МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ РАДИОФИЗИКИ И КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

Кафедра радиофизики и цифровых медиа технологий

Методы анализа влияния входных признаков моделей машинного обучения с учителем

Курсовая работа

Раковца Андрея Владимировича

студента 3 курса, специальность

«прикладная информатика»

Научный руководитель:

старший преподаватель

Курочкин А.В.

Минск,2020

Содержание:

Введение

Одна из фундаментальных проблем машинного обучения в том, что обученные модели работают как "черный ящик". Машинное обучение в целом сводится к тому, что мы какую-то очень гибкую и настраиваемую функцию подгоняем под такой вид, который хорошо описывает какие-то имеющиеся данные. Например, мы знаем, как формируется нейронная сеть, знаем, как происходит её обучение, но после того, как она обучена, её поведение очень сложно интерпретировать - она может имеющиеся у нас данные описывать просто идеально, но мы всё равно не будем знать, какие именно принципы лежат в полученном функциональном преобразовании. Т.е. мы знаем выходной результат, но мы не знаем, почему мы получили именно такой результат, а не какой-то другой.

Цель: исследовать возможность интерпретации поведения обученных моделей машинного обучения с учителем на основе методов анализа влияния входных признаков на результат

Задачи:

- рассмотреть проблему "черного ящика" при построении моделей машинного обучения с учителем,

- рассмотреть подход к анализу значимости признаков в моделях на основе деревьев принятия решений,

- рассмотреть задачу определения значимости признаков в контексте понижения размерности,

- исследовать рассмотренные алгоритмы на устойчивость к шумовым признакам с различными видами распределений

ГЛАВА 1. МОДЕЛИ С РЕШЕННОЙ ЗАДАЧЕЙ ОЦЕНКИ ЗНАЧИМОСТИ ПРИЗНАКОВ

1.1 Бинарные деревья

Рассмотрим бинарное дерево, в котором:

- каждой внутренней вершине v приписана функция (или предикат) $\beta_v : X \to \{0, 1\}$;
- каждой листовой вершине v приписан прогноз $c_v \in Y$ (в случае с классификацией листу также может быть приписан вектор вероятностей).

Рассмотрим теперь алгоритм a(x), который стартует из корневой вершины v_0 и вычисляет значение функции β_{v0}. Если оно равно нулю, то алгоритм переходит в левую дочернюю вершину, иначе в правую, вычисляет значение предиката в новой вершине и делает переход или влево, или вправо. Процесс продолжается, пока не будет достигнута листовая вершина; алгоритм возвращает тот класс, который приписан этой вершине. Такой алгоритм называется бинарным решающим деревом.

На практике в большинстве случаев используются одномерные предикаты β_v, которые сравнивают значение одного из признаков с порогом:

На практике в большинстве случаев используются одномерные предикаты β_v, которые сравнивают значение одного из признаков с порогом:

$$\beta_v(x; j, t) = [x_j < t].$$

Существуют и многомерные предикаты, например:

- линейные $\beta_v(x) = [\langle w, x \rangle < t]$;
- метрические $\beta_v(x) = [\rho(x, x_v) < t]$, где точка x_v является одним из объектов выборки любой точкой признакового пространства.

Многомерные предикаты позволяют строить ещё более сложные разделяющие поверхности, но очень редко используются на практике — например, из-за того, что усиливают и без того выдающиеся способности деревьев к переобучению. Далее мы будем говорить только об одномерных предикатах.

Легко убедиться, что для любой выборки можно построить решающее дерево, не допускающее на ней ни одной ошибки — даже с простыми одномерными предикатами можно сформировать дерево, в каждом листе

которого находится ровно по одному объекту выборки. Скорее всего, это дерево будет переобученным и не сможет показать хорошее качество на новых данных. Можно было бы поставить задачу поиска дерева, которое является минимальным (с точки зрения количества листьев) среди всех деревьев, не допускающих ошибок на обучении — в этом случае можно было бы надеяться на наличие у дерева обобщающей способности. К сожалению, эта задача является NP-полной, и поэтому приходится ограничиваться жадными алгоритмами построения дерева.

Опишем базовый жадный алгоритм построения бинарного решающего дерева. Начнем со всей обучающей выборки X и найдем наилучшее ее разбиение на две части $R_1(j, t) = \{x \mid x_j < t\}$ и $R_2(j, t) = \{x \mid x_j \geq t\}$ с точки зрения заранее заданного функционала качества $Q(X, j, t)$. Найдя наилучшие значения j и t, создадим корневую вершину дерева, поставив ей в соответствие предикат $[x_j < t]$. Объекты разобьются на две части — одни попадут в левое поддерево, другие в правое. Для каждой из этих подвыборок рекурсивно повторим процедуру, построив дочерние вершины для корневой, и так далее. В каждой вершине мы проверяем, не выполнилось ли некоторое условие останова — и если выполнилось, то прекращаем рекурсию и объявляем эту вершину листом. Когда дерево построено, каждому листу ставится в соответствие ответ. В случае с классификацией это может быть класс, к которому относится больше всего объектов в листе, или вектор вероятностей (скажем, вероятность класса может быть равна доле его объектов в листе). Для регрессии это может быть среднее значение, медиана или другая функция от целевых переменных объектов в листе. Выбор конкретной функции зависит от функционала качества в исходной задаче.

Решающие деревья могут обрабатывать пропущенные значения — ситуации, в которых для некоторых объектов неизвестны значения одного или нескольких признаков. Для этого необходимо модифицировать процедуру разбиения выборки в вершине, что можно сделать несколькими способами.

После того, как дерево построено, можно провести его стрижку (pruning) — удаление некоторых вершин с целью понижения сложности и повышения обобщающей способности. Существует несколько подходов к стрижке, о которых мы немного упомянем ниже.

Таким образом, конкретный метод построения решающего дерева определяется:

1. Видом предикатов в вершинах;

2. Функционалом качества $Q(X, j, t)$;

3. Критерием останова;

4. Методом обработки пропущенных значений;

5. Методом стрижки.

Также могут иметь место различные расширения, связанные с учетом весов объектов, работой с категориальными признакам и т.д. Ниже мы обсудим варианты каждого из перечисленных пунктов.

При построении дерева необходимо задать функционал качества, на основе которого осуществляется разбиение выборки на каждом шаге. Обозначим через R_m множество объектов, попавших в вершину, разбиваемую на данном шаге, а множество объектов, попавших в вершину, разбиваемую на данном шаге, а через R_ℓ и R_r — объекты, попадающие в левое и правое поддерево соответственно при заданном предикате. Мы будем использовать функционалы следующего вида:

$$Q(R_m, j, s) = H(R_m) - \frac{|R_l|}{|R_m|} H(R_l) - \frac{|R_r|}{|R_m|} H(R_r) \qquad (1.1)$$

Здесь $H(R)$ — это критерий информативности (impurity criterion), который оценивает качество распределения целевой переменной среди объектов множества R. Чем меньше разнообразие целевой переменной, тем меньше должно быть значение критерия информативности — и, соответственно, мы будем пытаться минимизировать его значение. Функционал качества $Q(R_m, j, s)$ мы при этом будем максимизировать. Как уже обсуждалось выше, в каждом листе дерево будет выдавать константу — вещественное число, вероятность или класс. Исходя из этого, можно предложить оценивать качество множества объектов R тем, насколько хорошо их целевые переменные предсказываются константой (при оптимальном выборе этой константы):

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} L(y_i, c) \qquad (1.2)$$

где L (y, c) — некоторая функция потерь. Далее мы обсудим, какие именно критерии информативности часто используют в задачах регрессии и классификации.

1.1.1 Регрессия

Как обычно, в регрессии выберем квадрат отклонения в качестве функции потерь. В этом случае критерий информативности будет выглядеть как

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} (y_i - c)^2 \quad (1.3)$$

Как известно, минимум в этом выражении будет достигаться на среднем значении целевой переменной. Значит, критерий можно переписать в следующем виде:

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \left(y_i - \frac{1}{|R|} \sum_{(x_i, y_i) \in R} y_i \right)^2 \quad (1.4)$$

Мы получили, что информативность вершины измеряется её дисперсией — чем ниже разброс целевой переменной, тем лучше вершина. Разумеется, можно использовать и другие функции ошибки L — например, при выборе абсолютного отклонения мы получим в качестве критерия среднее абсолютное отклонение от медианы.

1.1.2 Классификация

Обозначим через p_k долю объектов класса k ($k \in \{1, \ldots, K\}$), попавших в вершину R:

$$p_k = \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i = k]. \quad (1.5)$$

Через k_* обозначим класс, чьих представителей оказалось больше всего среди объектов, попавших в данную вершину: $k_* = arg \max_k p_k$

1.1.2.1 Ошибка классификации

Рассмотрим индикатор ошибки как функцию потерь:

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i \neq c]. \quad (1.6)$$

Легко видеть, что оптимальным предсказанием тут будет наиболее популярный класс k_* — значит, критерий будет равен следующей доле ошибок:

$$H(R) = \min_{c \in Y} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} [y_i \neq k_*] = 1 - p_k. \quad (1.7)$$

Данный критерий является достаточно грубым, поскольку учитывает частоту p_{k*} лишь одного класса.

1.1.2.2 Критерий Джини

Рассмотрим ситуацию, в которой мы выдаем вершине не один класс, а распределение на всех классах $c = (c_1, \dots, c_k), \sum_{k=1}^{K} c_k = 1$ Качество такого распределения можно измерять, например, с помощью критерия Бриера (Brier score):

$$H(R) = \min_{\sum_k c_k = 1} \frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^{K} (c_k - [y_i = k])^2 \quad (1.8)$$

Можно показать, что оптимальный вектор вероятностей состоит из долей классов p_k:

$$c_* = (p_1, \dots, p_k)$$

Если подставить эти вероятности в исходный критерий информативности и провести ряд преобразований, то мы получим критерий Джини:

$$H(R) = \sum_{k=1}^{K} p_k (1 - p_k) \quad (1.9)$$

1.1.2.3 Энтропийный критерий

Мы уже знакомы с более популярным способом оценивания качества вероятностей — логарифмическими потерями, или логарифмом правдоподобия:

$$H(R) = \min_{\sum_k c_k} \left(-\frac{1}{|R|} \sum_{(x_i, y_i) \in R} \sum_{k=1}^{K} [y_i = k] log c_k \right) \quad (1.10)$$

Для вывода оптимальных значений c_k вспомним, что все значения c_k должны суммироваться в единицу. Как известного из методов оптимизации, для учёта этого ограничения необходимо искать минимум лагранжиана:

$$L(c,\lambda) = -\frac{1}{|R|}\sum_{(x_i,y_i)\in R}\sum_{k=1}^{K}[y_i = k]logc_k + \lambda\sum_{k=1}^{K}c_k \rightarrow \min_{c_k} \quad (1.11)$$

Дифференцируя, получаем:

$$\frac{\partial}{\partial c_k}L(c,\lambda) = -\frac{1}{|R|}\sum_{(x_i,y_i)\in R}[y_i = k]\frac{1}{c_k} + \lambda = -\frac{p_k}{c_k} + \lambda = 0 \quad (1.12)$$

откуда выражаем $c_k = p_k/\lambda$. Суммируя эти равенства по k, получим

$$1 = \sum_{k=1}^{K}c_k = \frac{1}{\lambda}\sum_{k=1}^{K}p_k = \frac{1}{\lambda}. \quad (1.13)$$

откуда $\lambda = 1$. Значит, минимум достигается при $c_k = p_k$, как и в предыдущем случае. Подставляя эти выражения в критерий, получим, что он будет представлять собой энтропию распределения классов:

$$H(R) = -\sum_{k=1}^{K}p_k \log(p_k). \quad (1.14)$$

Из теории вероятностей известно, что энтропия ограничена снизу нулем, причем минимум достигается на вырожденных распределениях ($p_i = 1$, $p_j = 0$ для $i \neq j$). Максимальное же значение энтропия принимает для равномерного распределения. Отсюда видно, что энтропийный критерий отдает предпочтение более «вырожденным» распределениям классов в вершине.

1.1.3 Критерии останова

Можно придумать большое количестве критериев останова. Перечислим некоторые ограничения и критерии:

- Ограничение максимальной глубины дерева.
- Ограничение минимального числа объектов в листе.
- Ограничение максимального количества листьев в дереве.
- Останов в случае, если все объекты в листе относятся к одному классу.

• Требование, что функционал качества при дроблении улучшался как минимум на s процентов.

С помощью грамотного выбора подобных критериев и их параметров можно существенно повлиять на качество дерева. Тем не менее, такой подбор является трудозатратным и требует проведения кросс-валидации.

1.1.4 Методы стрижки дерева

Стрижка дерева является альтернативой критериям останова, описанным выше. При использовании стрижки сначала строится переобученное дерево (например, до тех пор, пока в каждом листе не окажется по одному объекту), а затем производится оптимизация его структуры с целью улучшения обобщающей способности. Существует ряд исследований, показывающих, что стрижка позволяет достичь лучшего качества по сравнению с ранним остановом построения дерева на основе различных критериев. Тем не менее, на данный момент методы стрижки редко используются и не реализованы в большинстве библиотек для анализа данных. Причина заключается в том, что деревья сами по себе являются слабыми алгоритмами и не представляют большого интереса, а при использовании в композициях они либо должны быть переобучены (в случайных лесах), либо должны иметь очень небольшую глубину (в бустинге), из-за чего необходимость в стрижке отпадает. Одним из методов стрижки является cost-complexity pruning. Обозначим дерево, полученное в результате работы жадного алгоритма, через T_0. Поскольку в каждом из листьев находятся объекты только одного класса, значение функционала $R(T)$ будет минимально на самом дереве T_0 (среди всех поддеревьев). Однако данный функционал характеризует лишь качество дерева на обучающей выборке, и чрезмерная подгонка под нее может привести к переобучению. Чтобы преодолеть эту проблему, введем новый функционал $R_\alpha(T)$, представляющий собой сумму исходного функционала $R(T)$ и штрафа за размер дерева:

$$R_\alpha(T) = R(T) + \alpha|T|, \tag{1.15}$$

где $|T|$ — число листьев в поддереве T, а $\alpha \geq 0$ — параметр. Это один из примеров регуляризованных критериев качества, которые ищут баланс между качеством классификации обучающей выборки и сложностью построенной модели. Можно показать, что существует последовательность вложенных деревьев с одинаковыми корнями:

$$T_k \subset T_{k-1} \subset \cdots \subset T_0,$$

(здесь T_k — тривиальное дерево, состоящее из корня дерева T_0), в которой каждое дерево T_i минимизирует критерий (1) для α из интервала $\alpha \in [a_i, \alpha_{i+1})$, причем

$$0 < \alpha_0 < \alpha_1 < \cdots < a_k < \infty.$$

Эту последовательность можно достаточно эффективно найти путем обхода дерева. Далее из нее выбирается оптимальное дерево по отложенной выборке или с помощью кросс-валидации.

1.1.5 Обработка пропущенных значений

Одним из основных преимуществ решающих деревьев является возможность работы с пропущенными значениями. Рассмотрим некоторые варианты. Пусть нам нужно вычислить функционал качества для предиката $\beta(x) = [x_j < t]$, но в выборке R для некоторых объектов не известно значение признака j — обозначим их через V_j . В таком случае при вычислении функционала можно просто проигнорировать эти объекты, сделав поправку на потерю информации от этого:

$$Q(R, j, s) \approx \frac{\left|R \setminus V_j\right|}{|R|} Q\left(R \setminus V_j, j, s\right) \tag{1.16}$$

Затем, если данный предикат окажется лучшим, поместим объекты из V_j как в левое, так и в правое поддерево. Также можно присвоить им при этом веса $\frac{|R_l|}{|R|}$ в левом поддереве и $\frac{|Rr|}{|R|}$ в правом. В дальнейшем веса можно учитывать, добавляя их как коэффициенты перед индикаторами $[y_i = k]$ во всех формулах. На этапе применения дерева необходимо выполнять похожий трюк. Если объект попал в вершину, предикат которой не может быть вычислен из-за пропуска, то прогнозы для него вычисляются в обоих поддеревьях, и затем усредняются с весами, пропорциональными числу обучающих объектов в этих поддеревьях. Иными словами, если прогноз вероятности для класса k в поддереве Rm обозначается через $a_{mk}(x)$, то получаем такую формулу:

$$a_{mk}(x) = \begin{cases} a_{lk}(x), & \beta_m = 0; \\ a_{rk}(x), & \beta_m(x) = 1; \\ \dfrac{|R_l|}{|R_m|} a_{lk}(x) + \dfrac{|R_r|}{|R_m|} a_{rk}(x), & \beta_m \text{ нельзя вычислить.} \end{cases} \tag{1.17}$$

Другой подход заключается в построении суррогатных предикатов в каждой вершине. Так называется предикат, который использует другой признак, но при этом дает разбиение, максимально близкое к данному. Отметим, что нередко схожее качество показывают и гораздо более простые способы

обработки пропусков — например, можно заменить все пропуски на ноль. Для деревьев также разумно будет заменить пропуски в признаке на числа, которые превосходят любое значение данного признака. В этом случае в дереве можно будет выбрать такое разбиение по этому признаку, что все объекты с известными значениями пойдут в левое поддерево, а все объекты с пропусками — в правое.

1.1.6 Учет категориальных признаков

Самый очевидный способ обработки категориальных признаков — разбивать вершину на столько поддеревьев, сколько имеется возможных значений у признака (multi-way splits).

Рассмотрим подробнее другой подход. Пусть категориальный признак xj имеет множество значений $Q = \{u_1, \ldots, u_q\}$, $|Q| = q$. Разобьем множество значений на два непересекающихся подмножества: $Q = Q_1 \sqcup Q_2$, и определим предикат как индикатор попадания в первое подмножество: $\beta(x) = [x_j \in Q_1]$. Таким образом, объект будет попадать в левое поддерево, если признак x_j попадает в множество Q1, и в первое поддерево в противном случае. Основная проблема заключается в том, что для построения оптимального предиката нужно перебрать $2^{q-1} - 1$ вариантов разбиения, что может быть не вполне возможным.

Оказывается, можно обойтись без полного перебора в случаях с бинарной классификацией и регрессией [1]. Обозначим через $R_m(u)$ множество объектов, которые попали в вершину m и у которых j-й признак имеет значение u; через $N_m(u)$ обозначим количество таких объектов. В случае с бинарной классификацией упорядочим все значения категориального признака на основе того, какая доля объектов с таким значением имеет класс +1:

$$\frac{1}{N_m(u_{(1)})} \sum_{x_i \in R_m(u_{(1)})} [y_i = +1] \leq \cdots \leq \frac{1}{N_m(u_{(q)})} \sum_{x_i \in R_m(u_{(q)})} [y_i = +1], \qquad (1.18)$$

после чего заменим категорию $u_{(i)}$ на число i, и будем искать разбиение как для вещественного признака. Можно показать, что если искать оптимальное разбиение по критерию Джини или энтропийному критерию, то мы получим такое же разбиение, как и при переборе по всем возможным $2^{q-1} - 1$ вариантам. [7,12]

1.2 Случайные леса

Ансамблевые методы — это парадигма машинного обучения, где несколько моделей (часто называемых «слабыми учениками») обучаются для решения одной и той же проблемы и объединяются для получения лучших

результатов. Основная гипотеза состоит в том, что при правильном сочетании слабых моделей мы можем получить более точные и/или надежные модели.

1.2.1 Один слабый ученик

В машинном обучении, независимо от того, сталкиваемся ли мы с проблемой классификации или регрессии, выбор модели чрезвычайно важен, чтобы иметь какие-либо шансы получить хорошие результаты. Этот выбор может зависеть от многих переменных задачи: количества данных, размерности пространства, гипотезы распределения…

Слабое ***смещение (bias)*** и ***разброс (variance)*** модели, хотя они чаще всего изменяются в противоположных направлениях, являются двумя наиболее фундаментальными особенностями, ожидаемыми для модели. Действительно, чтобы иметь возможность «решить» проблему, мы хотим, чтобы в нашей модели было достаточно степеней свободы для разрешения базовой сложности данных, с которыми мы работаем, но мы также хотим, чтобы у нее было не слишком много степеней свободы, чтобы избежать ее высокого разброса и быть более устойчивой. Это хорошо известный **компромисс между смещением и разбросом**.

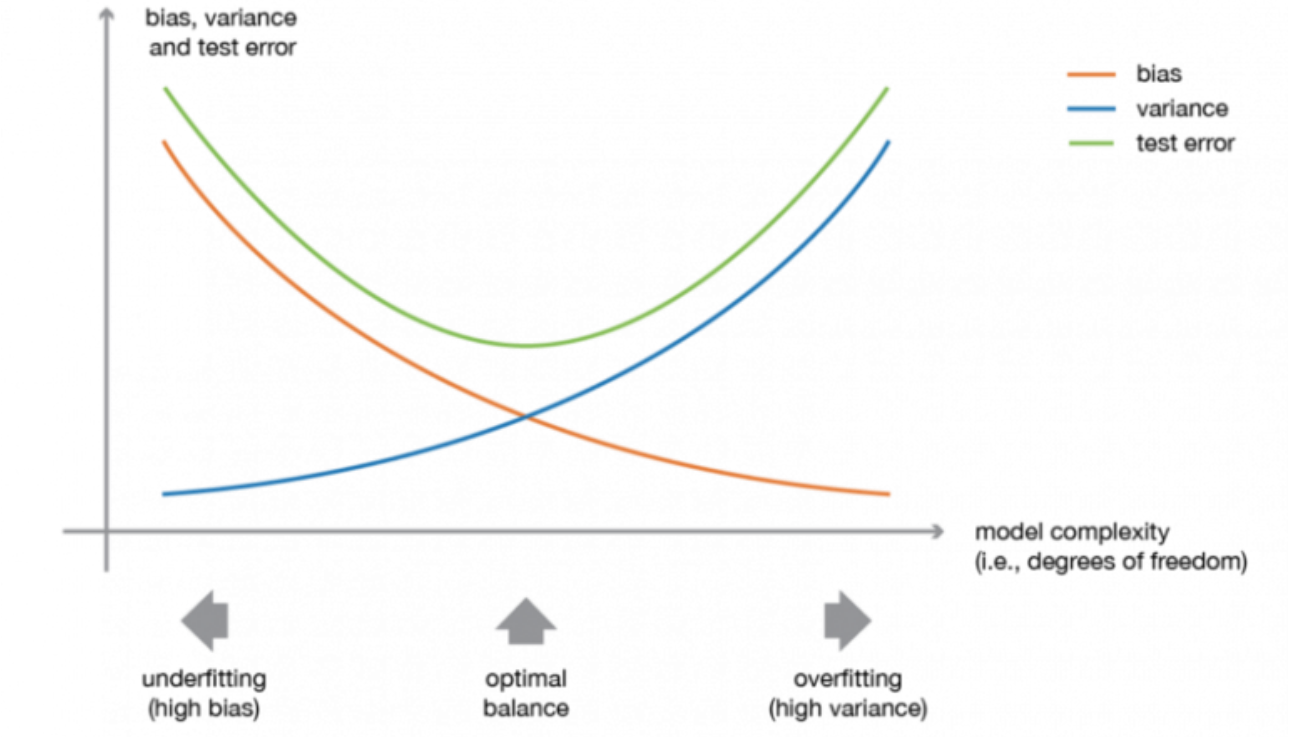

Рисунок 1. - Иллюстрация компромисса между смещением и разбросом

В ансамблевой теории обучения мы вводим понятия ***слабых учеников*** (или ***базовых моделей***), которых можно использовать в качестве строительных блоков для проектирования более сложных моделей путем объединения нескольких из них. В большинстве случаев эти базовые модели работают сами по себе не так хорошо в связи с тем, что они имеют высокое смещение (например, модели с низкой степенью свободы), либо с тем, что имеют слишком большой разброс, чтобы быть устойчивыми (например, модели с высокой степенью свободы). Тогда идея ансамблевых методов состоит в том, чтобы попытаться уменьшить смещение и/или разброс таких слабых учеников, объединяя несколько из них вместе, чтобы создать ***сильного ученика*** (или ***модель ансамбля***), который достигает лучших результатов.

1.2.2 Объединение слабых учеников

Чтобы реализовать ансамблевый метод, нам сначала нужно отобрать наших слабых учеников для агрегирования. В основном (в том числе в хорошо известных методах бэггинга и бустинга) используется единственный базовый алгоритм обучения, так что у нас есть однородные слабые ученики, которые обучаются по-разному. Получаемая нами модель ансамбля называется «***однородной***». Тем не менее, существуют также некоторые методы, которые используют различные типы базовых алгоритмов обучения: некоторые ***разнородные слабые ученики*** затем объединяются в «***разнородную ансамблевую модель***».

- **Бэггинг.** В этом случае часто рассматривают однородных слабых учеников, обучают их параллельно и независимо друг от друга, а затем объединяют их, следуя некоторому детерминированному процессу усреднения.

Слабых учеников можно объединить, чтобы получить модель с лучшими показателями. Способ объединения базовых моделей должен быть адаптирован к их типам. Модели с низким смещением и высоким разбросом следует объединять таким образом, чтобы сделать сильную модель более устойчивой, тогда как модели с низким разбросом и высоким смещением лучше объединять таким образом, чтобы сделать ансамблевую модель менее смещенной.

1.2.3 Бутстрэп

Давайте начнем с определения бутстрэпа. Этот статистический метод заключается в генерации выборок размера B (так называемых бутстрэп выборок)

из исходного датасета размера N путем случайного выбора элементов с повторениями в каждом из наблюдений B.

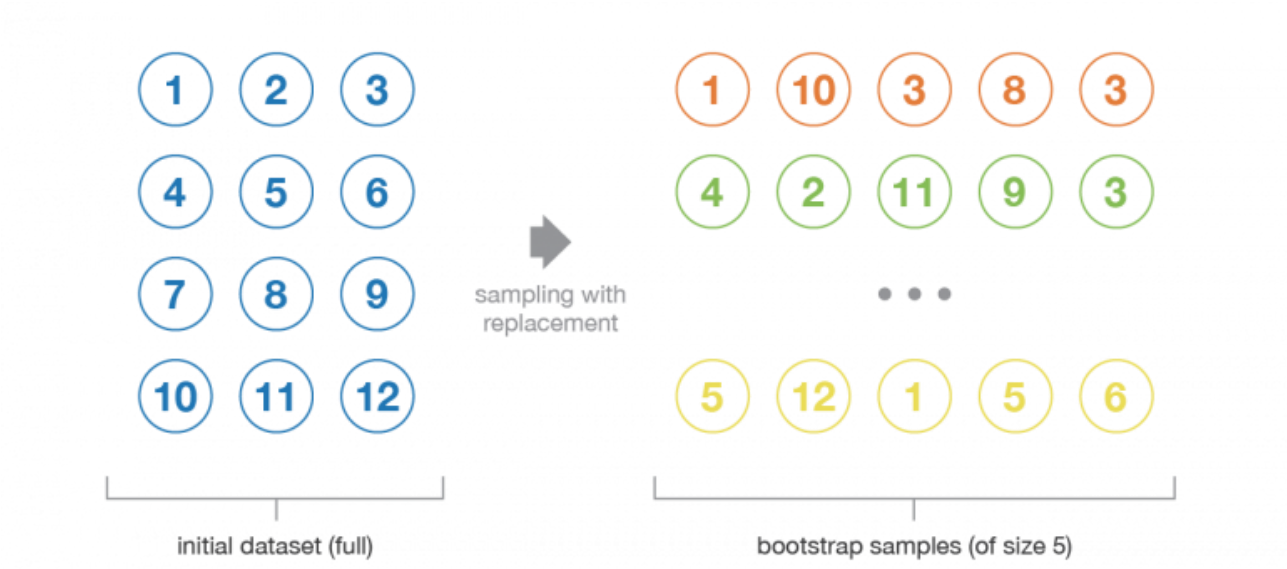

Рисунок 2. - Иллюстрация процесса бустрэпа

При некоторых допущениях эти выборки имеют довольно хорошие статистические свойства: в первом приближении их можно рассматривать как взятые непосредственно из истинного базового (и часто неизвестного) распределения данных, так и независимо друг от друга. Таким образом, их можно рассматривать как репрезентативные и независимые выборки истинного распределения данных (почти идентичные выборки). Гипотеза, которая должна быть проверена, чтобы сделать это приближение действительным, имеет две стороны. Во-первых, размер N исходного датасета должен быть достаточно большим, чтобы охватить большую часть сложности базового распределения, чтобы выборка из датасета была хорошим приближением к выборке из реального распределения **(репрезентативность)**. Во-вторых, размер датасета N должен быть достаточно большим по сравнению с размером бутстрэп выборок B, чтобы выборки не слишком сильно коррелировали **(независимость)**. Обратите внимание, что в дальнейшем мы иногда будем ссылаться на эти свойства (репрезентативность и независимость) бутстрэп выборок: читатель всегда должен помнить, что **это только приближение**.

1.2.4 Бэггинг

При обучении модели, независимо от того, имеем ли мы дело с проблемой классификации или регрессии, мы получаем функцию, которая принимает входные данные, возвращает выходные данные и определяется в

отношении обучающего датасета. Из-за теоретического разброса обучающего датасета (мы напоминаем, что датасет является наблюдаемой выборкой, исходящей из истинно неизвестного базового распределения), подобранная модель также подвержена изменчивости: **если бы наблюдался другой датасет, мы получили бы другую модель**.

Идея бэггинга в таком случае проста: мы хотим подобрать несколько независимых моделей и «усреднить» их прогнозы, чтобы получить модель с меньшим разбросом. Однако на практике мы не можем подобрать полностью независимые модели, потому что для этого потребуется слишком много данных. Таким образом, мы полагаемся на хорошие «приблизительные свойства» бутстрэп выборок (репрезентативность и независимость) для подбора моделей, которые практически независимы.

Сначала мы генерируем несколько бутстрэп выборок так, чтобы каждая новая бутстрэп выборка выполняла роль (почти) еще одного независимого датасета, взятого из истинного распределения. Затем мы можем **обучить слабого ученика для каждой из этих выборок и, наконец, агрегировать их так, чтобы мы как бы «усреднили» их результаты** и, таким образом, получили модель ансамбля с разбросом меньшим, чем ее отдельные компоненты.

Итак, предположим, что у нас есть L бутстрап выборок (аппроксимации L независимых датасетов) размера B. Это обозначается:

$$\{z_1^1, z_2^1, \dots, z_B^1\}, \{z_1^2, z_2^2, \dots, z_B^2\}, \dots, \{z_1^L, z_2^L, \dots, z_B^L\}$$

$$z_b^L \equiv b - th\ observation\ of\ the\ l - th\ bootstrap\ sample$$

Мы можем обучить L почти независимых слабых учеников (по одному на каждый датасет): $\omega_1(.), \omega_2(.), \dots, \omega_L(.)$.

А затем объединим их некоторым процессом усреднения, чтобы получить модель ансамбля с меньшим разбросом. Например, мы можем определить нашу сильную модель так, чтобы

$$s_L(.) = \frac{1}{L}\sum_{l=1}^{L} w_l(.) \qquad \text{(simple average, for regression problem)}$$

$$s_L(.) = \arg\max_k [card(l|w_l(.) = k)] \qquad \text{(simple majority vote, for classification problem)}$$

Существует несколько возможных способов объединить несколько моделей, обученных параллельно. Для задачи регрессии выходные данные

отдельных моделей могут быть буквально усреднены для получения выходных данных модели ансамбля. Для задачи классификации класс, предсказываемый каждой моделью, можно рассматривать как голос, а класс, который получает большинство голосов, является ответом модели ансамбля (это называется **мажоритарным голосованием**). Что касается задачи классификации, мы также можем рассмотреть вероятности каждого класса, предсказываемые всеми моделями, усреднить эти вероятности и сохранить класс с самой высокой средней вероятностью (это называется **мягким голосованием**). Средние значения или голоса могут быть простыми или взвешенными, если будут использоваться любые соответствующие им веса.

1.2.5 Случайные леса

Деревья решений являются очень популярными базовыми моделями для ансамблевых методов. Сильные ученики, состоящие из нескольких деревьев решений, можно назвать «лесами». Деревья, составляющие лес, могут быть выбраны либо неглубокими (глубиной в несколько узлов), либо глубокими (глубиной в множество узлов, если не в полную глубину со всеми листьями). Неглубокие деревья имеют меньший разброс, но более высокое смещение, и тогда для них лучшим выбором станут **последовательные методы**. Глубокие деревья, с другой стороны, имеют низкое смещение, но высокий разброс и, таким образом, являются подходящим выбором для бэггинга, который в основном направлен на уменьшение разброса.

Случайный лес представляет собой метод бэггинга, где глубокие деревья, обученные на бутстрап выборках, объединяются для получения результата с более низким разбросом. Тем не менее, случайные леса также используют другой прием, чтобы несколько обученных деревьев были менее коррелированными друг с другом: при построении каждого дерева вместо выбора всех признаков из датасета для генерации бутстрэпа мы выбираем и сохраняем только случайное их подмножество для построения дерева (обычно одинаковое для всех бутстрэп выборок).

Выборка по признакам действительно приводит к тому, что все деревья не смотрят на одну и ту же информацию для принятия своих решений и, таким образом, уменьшают корреляцию между различными возвращаемыми выходными данными. Другое преимущество выборки по признакам заключается в том, что **она делает процесс принятия решений более устойчивым к отсутствующим данным**: значения наблюдения (из обучающего датасета или нет) с отсутствующими данными можно восстанавливать с помощью регрессии или классификации на основе деревьев, которые учитывают только те признаки, где данные не отсутствуют. Таким образом, алгоритм случайного леса сочетает в себе концепции бэггинга и выбора подпространства случайных объектов для создания более устойчивых моделей[4].

ГЛАВА 2. ОТБОР ПРИЗНАКОВ

2.1 Задача отбора признаков

Целевые признаки (feature), используемые для обучения модели, оказывают большое влияние на качество результатов. Неинформативные или слабо информативные признаки могут существенно понизить эффективность и точность многих моделей, особенно линейных, таких как линейная и логистическая регрессия. После устранения или преобразования неинформативных/слабо информативных признаков модель упрощается, и соответственно уменьшается размер набора данных в памяти и ускоряется работа алгоритмов ML на нем.

Можно выделить 3 задачи:

- **feature selection** – отсечение ненужных признаков (избыточных, слабо информативных) и выбор признаков, имеющих наиболее тесные взаимосвязи с целевой переменной.

- **feature extraction and feature engineering** – превращение данных, специфических для предметной области, в понятные для модели векторы;

- **feature transformation** – трансформация данных для повышения точности алгоритма (например, нормализация данных).

Каждая из этих задач направлена на обеспечение следующих преимуществ:

- **Уменьшение переобучения.** Чем меньше избыточных данных, тем меньше возможностей для модели принимать решения на основе «шума».

- **Повышение точности предсказания модели.** Чем меньше противоречивых данных, тем выше точность.

- **Сокращение времени обучения.** Чем меньше данных, тем быстрее обучается модель.

- **Увеличивается семантическое понимание модели.**

Выделяют методы ручного и автоматизированного отбора признаков.

Методы ручного отбора признаков основаны на содержательном анализе каждого признака и их совокупности, когда решение о включении/исключении признака из модели принимает исследователь.

В данном разделе рассмотрены различные методы автоматизированного отбора признаков (feature selection), применяемые для подготовки данных. Они могут быть реализованы с помощью Python и библиотеки scikit-learn. Подробное руководство по отбору признаков с помощью scikit-learn вы можете найти в документации к этой библиотеке в разделе Feature selection.

Методы отбора признаков делятся на три группы:

1) методы-фильтры (filters) - они оценивают признаки только на основе информации, полученной из обучающей выборки. Применяются на этапе предобработки, до запуска алгоритма обучения;

2) методы-обертки (wrappers) - классификатор запускается на конкретных подмножествах обучающей выборки, а затем выбирается подмножество наиболее информативных для обучения признаков;

Рисунок 1 - Процесс работы оберточных методов

3) встроенные методы (embedded) - позволяют не отделять отбор признаков и обучение классификатора. Данный тип методов также обладает рядом других преимуществ: они хорошо приспособлены к конкретной модели; не требуется выделять специальное подмножество для тестирования, как в предыдущих методах, и, как следствие из этого, меньше риск переобучения.

Рисунок 2 - Процесс работы встроенных методов

Рассмотрим подробнее каждую группу.

2.2 Методы – фильтры(filters)

Фильтры (англ. *filter methods*) измеряют релевантность признаков на основе функции μμ, и затем решают по определенному правилу, какие признаки оставить в результирующем множестве.

Фильтры могут быть:

- Одномерные (англ. *univariate*) — функция μμ определяет релевантность одного признака по отношению к целевой переменной. В таком случае обычно измеряют "качество" каждого признака с помощью, например, статистических критериев (коэффициент ранговой корреляции Спирмена, Information gain и др.) и удаляют худшие. Например, библиотека scikit-learn содержит класс SelectKBest, реализующий одномерный отбор признаков (univariate feature selection). Этот класс можно применять совместно с различными статистическими критериями для отбора заданного количества признаков.

- Многомерные (англ. *multivariate*) — функция μμ определяет релевантность некоторого подмножества исходного множества признаков относительно выходных меток.

Преимуществом группы фильтров является простота вычисления релевантности признаков в наборе данных, но недостатком в таком подходе является игнорирование возможных зависимостей между признаками.

2.3 Методы-обертки (Wrapper methods)

Принцип методов обертки состоит в следующем:

• выполняется поиск по пространству подмножеств исходного множества признаков;

• для каждого шага поиска используется информация о качестве обучения на текущем подмножестве признаков (в качестве функции оценки качества обучения на текущем подмножестве признаков часто используется точность классификатора).

Этот процесс является циклическим и продолжается до тех пор, пока не будут достигнуты заданные условия останова. Оберточные методы учитывают зависимости между признаками, что является преимуществом по сравнению с фильтрами, к тому же показывают большую точность, но вычисления занимают длительное время, и повышается риск переобучения.

Существует несколько типов оберточных методов: детерминированные, которые изменяют множество признаков по определенному правилу, а также рандомизированные, которые используют генетические алгоритмы для выбора искомого подмножества признаков. Среди детерминированных алгоритмов самыми простыми являются алгоритмы последовательного поиска:

- SFS (Sequential Forward Selection) — жадный алгоритм, который начинает с пустого множества признаков, на каждом шаге добавляя лучший из еще не выбранных признаков в результирующее множество. На первом этапе производительность классификатора оценивается по отношению к каждому признаку. Выбирается признак, который работает лучше всего. На втором этапе первый признак пробуется в сочетании со всеми другими функциями. Выбирается комбинация двух функций, обеспечивающих наилучшую производительность алгоритма. Процесс продолжается до тех пор, пока не будет выбрано указанное количество признаков.

- SBS (Sequential Backward Selection) — алгоритм обратный SFS, который начинает с изначального множества признаков, и удаляет по одному или несколько худших признаков на каждом шаге; следует отметить, что эмпирически обратный жадный алгоритм даёт обычно лучшие результаты по сравнению с прямым жадным алгоритмом. Это связано с тем, что обратный жадный алгоритм учитывает признаки, информативные в совокупности, но неинформативные, если рассматривать их по отдельности.

- SSS (Sequential Stepwise Selection) – двунаправленное исключение/отбор - комбинация вышеперечисленных алгоритмов: тестирование на каждом шаге после включения/исключения признаков.

На каждом шаге описанных алгоритмов для оценки качества обычно используются такие критерии, как F-тест (статистика Фишера), t-тест (статистика Стъюдента), скорректированный коэффициент детерминации R^2 и прочие. Сам алгоритм при этом принимает форму последовательности F-тестов, t-тестов.

Популярным оберточным методом также является SVM-RFE (SVM-based Recursive Feature Elimination, рекурсивное исключение признаков), который иногда также относят к встроенным методам отбора. Этот метод использует как классификатор SVM и работает итеративно: начиная с полного множества признаков обучает классификатор, ранжирует признаки по весам, которые им присвоил классификатор, убирает какое-то число признаков и повторяет процесс с оставшегося подмножества признаков, если не было достигнуто их требуемое количество. Таким образом, этот метод очень похож на встроенный, потому что непосредственно использует знание того, как устроен классификатор.

2.4 Встроенные методы (embedded)

Встроенные алгоритмы требуют меньше вычислений, чем wrapper methods (хотя и больше, чем методы фильтрации).

Очень похожи на оберточные методы, но для выбора признаков используется непосредственно структура некоторого классификатора. В оберточных методах классификатор служит только для оценки работы на данном множестве признаков, тогда как встроенные методы используют какую-то информацию о признаках, которую классификаторы присваивают во время обучения.

Основным методом из этой категории является регуляризация.

Регуляризация — это своеобразный штраф за излишнюю сложность модели, который позволяет защитить себя от перетренировки в случае наличия среди признаков неинформативных. Не стоит думать, что регуляризация бывает только в линейных моделях, и для бустинга и для нейросетей существуют свои методы регуляризации.

Существуют различные ее разновидности, но основной принцип общий. Если рассмотреть работу классификатора без регуляризации, то она состоит в построении такой модели, которая наилучшим образом настроилась бы на предсказание всех точек тренировочного сета. Например, если алгоритм классификации линейная регрессия, то подбираются коэффициенты полинома, который аппроксимирует зависимость между признаками и целевой переменной. В качестве оценки качества подобранных коэффициентов выступает среднеквадратичная ошибка (RMSE). Т.е. параметры подбираются так, чтобы суммарное отклонение (точнее суммарный квадрат отклонений) у точек, предсказанных классификатором от реальных точек, было минимальным. Идея регуляризации в том, чтобы построить алгоритм, минимизирующий не только ошибку, но и количество используемых переменных.

2.3.1 **Лассо-регрессия / Ридж-регрессия** (метод регуляризации Тихонова, ridge regression)

Эти методы часто применяются для борьбы с переизбыточностью данных, когда независимые переменные коррелируют друг с другом (т.е. имеет место мультиколлинеарность). Следствием этого является плохая обусловленность матрицы $X^T X$ и неустойчивость оценок коэффициентов регрессии. Оценки, например, могут иметь неправильный знак или значения, которые намного превосходят те, которые приемлемы из физических или практических соображений.

Обе техники позволяют уменьшить размерность данных и устранить/смягчить проблему переобучения. Для этого применяются два способа:

- L1-регуляризация — добавляет штраф к сумме абсолютных значений коэффициентов. Этот метод используется в **Лассо-регрессии**. В процессе работы алгоритма величина приписанных алгоритмом коэффициентов будет пропорциональна важности соответствующих переменных для классификации, а для переменных, которые дают наименьший вклад в устранение ошибки, коэффициенты станут нулевыми. Таким образом, более значимые признаки сохранят свои коэффициенты ненулевыми, а менее значимые – обнулятся. Стоит также отметить, что большие по модулю отрицательные значения коэффициентов тоже говорят о сильном влиянии.
- L2-регуляризация — добавляет штраф к сумме квадратов коэффициентов. Этот метод используется в ридж-регрессии.

Пара слов о параметре альфа. Он позволяет настраивать вклад регуляризирующего оператора в общую сумму. С его помощью мы можем указать приоритет — точность модели или минимальное количество используемых переменных.
В LASSO – всё аналогично, за исключением регуляризирующего оператора. Он представляет собой не сумму квадратов, а сумму модулей коэффициентов. Несмотря на незначительность различия, свойства отличаются. Если в ridge по мере роста альфа все коэффициенты получают значения все ближе к нулевым, но обычно при этом все-таки не зануляются. То в LASSO с ростом альфа все больше коэффициентов становятся нулевыми и совсем перестают вносить вклад в модель. Таким образом, мы получаем действительно отбор признаков. Более значимые признаки сохранят свои коэффициенты ненулевыми, менее значимые — обнулятся. [2,3,12]

2.3.2 Методы выделения признаков.

Эти методы каким-то образом составляют из уже исходных признаков новые, все также полностью описывающие пространство набора данных, но уменьшая его размерность и теряя в репрезентативности данных, т.к. становится непонятно, за что отвечают новые признаки. Все методы feature extraction можно разделить на **линейные** и **нелинейные**.

Одним из самых известных методов **линейного** выделения признаков является PCA (Principal Component Analysis, рус. *метод главных компонент*). Основной идеей этого метода является поиск такой гиперплоскости, на которую при ортогональной проекции всех признаков максимизируется дисперсия. Данное преобразование может быть произведено с помощью сингулярного разложения матриц и создает проекцию только на линейные многомерные плоскости, поэтому и метод находится в категории линейных [5].

ГЛАВА 3. Применение методов-фильтров, оберточных и встроенных методов в реальной задаче

3.1 Описание задачи

Рассмотрим применение методов-фильтров и встроенных методов в реальной задаче [8] – предсказать, зарабатывает ли человек больше $50 тыс. Загрузим библиотеки и данные, для удобства оставив только численные признаки:

- age – возраст
- fnlwgt (final weight) – примерная оценка количества людей, которое представляет каждая строка данных
- educational-num – длительность обучения
- capital-gain – прирост капитала
- capital-loss – потеря капитала
- hours-per-week – количество рабочих часов в неделю

3.2 Обучение моделей

3.2.1 Точность на кросс-валидации и важность признаков для случайного леса:

mean score = 0.82581 +/- 0.00478

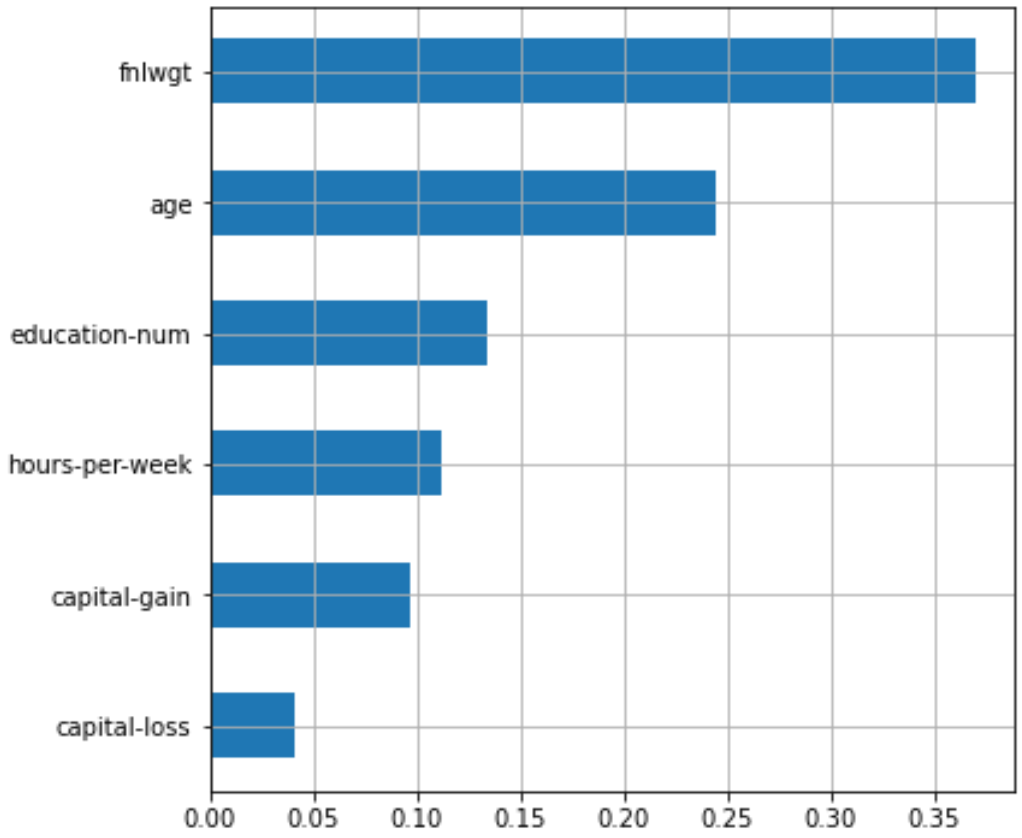

Рисунок 1. – Важность признаков случайного леса

Самым важным признаком для случайного леса [10] является fnlwgt. Это можно интерпретировать как то, что главным фактором того, что человек зарабатывает больше $50 тыс. является количество людей с такими же характеристиками. Такая интерпретация выглядит нелогичной, и происходит это потому, что модели с деревьями могут выдавать сильно смещённую оценку признаков.[6] Притом, чем хуже настроена модель, тем сильнее может быть смещение, поэтому доверять оценкам таких моделей надо с осторожностью.

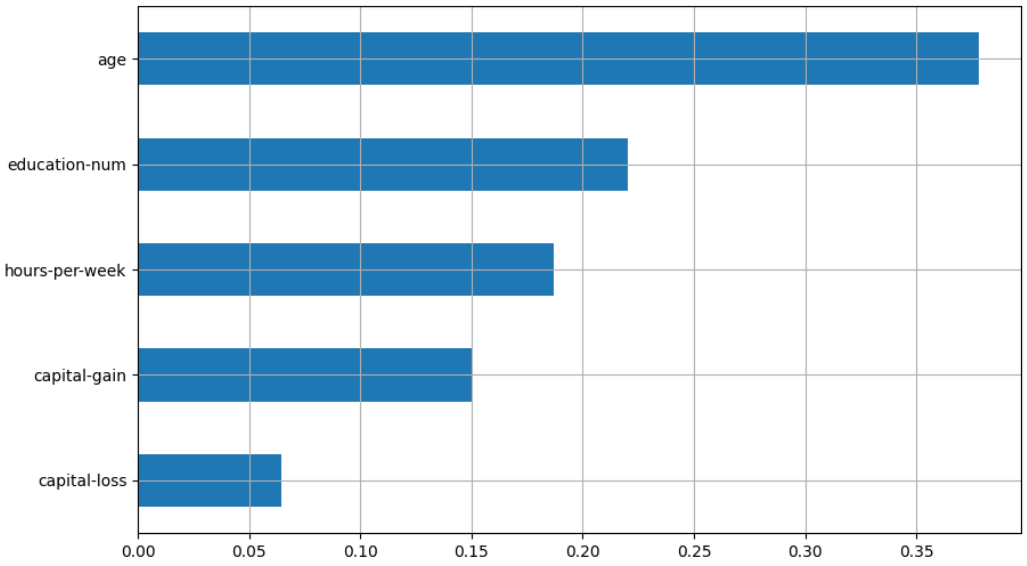

Рисунок 2. Важность признаков для метода прямого последовательного отбора признаков для случайного леса

3.2.3 Точность на кросс-валидации и коэффициенты регрессии для логистической регрессии:

Повторим процедуру для линейной модели (с L1-регуляризацией). Для нормализации данных будем использовать метод PowerTransformer[11].Коэффициенты регрессии нормируем на их сумму.

mean score = 0.82772 +/- 0.00732

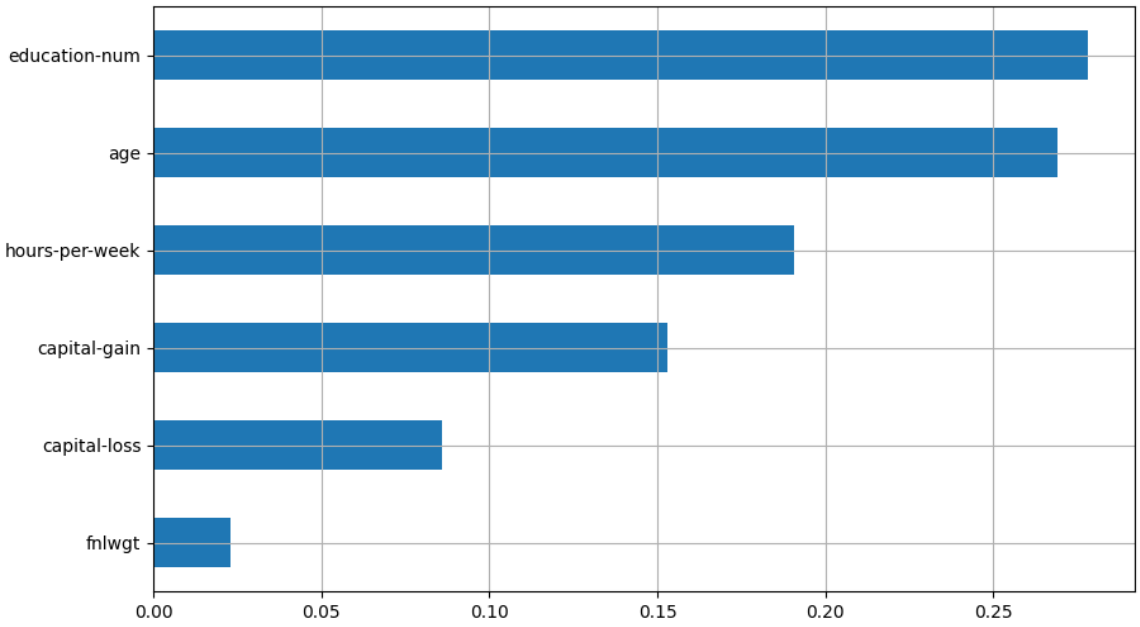

Рисунок 3. – Коэффициенты регрессии логистической регресссии с L1-регуляризацией

Для линейной модели с L2-регуляризацией:

mean score = 0.82772 +/- 0.00732

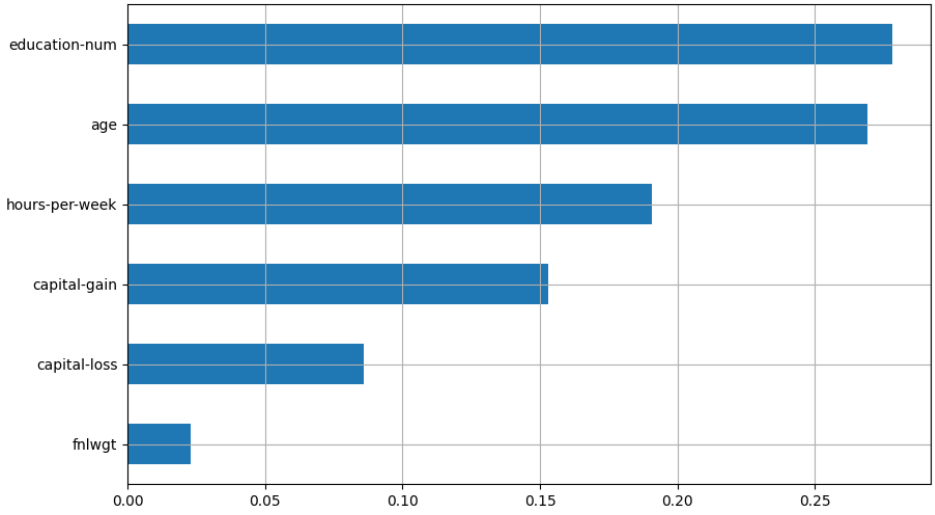

Рисунок 4. – Коэффициенты регрессии логистической регресссии с L2-регуляризацией

3.3 Внесение дополнительных шумовых признаков

Создадим 12 шумовых признаков, элементами которых будут некоррелируемые случайные числа из выборок с нормальным, равномерным и Лапласовым распределениями. Параметры каждого распределения подбираются случайным образом независимо друг от друга.

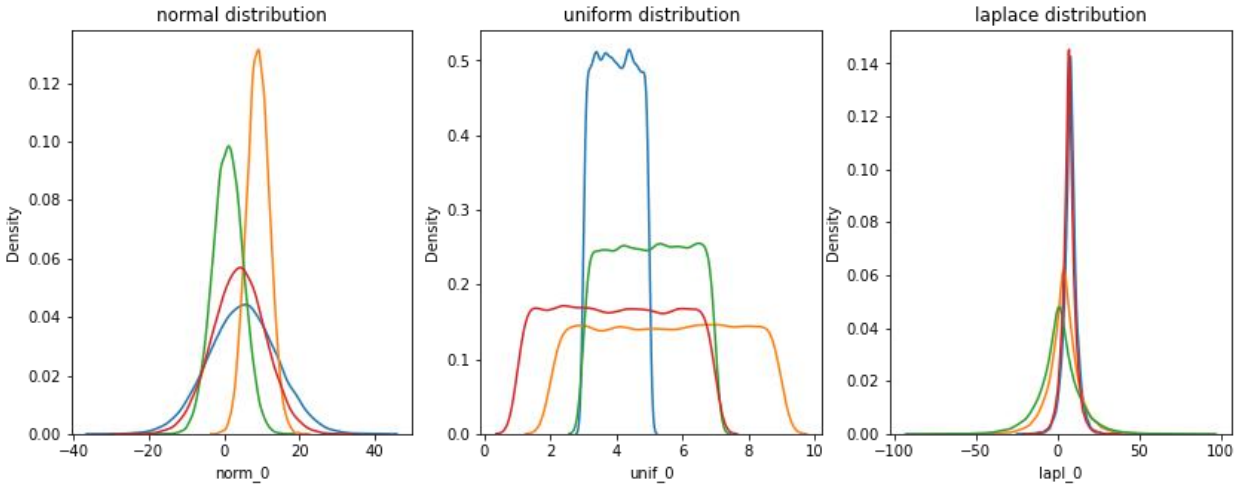

Рисунок 5. - Графики распределения плотности вероятностей шумовых признаков

	score
capital-gain	0.080221
age	0.065703
education-num	0.064743
hours-per-week	0.043655
capital-loss	0.033617
fnlwgt	0.033390
norm_3	0.003217
unif_3	0.002696
norm_0	0.002506
norm_2	0.002052
lapl_3	0.001201
unif_1	0.001144

Таблица 1. Взаимная информация признаков по сравнению с целевой переменной

Заметим, что основные признаки имеют на порядок большую оценку количества взаимной информации по сравнению с шумовыми.

3.3.1 Точность кросс-валидации на зашумлённых данных и важность признаков:

mean score = 0.85439 +/- 0.00447

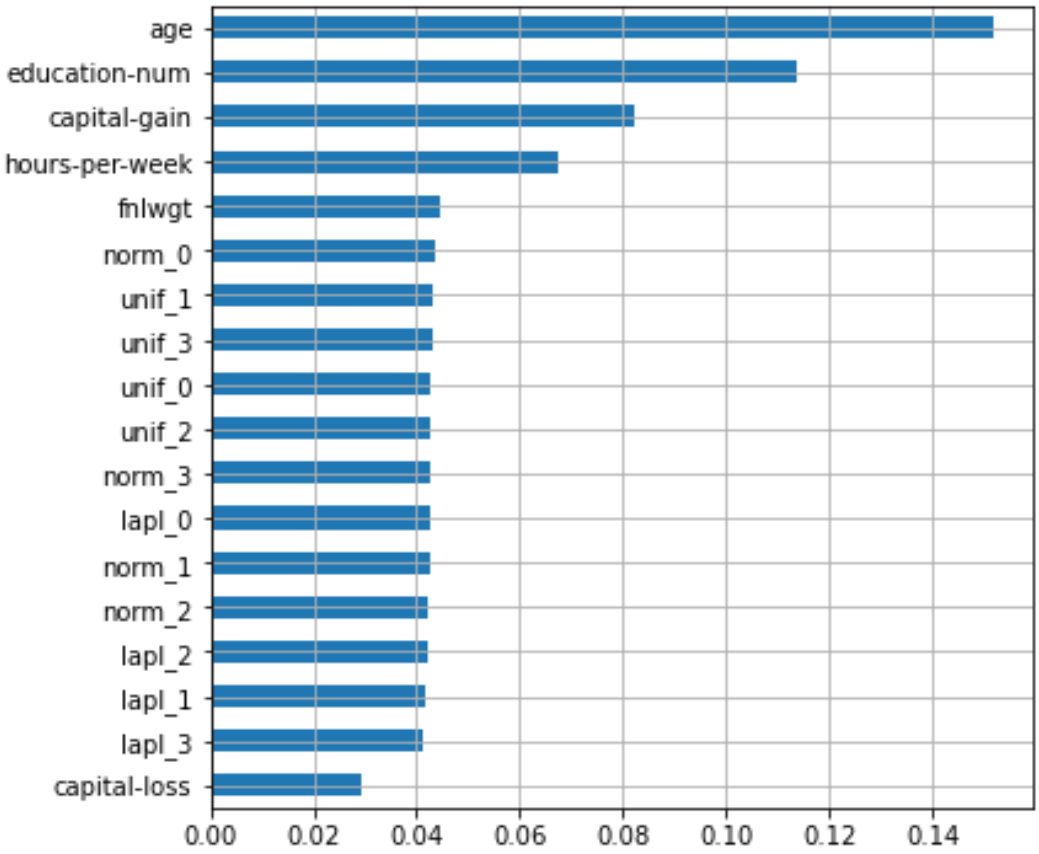

Рисунок 6. – Важность признаков случайного леса с шумовыми признаками

Несмотря на большое количество добавленных шумовых признаков, точность модели на кросс-валидации значительно возросла как на каждом фолде, так и в среднем! Кроме этого, все шумовые признаки имеют высокую важность, сравнимую с двумя оригинальными. Очевидно, что наша модель переобучена, однако в реальных задачах такие ситуации бывает очень сложно распознать, особенно когда при удалении некоторых признаков (про которые неизвестно – шумовые они, или нет) падает валидационная точность. Кроме того, часто бывает сложно подобрать пороговое значение важности признаков для исключения их из модели.

3.3.2 Точность кросс-валидации на зашумлённых данных и коэффициенты регрессии:

mean score = 0.82750 +/- 0.00738

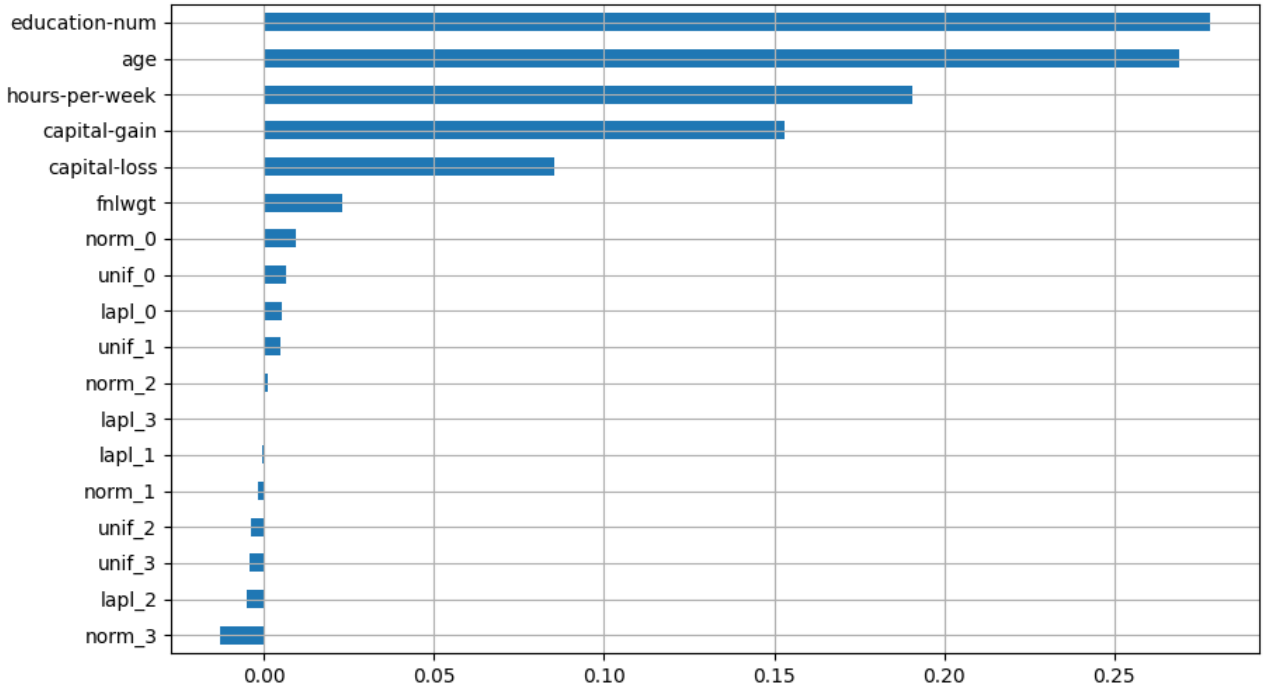

Рисунок 7. – Коэффициенты регрессии с шумовыми признаками при L1-регуляризации

mean score = 0.82750 +/- 0.00739

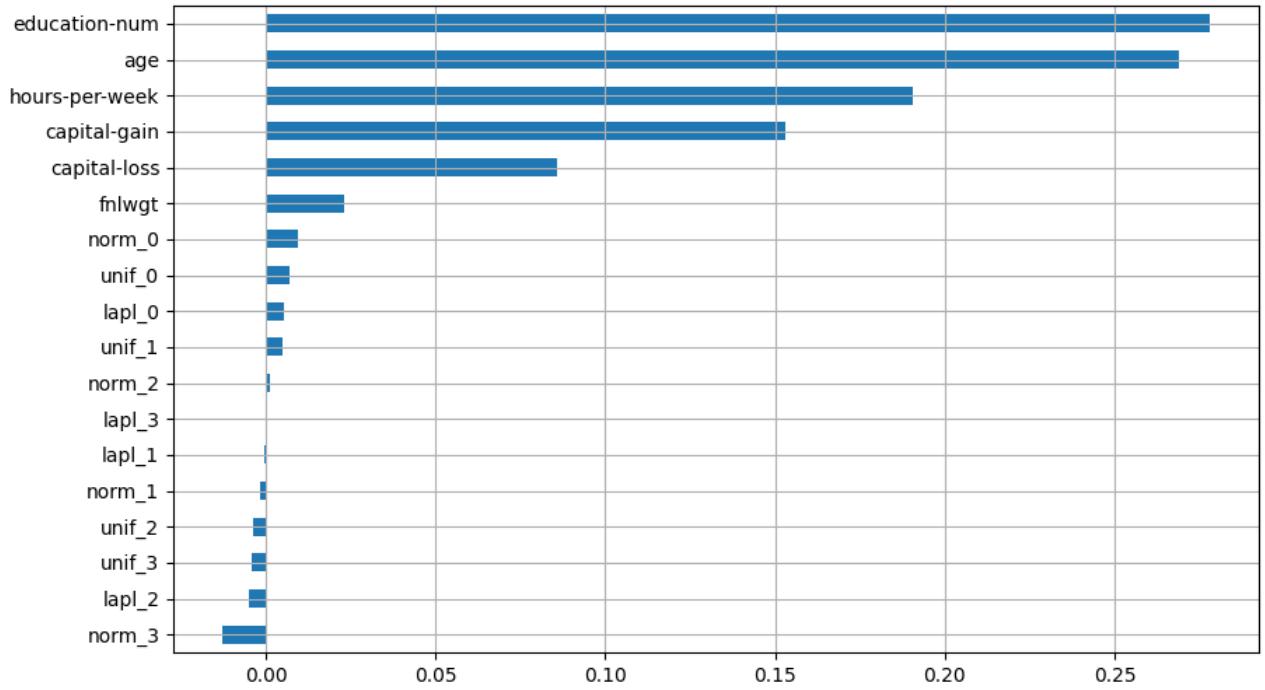

Рисунок 8. – Коэффициенты регрессии с шумовыми признаками при L2-регуляризации

После добавления шумовых признаков модель не переобучилась, к тому же эти признаки имеют значительно меньшие коэффициенты, чем оригинальные. Отметим, что распределение коэффициентов в линейных моделях часто зависит от способа нормализации или масштабирования признаков.

3.4 Подбор гиперпараметров

3.4.1 Случайный лес

Проведём отбор признаков статистическими методами, для чего будем использовать обобщённый вариант SelectKBest и SelectPercentile, который называется GenericUnivariateSelect. Он принимает на вход 3 параметра – функцию оценки, режим отбора и его характеристики. В качестве функции оценки будем использовать взаимную информацию.

Сгенерированные нами признаки имеют низкое значение оценочной функции (scores_), поэтому в дальнейшем селектор не будет их использовать (get_support()=False).

В реальной задаче (когда количество шумовых признаков неизвестно) параметры GenericUnivariateSelect можно находить на кросс-валидации вместе с другими гиперпараметрами модели. Посмотрим, как изменится точность классификаторов после подбора их гиперпараметров, а также количества признаков селектора:

mean score = 0.86671 +/- 0.00364

best params = {'rf_max_depth': 12, 'rf__max_features': 0.3, 'selector_param': 5}

Точность на кросс-валидации значительно выросла, а лучший результат получился всего для 5 признаков:

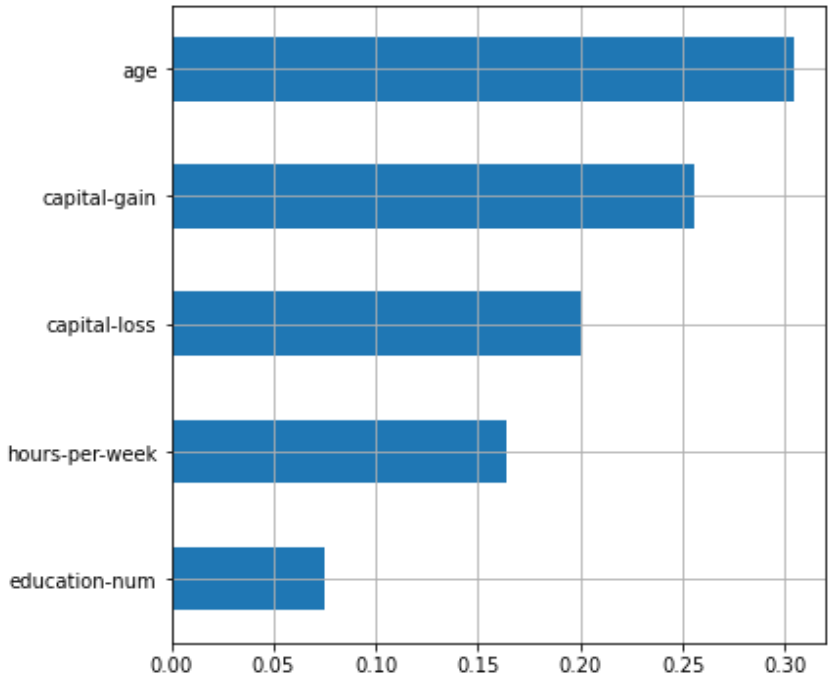

Рисунок 9. – Важность признаков случайного леса после фильтрации

Этот результат был получен после удаления шумовых признаков и признака fnlwgt, который при первоначальной оценке был самым значимым для модели. Однако из всех оригинальных признаков он имел наименьшее значение оценочной функции в GenericUnivariateSelect. Результаты оценки важности признаков после их отбора и настройки модели имеют более логичную интерпретацию – на заработок человека влияют именно характеристики человека, а не параметры самой выборки. Таким образом, статистический отбор признаков бывает полезен для увеличения точности некоторых типов моделей и получения менее смещённой оценки при интерпретации их результатов.

3.4.2 Логистическая регрессия

Посмотрим, как изменятся коэффициенты у признаков после подбора коэффициента регуляризации у логистической регрессии.

mean score = 0.82765 +/- 0.00729

best params = {'lr__C': 0.01}

Средняя точность на кросс-валидации почти не изменилась, но скорректировались коэффициенты у шумовых признаков. Отметим, что сильная регуляризация (L1) может занулить излишне количество признаков.

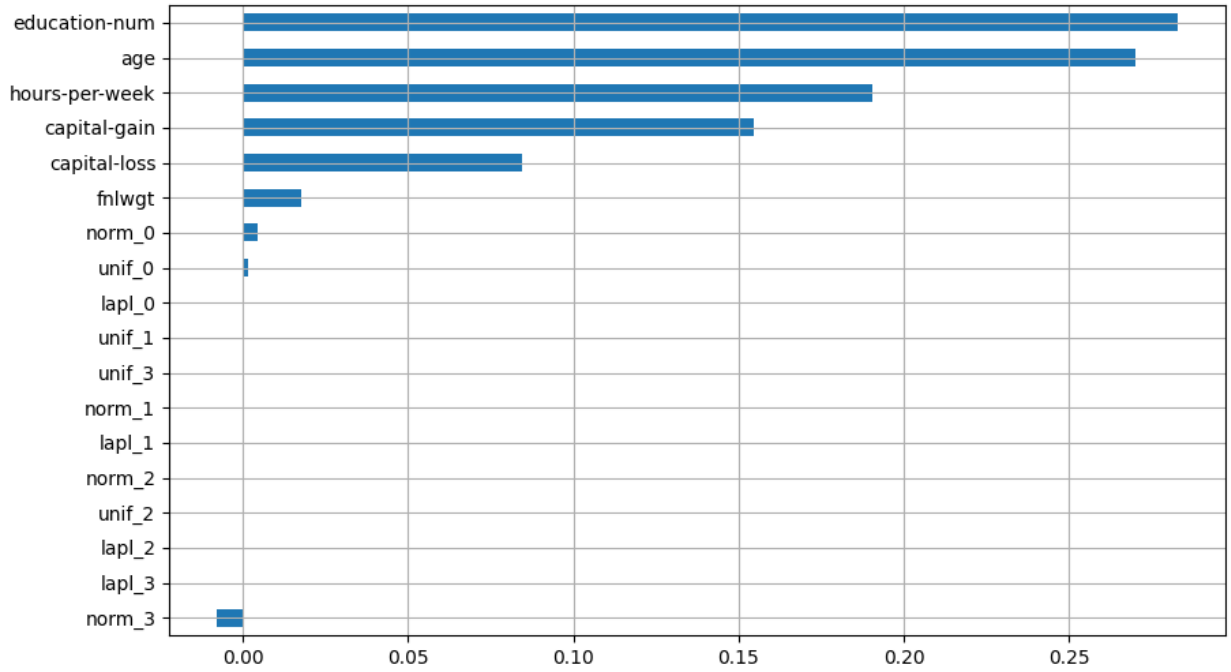

Рисунок 10. – Коэффициенты регрессии после подбора коэффициента L1-регуляризации

mean score = 0.82754 +/- 0.00740

best params = {'lr__C': 0.009}

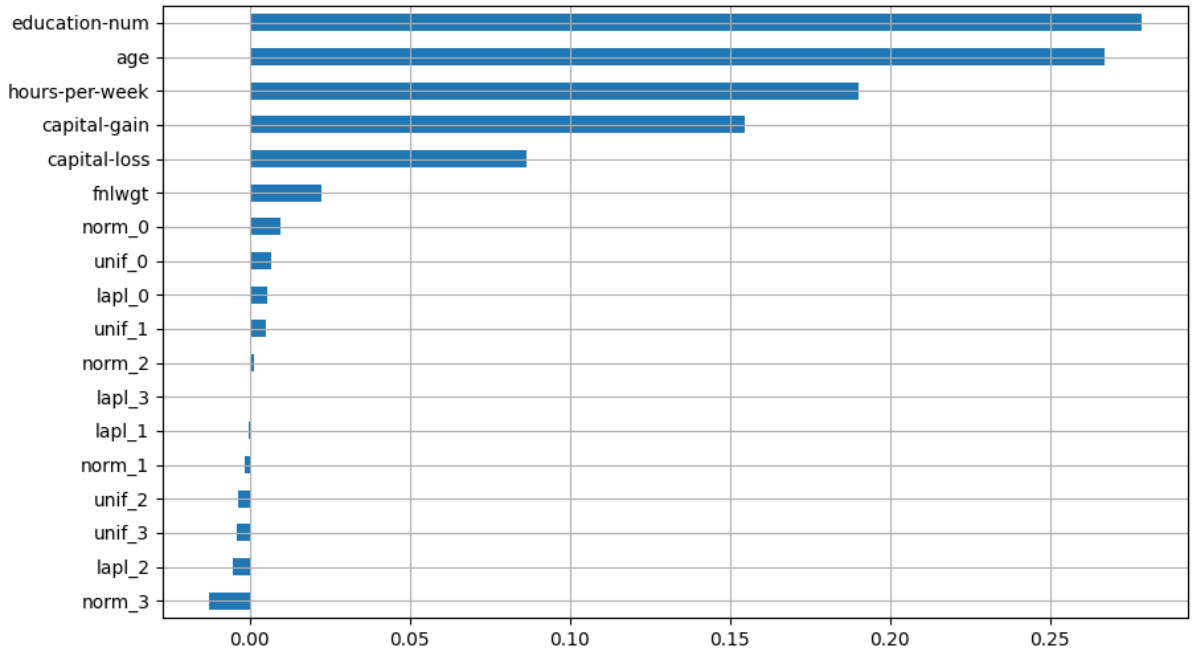

Рисунок 11. – Коэффициенты регрессии после подбора коэффициента L2-регуляризации

Заключение:

В работе были рассмотрено использование фильтров и встроенных методов в задаче отбора признаков.

Достоинства фильтров

- низкая стоимость вычислений (линейно зависит от количества признаков) и интерпретируемость.

Недостатки

- – то, что они рассматривают каждый признак изолировано, поэтому не могут выявить более сложные зависимости в данных, например, зависимость от нескольких предикторов.

Эти методы хорошо подойдут, если в данных большое количество признаков, но малое количество объектов (что встречается, например, в медицинских, или биологических исследованиях).

Также в работе были рассмотрены модель решающего дерева и случайного леса.

Достоинства решающего дерева

- порождение четких правил классификации и быстрые процессы обучения и прогнозирования

Недостатки решающего дерева

- чувствительность к шумам во входных данных
- необходимость отсекать ветви дерева (pruning) или устанавливать минимальное число элементов в листьях дерева или максимальную глубину дерева для борьбы с переобучением

Достоинства случайного леса -

- существование методов оценивания значимости отдельных признаков в модели
- способность эффективно обрабатывать данные с большим числом признаков и классов.

Недостатки случайного леса –

- увеличенная сложность интерпретации по сравнению с решающим деревом.

Список используемых источников:

1. R. Tibshirani. Regression Shrinkage and Selection via LASSO / R.Tibshirani // Journal of the Royal Statistical Society,- 1996 - V. 58, I. 1, pp. 267–288.
2. Practical part code [Electronic resource] / Ed. J.k. Reveal. – College Park M.D., 2019. – Mode of access: https://github.com/ghbdtncjctl/kkkkk. – Date of access: 24.05.2021.
3. Least Angle Regression / B. Efron, T. Hastie, I. Johnstone, R. Tibshirani. // Annals of Statistics. – 2004. - V. 32, No. 2. - pp. 407–499.
4. Ensemble methods (begging, busting, steking) [Electronic resource] / Ed. J.k. Reveal. – College Park M.D., 1996. – Mode of access: https://neurohive.io/ru/osnovy-data-science/ansamblevye-metody-begging-busting-i-steking/. – Date of access: 24.05.2021.
5. Feature Engeneering [Electronic resource] / El. J.K. Higual. – College Park M.K., 2001. – Mode of access: https://nagornyy.me/courses/data-science/feature_engineering/. – Date of access: 24.05.2021.
6. Random Forest Importance [Electronic resource] / Terence Parr, Kerem Turgutlu, Christopher Csiszar, and Jeremy Howard March 26, 2018 – Mode of access: https://explained.ai/rf-importance/. – Date of access: 24.05.2021.
7. Random Forest [Electronic resource] / Terofce Rorr, Kem Turlu, Cher Crenzar, and Jeremy Reveal. - March 26, 2018 – Mode of access: https://www.stat.berkeley.edu/~breiman/randomforest2001.pdf. – Date of access: 24.05.2021.
8. Data Mining and Visualization [Electronic resource] / Ronny Kohavi and Barry Becker // Silicon Graphics. – Mode of access: https://archive.ics.uci.edu/ml/datasets/Adult. - Date of access:24.05.2021.
9. The Elements of Statistical Learning / Hastie T., Tibshirani R., Friedman J. - 2009.
10. Sklearn Library [Electronic resource] / Kerem Tibron, Christopher Higual, March 26, 2009 – Mode of access: ttps://mlcourse.ai/articles/topic5-part3-feature-importance/#3.-Sklearn-Random-Forest-Feature-Importance/. – Date of access: 24.05.2021.
11. Scikit-Learn Library [Electronic resource] / Kerem Tibron, Christopher Higual, March 26, 2009 – Mode of access: https://scikit-learn.org/stable/index.html. – Date of access: 24.05.2021.
12. Least Angle Regression for Stepwise regression [Electronic resource] / Vetrov Semen, Keron Hireal. - May 2, 2001. – Mode of access: http://www.machinelearning.ru/wiki/images/7/7e/VetrovSem11_LARS.pdf. – Date of access: 24.05.2021