Машинное обучение в финансах

Лекция 11: Деревья решений. Тюнинг гиперпараметров. Ансамблевые методы. Случайный лес. Бэггинг. Бустинг

Роман В. Литвинов*

*CRO
Финансовая Группа БКС

Высшая школа экономики, Июнь 2021

Содержание

Содержание

Деревья решений (decision trees)

В отличии от моделей, которые мы видели на прошлых лекциях, деревья решений (decision trees) одинаково успешно решают как задачи регрессии, так и классификации.

По сравнению с линейными моделями деревья могут хорошо предсказывать результат, в случаях наличия довольно сложных, нелинейных взаимосвязей между факторами/предикторами и прогнозируемой величиной.

При этом, при разумном использовании модели, основанные на деревьх решений, остаются хорошо интерпретируемыми. Такие модели поддаются наглядной визуализации и интуитивно понятны, поскольку очень похожи, на процесс принятия решения человеком.

Давайте подробнее рассмотрим как работает эта модель на примере регрессионной проблемы.

Деревья решений (decision trees)

Начнем с простого базового блока нашей модели.

Возьмем предиктор/фактор F_i - он представляет собой определенный набор/вектор значений данного фактора (m точек).

$$F_i = f_1, f_2, ..., f_m$$

Зафиксируем некое значение фактора $s = f_k$, которое делит набор значений нашего вектора на две части: $f_j \geq s$ и $f_j < s$. Такое значение s мы назовем разделяющей переменной (split value).

Теперь давайте условимся, что в для каждого значения фактора, попавшего в первую часть разбиения ($f_j \geq s$) предсказанное значение целевой переменной $\hat{r}$ будет равно среднему значению этой переменной для всех наблюдений f_j в этой части.

Прогноз для второй части разбиения делается аналогичным образом.

Деревья решений (decision trees)

В качестве меры оценки качества такого упрощенного прогноза мы будем использовать показатель, с которым уже встречались ранее, RSS. Оценка общей величины RSS будет состоять из двух частей:

$$RSS_1 = \sum_{i=1}^{n \geq s} \epsilon_i^2 = \sum_{i=1}^{n \geq s} (r_i - \hat{r})^2$$

для первой части разбиения и для второй:

$$RSS_2 = \sum_{i=1}^{n < s} \epsilon_i^2 = \sum_{i=1}^{n < s} (r_i - \hat{r}')^2$$

Понятно, что:

$$RSS = RSS_1 + RSS_2$$

Теперь наша задача сводится к тому, чтобы найти оптимальное значение s, которое бы минимизировало ошибку (RSS).

Деревья решений (decision trees)

Дальнейшая механика построения дерева решений базируется на алгоритме, который называется рекурсивное разбиение (recursive partitioning) . Работает он 'сверху-вниз' следующим образом:

Шаг 1: для каждого предиктора F_i находится оптимальное значение разделяющей переменной s_i и соотвествующая оценка RSS_i.

Шаг 2: выбирается такой фактор F_i и соотвествующее ему значение s_i, комбинация которых дает минимальное значение ошибки в терминах RSS_i. В результате, пространство данных делится на две части.

Шаг 3: процедура описанная выше повторяется, но разбиение применяется не ко всей области данных, а уже к полученным на предыдущем этапе разбиениям области данных (регионам).

Шаг 4: Алгоритм останавливается, когда срабатывает заданный критерий остановки (например, ни один из регионов не содержит больше чем 5 наблюдений, невозможно улучшить качество прогноза).

Деревья решений (decision trees)

В результате на выходе мы имеем набор правил следующего вида:

$$\text{Если } F_1 \geq 2.5 \text{ и } F_2 < 8 \ldots, \text{ то } \hat{r} = c.$$

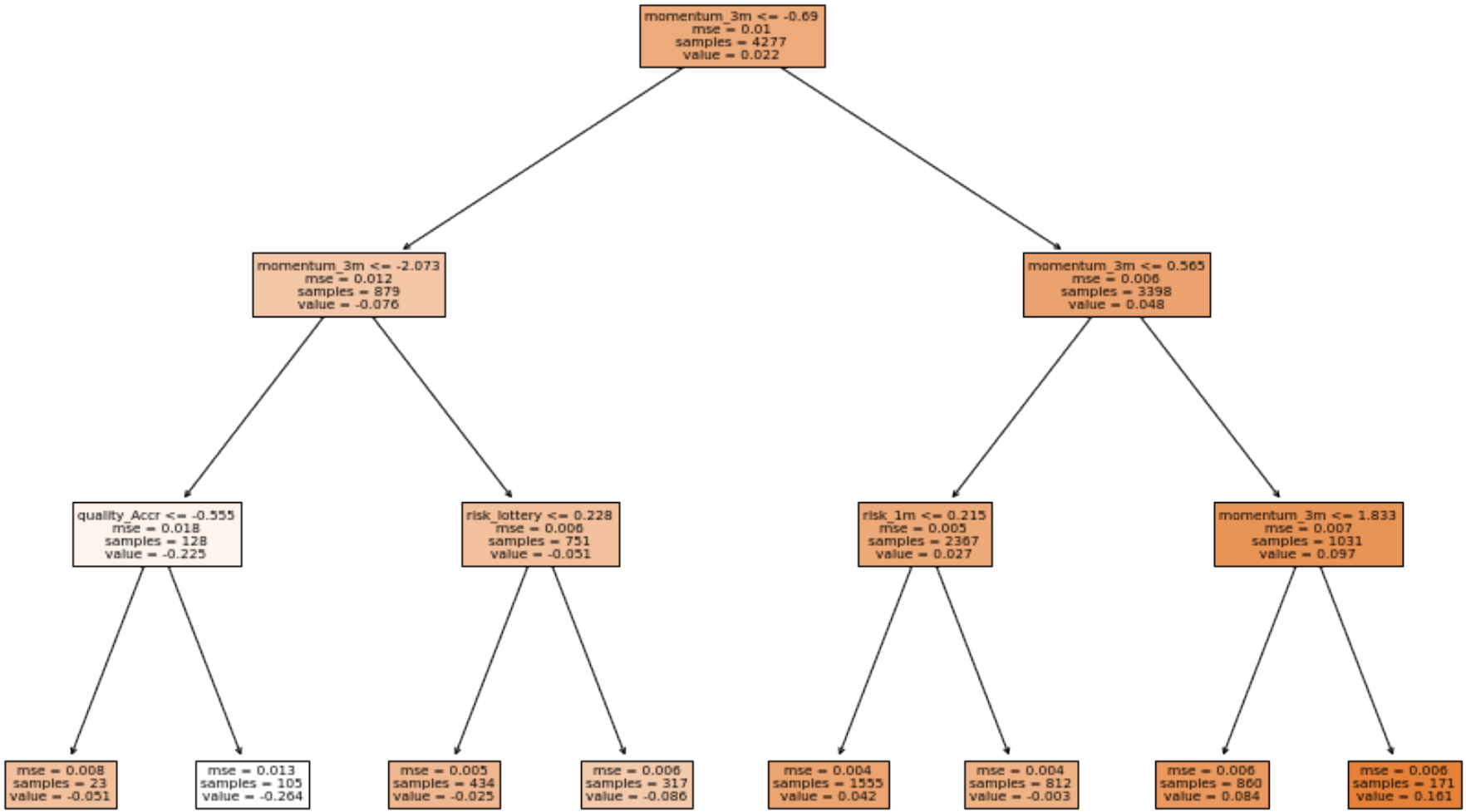
Графически это можно представить в виде дерева следующим образом (см следующий слайд).

Каждое условное правило/ разделяющая переменная представляет собой узел (node) такого дерева.

Граф, характеризующий набор условных правил и объединяющий несколько узлов, называется ветвью дерева.

Количество узлов/условных правил в самой "длинной"ветви дерева называется глубиной дерева (tree depth). Чем глубже дерево, тем сложнее (complexity) модель и больше условных правил/ ветвлений в ней используется.

Деревья решений (decision trees)



Деревья решений (decision trees)

Теперь давайте рассмотрим как будет работать такая модель в случае задачи классификации.

А работать она будет ровным образом так, как мы описали выше. За исключением того, что для определения качества разбиения/условного правила будет использоваться не RSS, а критерий, характеризующий гомогенность/однородность данных в разбиении.

Давайте рассмотрим кейс бинарной классификации (0 или 1). Пусть после применения условного правила/разделяющей переменной мы получили разбиение данных, состоящее из двух подмножеств.

Вероятность того, что наблюдения в одном из подмножеств относятся к классу 1 вычисляется по следующей формуле:

$$p = Prob(Y = 1 | s_1 \geq x) = \frac{k}{n}$$

где k - количество наблюдений в подмножестве, относящихся к 1 классу и n - общее количество наблюдений в подмножестве.

Деревья решений (decision trees)

Для оценки однородности данных в подмножестве часто используется коэффициент Gini impurity, вычисляемый по следующей формуле в случае для двух классов:

$$GI = p(1 - p)$$

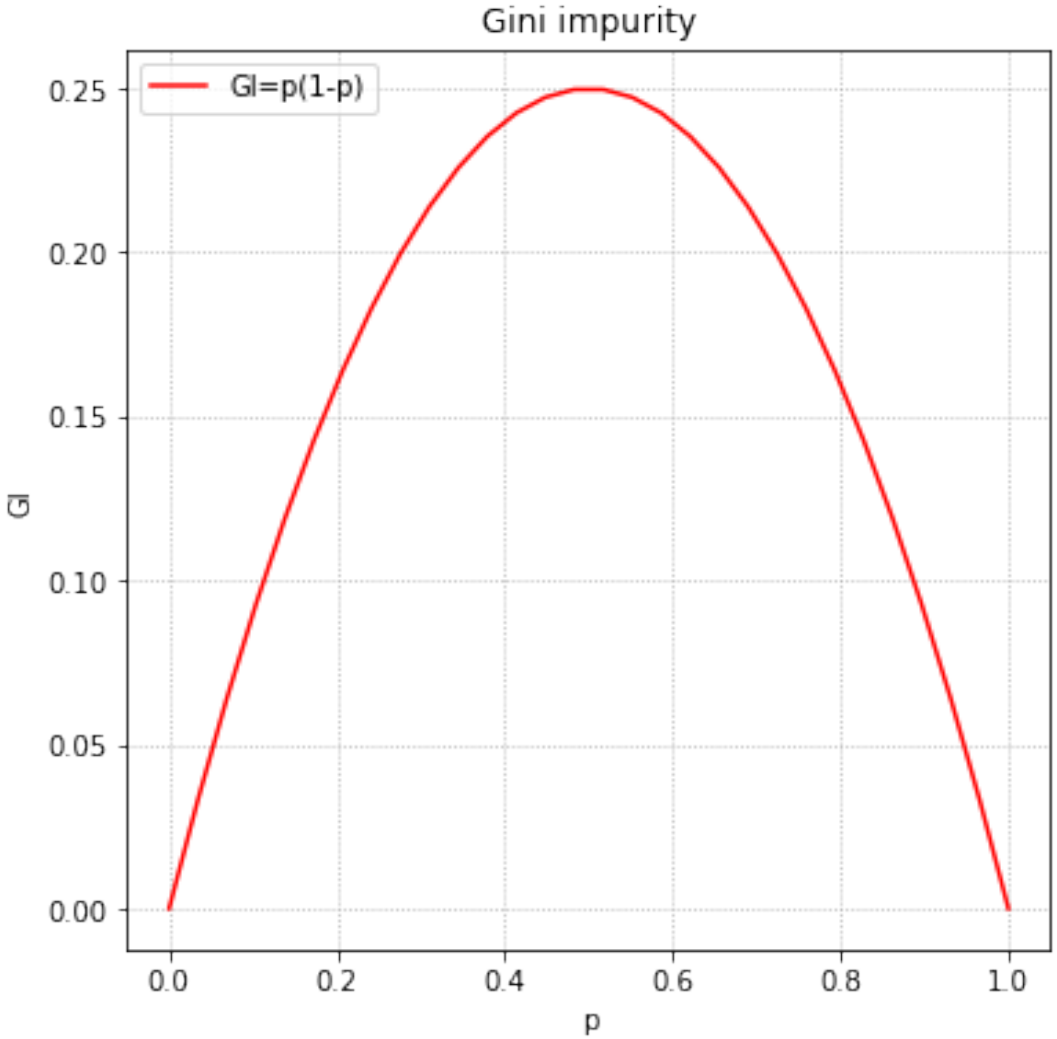
Чем ниже этот коэффициент тем более однородны данные в подмножестве. Чем сильнее преобладает/доминирует в подмножестве какой-либо из классов, тем ниже GI, тем 'чище' (purity) данные.

В случае нескольких классов (k классов) оценка Gini impurity осуществляется по формуле:

$$GI = 1 - \sum_{i=1}^{k} p_i^2$$

В остальном процедура не отличается от регрессионной - рекурсивная оценка и минимизация GI на каждом шаге.

Деревья решений (decision trees)



Содержание

Тюнинг гиперпараметров

Как мы с вами уже видели ранее (лекция 9) модель машинного обучения зачастую состоит из параметров:

- которые 'настраиваются' автоматически в процессе обучения модели (веса соотвествующих предикторов, например)
- параметры, которые требуется задать вручную перед фиттингом/обучением модели (например, коэффициенты регуляризации - λ в ридж и лассо регрессии).

Последние называются гиперпараметрами (hyperparameters) модели.

Процесс выбора оптимального значения таких параметров называется тюнингом/настройкой гиперпараметров модели.

Для деревьев решений одним из гиперпараметров является глубина дерева (max depth в модуле sklearn, который мы используем). Он регулирует насколько сложным будет наше дерево и сколько правил ветвления мы будем использовать.

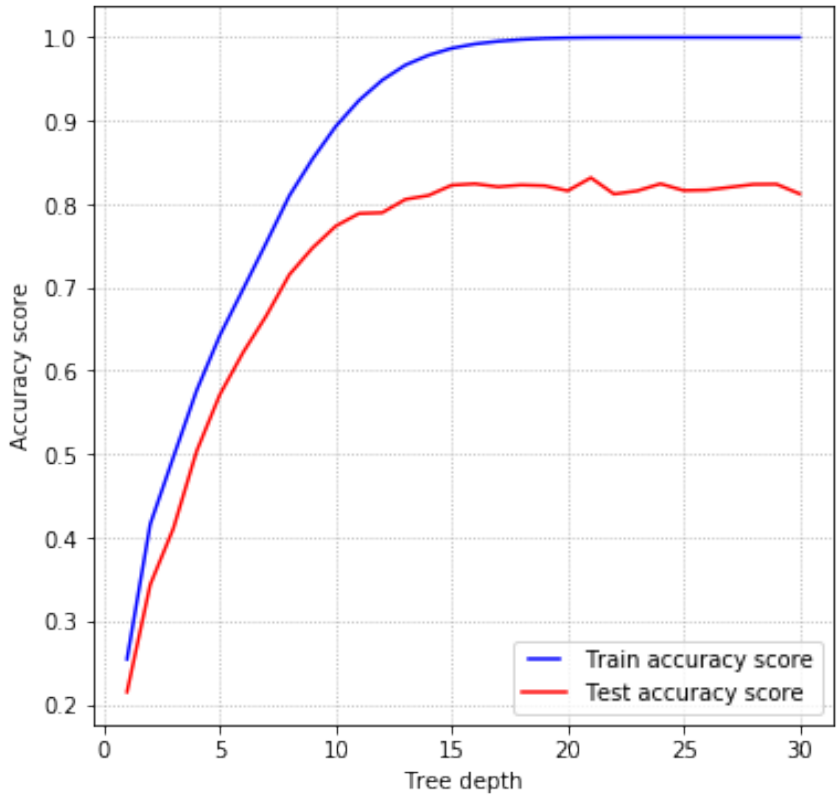
Тюнинг гиперпараметров

В общем виде, алгоритм гипертюнинга некого параметра выглядит следующим образом:

- как и на этапе обучения, выборка данных разбивается на тренировочную и тестовую (например в пропорции 70/30).
- фиксируется определенное значение гиперпараметра и модель обучают на тренировочных данных. Расчитываются соответствующие метрики качества модели (напр., коэффициент точности).
- после этого обученная модель тестируется на тестовых данных.
- значение гиперпараметра изменяется и шаги 2-3 повторяются.
- после n итераций строится профиль зависимости качества модели (для тренировочной и тестовой выборки) от значения гиперпараметра.
- по результатам анализа выбирается соотвествующее значение гиперпараметра, которое используется в итоговой модели. Как оно выбирается?

Тюнинг гиперпараметров

В какой-то момент качество модели на тестовой выборке стабилизируется и дальнейшее ее усложнение приводит только к оверфиттингу.



Выбирается значение параметра, которое дает баланс между сложностью модели (интерпретируемость, computational costs) и ее производительностью на тестовых данных.

Содержание

Ансамблевые методы

Ансамблевые методы (ensemble methods) представляют собой использование моделей машинного обучения, которые базируются на результатах обучения нескольких моделей одного или разных видов - ансамбля (ensemble) моделей.

Это похоже на случай, когда мы проводим опрос среди n экспертов о величине некой предсказываемой переменной и после этого, в качестве прогноза, используем усредненный результат опроса (или наиболее часто повторяемый - majority vote).

В общем виде ансамблевый метод строится на том, что мы обучаем нашу модель на имеющихся данных. После этого еще обучаем n моделей подобного типа (либо модифицированных) или иного типа на этих данных. Затем в качесте прогноза берем полученные предсказания моделей, которые агрегируем по определенной схеме (среднее, взвешенное среднее, majority vote и тп).

Содержание

Ансамблевые методы. Стэккинг

Основными схемами применяемыми в ансамблевом обучении (ensemble learning) являются:

- стэккинг (stacking)
- бэггинг (bagging)
- бустинг (boosting)

Стэккинг представляет собой метод, когда обучающийся алгоритм базирует свое предсказание на прогнозах нескольких других моделей машинного обучения. Т.е. финальный прогноз делается алгоритмом на основании предсказаний сделанных другими алгоритмами.

Например, нейронная сеть, которая в качестве входных сигналов (кроме прочих) использует предсказания линейной регрессионной модели, логистической модели и деревьев решений, будет представлять собой реализацию технологии стэккинга. Подобной реализацией будет и регрессионная модель, линейно взвешивающая сигналы более сложных (и менее интерпретируемых) моделей.

Содержание

Ансамблевые методы. Бэггинг

Бэггинг (bagging, сокр. bootstrap aggregation) это метод ансамблевого обучения, который работает следующим образом:

- из совокупности данных случайным образом генерируются/ выбираются с заменой (sampling with replacement) n выборок заданного размера. Такая процедура называется бутстрэппингом (bootstrapping).
- с использованием полученных выборок обучается n моделей заданного класса (например, деревья решений).
- в качестве прогноза используется агрегированный результат предсказаний n моделей (среднее, взвешенное среднее, majority vote и тп).

Примером, на котором можно продемонстрировать работу бэггинга, является алгоритм случайного леса (random forest).

Содержание

Случайный лес (random forest)

Случайный лес (random forest) представляет собой ансамблевую реализацию метода деревьев решений. В качестве предсказания здесь используется агрегация прогнозов большого количества моделей деревьев решений.

Естественно, если мы говорим о простом использовании n деревьев решений для усреднения прогноза, такой метод вряд ли даст прибавку производительность. Поскольку подобный набор деревьев будет с высокой вероятность представлять из себя одну базовую модель с незначительными вариациями (корзину сильнокоррелированных моделей/ деревьев решений).

Деревья в таком 'лесу' будут практически копиями друг друга.

Недостаточно использовать бэггинг для тренировки нескольких моделей, нужно попытаться каким-то образом декоррелировать результат работы деревьев.

Случайный лес (random forest)

Поэтому используется следующая уловка. Пусть мы имеем i факторов/предикторов:

- как и ранее мы с помощью бутстрэпинга генерируем n случайных выборок данных из имеющейся генеральной совокупности.
- но теперь, мы используем для тренировки соотвествующей модели (дерева решений) не всю совокупность факторов, а только некое (случайное) подмножество из k факторов, где $k < i$.
- на финальном этапе в качестве прогноза используется агрегированный результат предсказаний полученных n деревьев решений.

Такой ансамбль будет представлять из себя именно то, что описано в его названии - случайный лес состоящий из n деревьев решений.

Содержание

Бустинг (boosting)

Бустинг - техника, которая применима к любому ансамблю моделей как и бэггинг, но наиболее часто применяется к деревьям решений.

Бустинг в общем виде конструирует последовательность моделей, в которой каждая последующая модель пытается минимизировать ошибки предыдущей модели (weak learner).

Делается это следующим образом:

- на первом этапе мы задаем количество моделей в ансамбле и обучаем первую модель (дерево решений, в нашем случае).
- затем генерируем новую выборку для обучения следующей модели. При этом более высокая вероятность присваивается неправильно классифицированным наблюдениям (наблюдениям с большей ошибкой в случае регрессии). Новая модель обучается на этой 'смещенной' выборке.

Бустинг (boosting)

- второй шаг повторяется заданное количество раз.
- для предсказания используется взвешенный результат n моделей, где более высокий вес имеют модели полученные на последних этапах (strong learners).

Альтернативной реализацией бустинга является схема, когда вместо вероятности (шаг два) наблюдениям присваиваются определенные веса и соотвествующие полученные ошибки также взвешиваются (модель имеет взвешенную сумму ошибок).

По мере обучения моделей более высокие веса присваиваются неправильно классифицированным наблюдением. Задача сводится к тому, чтобы минимизировать взвешеннную сумму ошибок.

Распространенными библиотеками для бустинга являются AdaBoost из пакета SkLearn (ее мы будем использовать на практике) и open-source решение XGBoost.

Бустинг (boosting)

Обычно, в качестве задаваемых параметров (гиперпараметров) используется количество моделей в ансамбле, глубина дерева и параметр регулирующий скорость обучения (насколько агрессивно изменяется вероятность/ вес присваемые ошибочно классифицируемым наблюдениям).

Необходимо отметить, что ансамблевые методы по мере роста своей сложности могут давать дополнительную прибавку производительности, но при этом часто страдают оверфиттингом.

Также по мере увеличения количества моделей в ансамбле (100 моделей это довольно обыденный случай) такие модели трудно интерпретировать по сравнению с деревьями решений. Для их интерпретации требуется проведение дополнительного анализа используемых ансамблем факторов/предикторов, которые важны для предсказания (feature importance).

Бустинг (boosting)

Важно! Давайте обсудим, возможно ли применение бутсрепинга и аналогичных техник в финансах? В каких случаях - да, и в каких это делать опасно?

Литература

Дополнительная литература к сегодняшней лекции:

- Albon, C. (2018) Machine learning with Python cookbook. O'Reilly.
- Bruce, P., Bruce, A., Gedeck, P. (2020) Practical statistics for data scientists. O'Reilly.
- Hastie, T., Tibshirani, R., Friedman, J. (2017). The elements of statistical learning. Springer. (есть на русском)
- James, G., Witten, D., Hastie, T., Tibshirani, R. (2013). An introduction to statistical learning. Springer. (есть на русском)
- Kroese, D., Botev, Z. (2019) Data Science and Machine Learning. Chapman and Hall/CRC
- Kuhn, M., Johnson, K., (2016). Applied predictive modeling. Springer. (есть на русском)