Домашнее задание #3

Вопрос 1: Для чего и в каких случаях полезны различные варианты усреднения для метрик качества классификации: micro, macro, weighted?

Некоторые метрики, по сути, определены для задач бинарной классификации (например, f1_score, roc_auc_score). В этих случаях по умолчанию оценивается только положительный лейбл, предполагая, что по умолчанию положительный класс имеет лейбл 1, хоть это и можно настроить с помощью pos_label.
При расширении бинарной метрики до многоклассовых или мултилейбл проблем данные рассматриваются как набор бинарных проблем, по одной для каждого класса. Затем существует ряд способ усреднения бинарных метрических вычислений по набору классов, каждый из которых может быть полезен в определенном сценарии. Там, где это возможно, следует выбрать один из них, используя параметр усреднения.

- **Macro:**
  Просто вычисляет среднее значение бинарных метрик, давая равный вес каждому классу. В задачах, где редко встречающиеся классы, тем не менее, важны, макро-усреднение может быть средством подчеркивания их производительности. С другой стороны, предположение, что все классы одинаково важны, часто не соответствует действительности, так что макро-усреднение будет чрезмерно подчеркивать типично низкую производительность на редком классе.

- **Weighted:**
  Учитывает дисбаланс классов путем вычисления среднего значения бинарных метрик, в которых оценка каждого класса взвешивается по его наличию в истинной выборке данных.

- **Micro:**
  Дает каждой паре класса выборки равный вклад в общую метрику (за исключением результата веса выборки). Вместо суммирования метрики по классам суммируются дивиденды и делители, которые составляют метрики по классам для вычисления общего коэффициента. Микро-усреднение может быть предпочтительнее в многомаркировочных настройках, включая классификацию по нескольким классам, где класс большинства должен быть проигнорирован.

Источник: Документация Scikit-Learn

Вопрос 2: В чем разница между моделями xgboost, lightgbm и catboost или какие их основные особенности?

Время возникновения:

- Изначально XGBoost начинался как исследовательский проект Tianqi Chen в марте 2014 года, но после 2016 стал известен широкому кругу лиц.
- Компания Microsoft выпустила первую стабильную версию LightGBM в январе 2017 года.
- В апреле 2017 года Яндекс представил CatBoost публике.

Сравнение LightGBM и XGBoost

LightGBM использует новую технику градиентной односторонней выборки (GOSS) для фильтрации экземпляров данных для нахождения разделения, в то время как XGBoost использует алгоритм предварительной сортировки и алгоритм на основе гистограммы для вычисления лучшего разделения. Здесь экземпляры означают наблюдения / образцы.

Для начала необходимо проанализировать метод работы предварительной сортировки

- Для каждого node необходимо перечислить все features,
- Для каждой feature нужно отсортировать экземпляры согласно feature value,
- Использовать линейное сканирование, чтобы решить, как лучше разделить информацию по этой feature
- Взять лучшее спит-решение по всем feature

Проще говоря, алгоритм, основанный на гистограмме, разделяет все точки данных функции на дискретные бины и использует эти бины для нахождения разделенного значения гистограммы. Хотя по скорости обучения он эффективнее, чем алгоритм предварительной сортировки, который перечисляет все возможные точки разделения на предварительно отсортированных значениях функции, по скорости он все же отстает от GOSS.

GOSS

В AdaBoost вес образца служит хорошим индикатором важности образцов. Однако в дереве решений по градиентному форсированию (GBDT) нет собственных весов образцов, и поэтому методы выборки, предлагаемые для AdaBoost, не могут быть применены напрямую. В данном случае речь идет о градиентной выборке.

GOSS сохраняет все экземпляры с большими градиентами и выполняет случайную выборку на экземплярах с малыми градиентами. Например, у меня есть 500K строк данных, где 10k строк имеют более высокие градиенты. Поэтому мой алгоритм выберет (10k строк с более высоким градиентом + x% оставшихся 490k строк выбраны случайным образом). Если предположить, что x составляет 10%, то общие выбранные строки составляют 59k из 500K, на основе которых, если они будут найдены, то будут разбиты на части.

Как каждая модель работает с категорическими переменными?

CatBoost

CatBoost имеет возможность гибко задавать индексы категориальных столбцов, чтобы их можно было кодировать как одноразовую кодировку с использованием параметра one_hot_max_size.

Если ничего не передается в аргументе cat_features, CatBoost будет рассматривать все столбцы как числовые переменные.

Для остальных категориальных столбцов, которые имеют уникальное количество категорий больше, чем one_hot_max_size, CatBoost использует эффективный метод кодирования, который похож на среднюю кодировку, но уменьшает переподгонку.

LightGBM

Подобно CatBoost, LightGBM также может работать с категорическими функциями, принимая на вход названия функций. Он не преобразуется в одноразовое кодирование и работает намного быстрее, чем одноразовое кодирование. LGBM использует специальный алгоритм для нахождения разделенного значения категориальных признаков.

XGBoost

В отличие от CatBoost или LGBM, XGBoost сам по себе не может справиться с категорическими функциями, он принимает только численные значения, аналогичные Random Forest. Поэтому перед отправкой категориальных данных в XGBoost необходимо выполнить различные кодировки, такие как кодировка метки, средняя кодировка или одноразовая кодировка.

Сходство в гиперпараметрах

Все эти модели имеют множество параметров для настройки, ниже приведен список этих параметров в соответствии с их функциями и их аналогами в различных моделях.

Таблица представлена ниже.

	XGBoost	CatBoost	LightGBM
Важные параметры, которые контролируют переобучение	1. learning_rate или eta - оптимальные значения находятся между 0.01 и 0.2 2. max_depth 3. min_child_weigh	1. learning_rate 2. depth - значение может быть любым int до 16 3. Нет аналога для min_child_weight 4. l2-leaf-reg: l2 регуляторный коэффициент. Используется для подсчета leaf value	1. learning_rate 2. max_depth: стандартное значение 20. Важно отметить, что дерево все еще растет в листья. Следовательно, важно настроить num_leaves, которые должны быть меньше 2^(max_depth) 3. min_data_in_leaf: стандартно 20
Параметры для категориальных переменных	Нет	1. cat_features: он обозначает индекс категориальных признаков 2. one_hot_max_size: использует одноразовое кодирование для всех функций с количеством различных значений, меньшим или равным заданным значениям параметра (макс. 255)	1. categorical_feature: указывать категорические признаки для дальнейшего обучения
Параметры для контроля скорости	1. colsample_bytree: подвыборочное соотношение колонок 2. subsample: подвыборочное соотношение учебного экземпляра 3. n_estimators: максимальное количество деревьев принятия решений, высокое значение может привести к переобучению	1. rsm: Random Subspace Method. Процентное соотношение features для использования при каждом раздельном выборе. 2. Нет аналога 3. iterations: максимальное значение деревьев	1. feature_fraction: доля features, которые необходимо учитывать при каждой итерации 2. bagging_fraction: данные для каждой итерации и, как правило, используются для ускорения обучения 3. num_iterations: количество повышающих итераций, по умолчанию 100