), функция родословного диабета (DiabetesPedigreeFunction) и так далее. Всего 9 штук. В качестве модели использовались бустинг (LightGBM) и метод опорных векторов (SVC).

Повторим эксперименты секции 5.1.

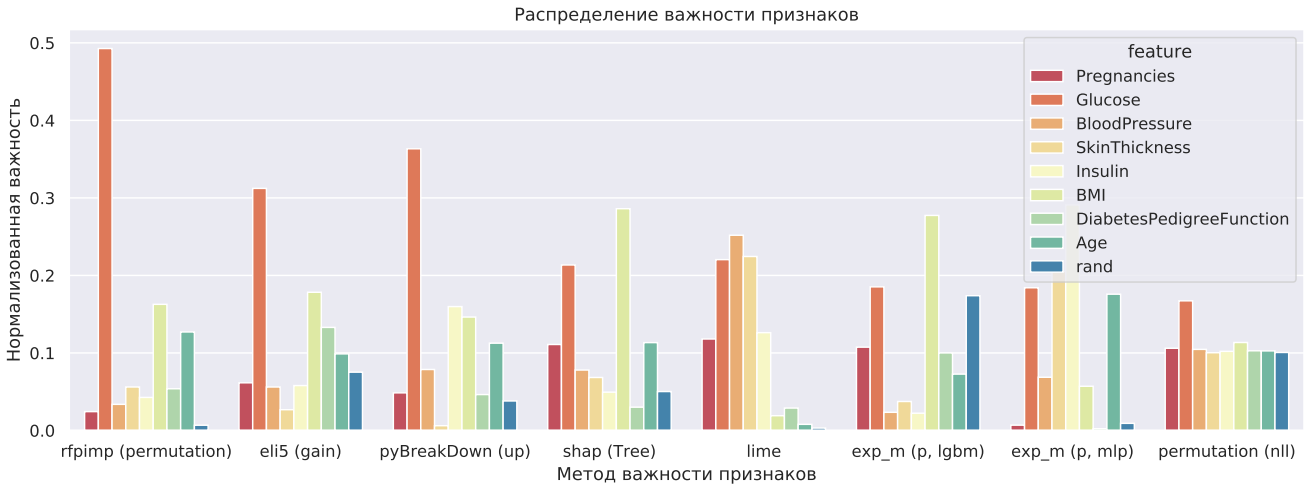
5.3.2 Важность



Логично увидеть возраст на первом месте.

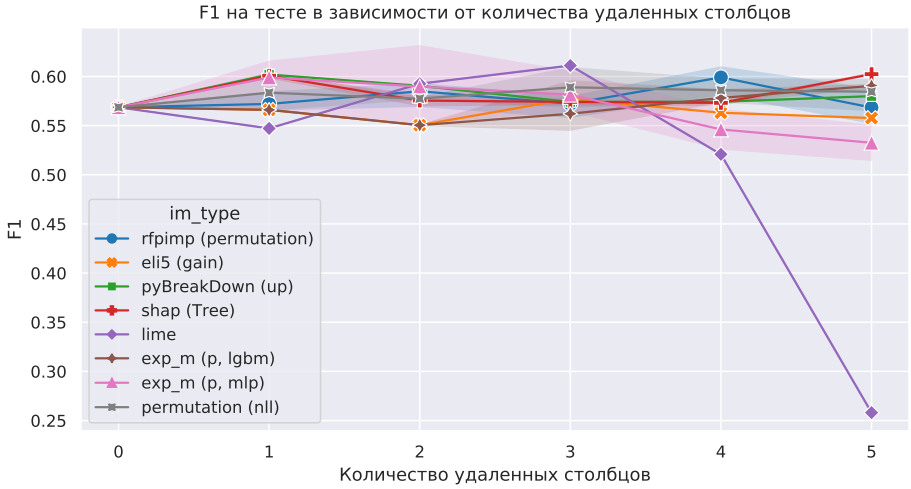


После 44 лет есть четкое пороговое значение по которому можно судить о диабете. При старении становится более важным соблюдать диету, так как небольшие отклонения приводят к резкому повышению вероятности его появления или наоборот.



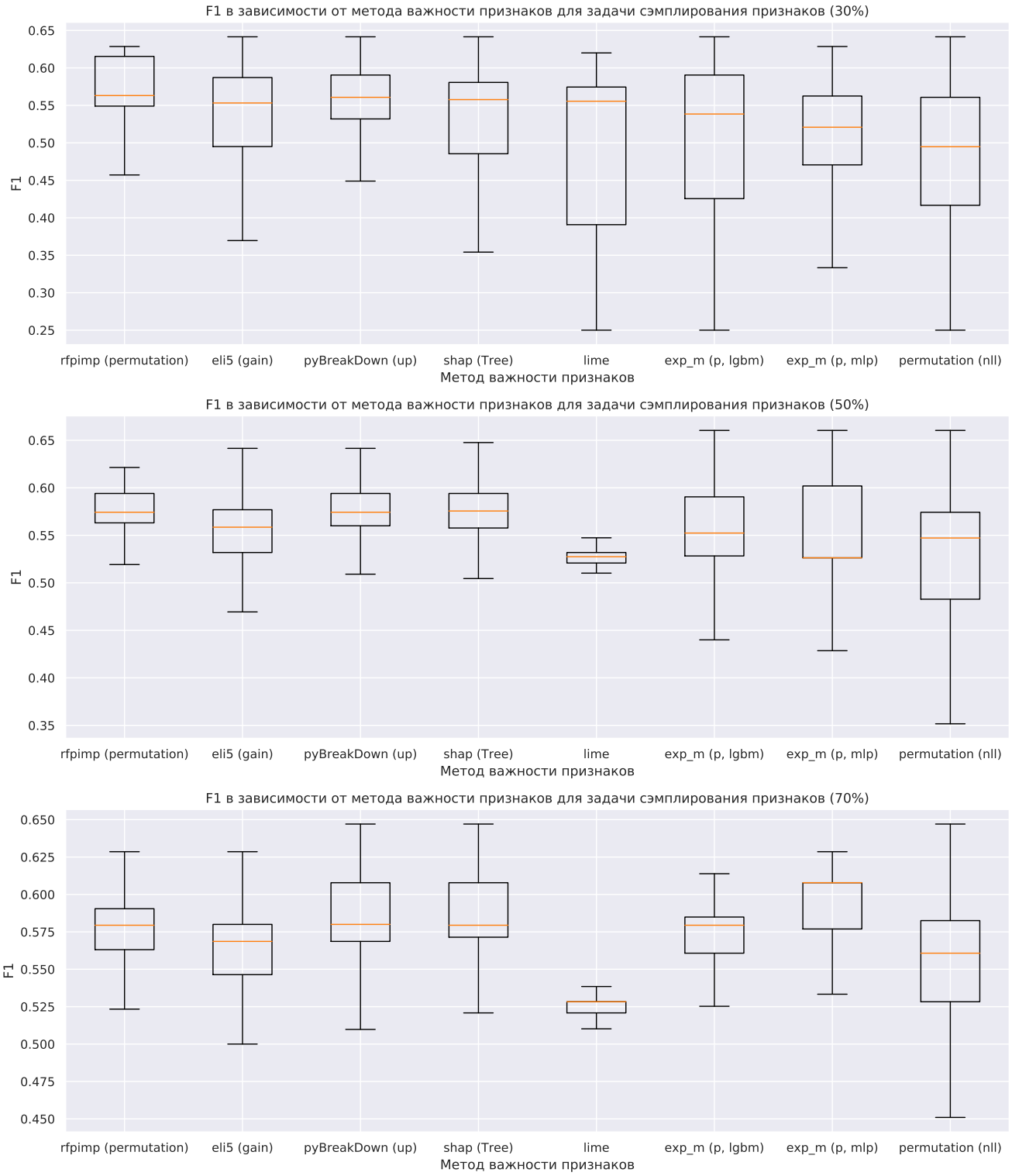
В большинстве случаев глюкоза и индекс массы тела влияют на наличие диабета. Gain показало высокую важность для случайного признака. Большое число уникальных значений приводит к искусственному увеличению значимости. Permutation (nll) имеет скорее всего неинформативные значения.

5.3.3 Удаление признаков



На реальных данных все методы, за исключением lime, работают приблизительно одинаково.

5.3.4 Сэмплирование признаков



Перестановочная важность лучше всех выделяет признаки, которые в общем нужны для решения задачи (30% от общего количества). При большом количестве признаков объясняющая модель на основе многослойного перцептрона (MLP) дала лучшие результаты. За исключением lime, признаки, выбранные алгоритмами, дают в среднем одинаковое качество.

§6. Заключение

В данной работе рассмотрены различные методы определения оценки важности признаков. Проведены эксперименты на искусственных и реальных данных, показывающие положительные и отрицательные стороны. Можно сделать следующие выводы:

- Прежде чем работать с данными их нужно отфильтровать: кластеризовать похожие признаки и оставить один из них.
- Перестановочная важность — первый метод, который может стать вашим инструментом в поиске важных для задачи признаков, из-за его простоты в применении и относительно небольшой сложности вычисления. Выбор ошибки сильно влияет на его значения.
- Если вы интересуетесь нахождением с большой вероятностью важных признаков в малом количестве, то pyBreakDown является хорошим вариантом.
- В эксперименте с сэмплированием признаков подход с объясняющей моделью дал лучшие результаты. Однако для релевантных интерпретаций поиск хорошей архитектуры занимает большую часть времени. Для быстроты работы необходима gpu.
- Локальные методы, например lime, плохо обобщают информацию на всю выборку данных.
- Взаимодействие признака с другими не показало хороший результат, как оценка важности признака.

§7. Список литературы

[1] Datta A., Sen S., Zick Y. Algorithmic transparency via quantitative input influence: Theory and experiments with learning systems // IEEE symposium on security and privacy (SP). — 2016. — Vol. 48. — P. 598–617.

[2] Fisher A., Rudin C., Dominici F. All models are wrong but many are useful: variable importance for black-box, proprietary, or misspecified prediction models, using model class reliance // arXiv preprint arXiv:1801.01489. — 2018.

[3] Frénay B., Doquire G., Verleysen M. Is mutual information adequate for feature selection in regression? // Neural Networks. — 2013. — Vol. 48. — P. 1–7.

[4] Gimenez J. R., Ghorbani A., Zou J. Knockoffs for the mass: new feature importance statistics with false discovery guarantees // The 22nd International Conference on Artificial Intelligence and Statistics. — 2019. — P. 2125–2133.

[5] Gregorutti B., Michel B., Saint-Pierre P. Grouped variable importance with random forests and application to multiple functional data analysis // Computational Statistics Data Analysis. — 2015. — Vol. 90. — P. 15–35.

[6] Gregorutti B., Michel B., Saint-Pierre P. Correlation and variable importance in random forests // Statistics and Computing. — 2017. — Vol. 27, no. 3. — P. 659–678.

[7] Louppe G. Understanding variable importances in forests of randomized trees // Advances in Neural Information Processing Systems 26. — 2013. — P. 26.

[8] Lundberg S., Lee S. A unified approach to interpreting model predictions // arXiv preprint arXiv:1705.07874. — 2017.

[9] Molnar C. Interpretable Machine Learning. — GitHub. — 2021. — Access mode: https://christophm.github.io/interpretable-ml-book/.

[10] Ribeiro M. T. lime. — GitHub. — 2021. — Access mode: https://github.com/marcotcr/lime.

[11] Ribeiro M. T., Singh S., Guestrin C. "Why should i trust you?"Explaining the predictions of any classifier // Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining. — 2016. — P. 1135–1144.

[12] Schwab P., Karlen W. Cxplain: Causal explanations for model interpretation under uncertainty // arXiv preprint arXiv:1910.12336. — 2019.

[13] Shrikumar A., Greenside P., Kundaje A. Learning important features through propagating activation differences // International Conference on Machine Learning. - PMLR. — 2017. — P. 3145–3153.

[14] Staniak M., Biecek P. Explanations of model predictions with live and breakDown packages // arXiv preprint arXiv:1804.01955. — 2018.

[15] Strobl C. Bias in random forest variable importance measures: Illustrations, sources and a solution // BMC bioinformatics. — 2007. — Vol. 8, no. 1. — P. 1–21.

[16] Strumbelj E., Kononenko I. An efficient explanation of individual classifications using game theory // The Journal of Machine Learning Research. — 2010. — Vol. 11. — P. 1–18.

[17] Sutera A. Context-dependent feature analysis with random forests // arXiv preprint arXiv:1605.03848. — 2016.

[18] Thiagarajan J. J. Accurate and Robust Feature Importance Estimation under Distribution Shifts // arXiv preprint arXiv:2009.14454. — 2020.

[19] Wojtas M., Chen K. Feature Importance Ranking for Deep Learning // arXiv preprint arXiv:2010.08973. — 2020.