Отчет по заданию спецкурса

“Основы ММП”

Кравцовой О. А.

Выполнил студент

Булкин Антон

208 группы ВМК МГУ

Содержание

Описание задачи

В данном задании я решил рассмотреть вопрос многоклассовой классификации.

Набор данных, используемый в данном задании далее, представляет собой набор данных про участки леса в США. Каждый экземпляр представляет собой информацию об участке леса 30x30, распределенную по 54 признакам, являющихся логическими индикаторами, а также дискретными или непрерывными результатами измерениями.

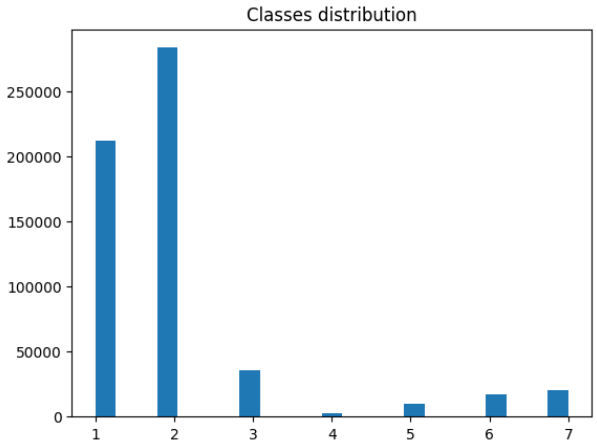
Существует всего 7 типов растительного покрова, что создает задачу многоклассовой классификации, т. е. по вектору значений признаков данного участка предсказать к какому типу относится данный участок.

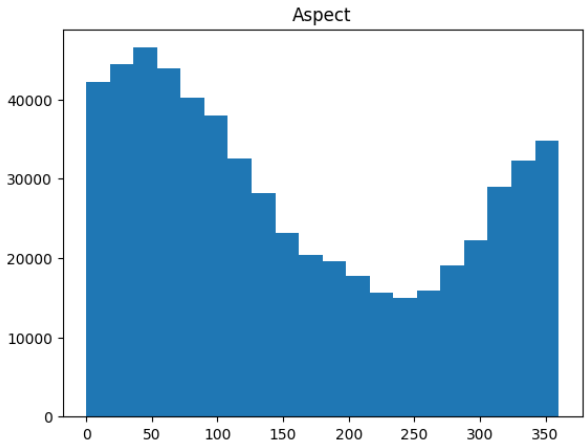
Данная проблема могла возникнуть в процессе работы специалистов-почвоведов при анализе большой поверхности территорий, а т. к. внутри даже каждого класса существуют различия по значениям результатов измерений, приходится тратить значительное время на типизацию каждого отдельного участка леса. С чем и могут помочь методы машинного обучения, чтобы сократить время и ресурсы на типизацию каждого отдельного участка по результатам измерений.

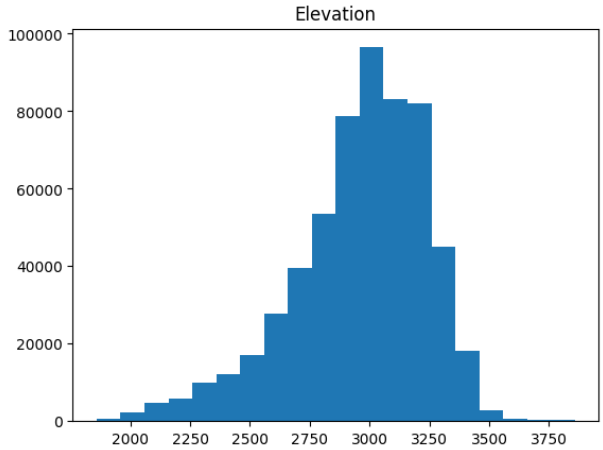
Набор данных был использован из библиотеки Scikit-learn из списка готовых к использованию. (использованная функция для получения данных - sklearn.datasets.fetch_covtype())

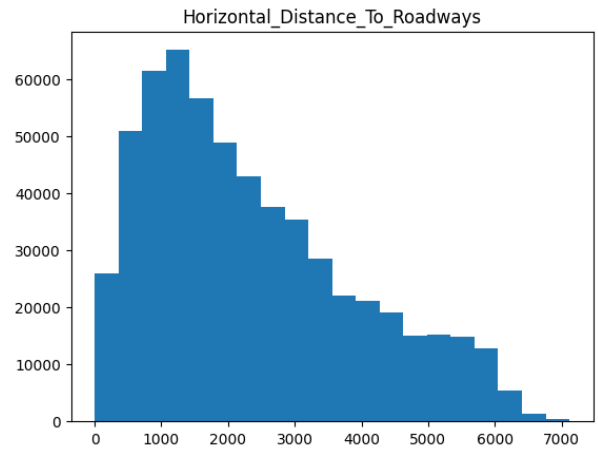
Анализ выборки

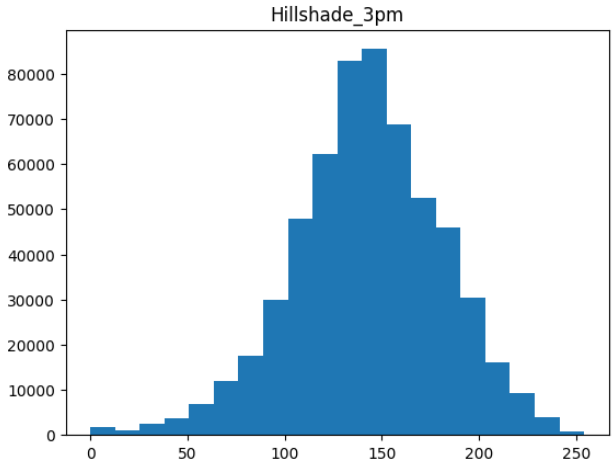
Ниже приведен анализ некоторых признаков выборки:

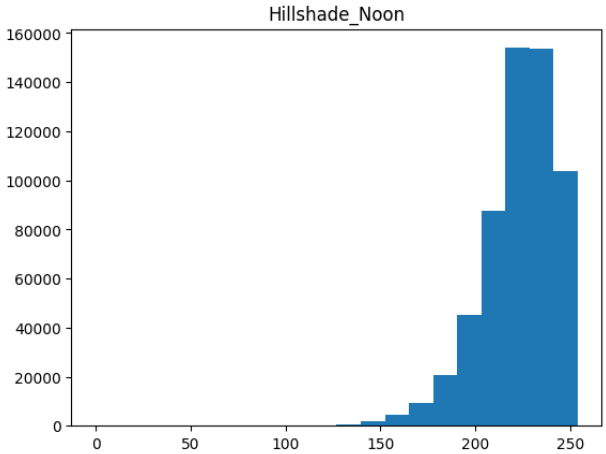












Метрика качества

В данной задачи рассматривалась точность определения классов, поэтому метрикой качества была выбрана оценка точности классификации (Accuracy = $\frac{TP+TN}{TP+TN+FP+FN}$, где TP - количество истинно положительных результатов, FN - количество ложноотрицательных результатов и FP - количество ложноположительных результатов, TN - количество ложноположительных результатов) и соответствующая ей метрика из scikit-learn - accuracy_score().

accuracy_score() вычисляет долю правильных прогнозов классификатора по формуле для точности: Accuracy(y, $\hat{y}$) = $\frac{1}{n_{samples}}\sum_{i=1}^{n_{samples}} 1(y_i = \hat{y}_i)$, где $\hat{y}$ - прогнозируемое значение i-го образца, а y - соответствующее истинное значение.

Модели для решения задачи

При рассмотрения данной задачи мной были использованы 2 метода для классификации из библиотеки Scikit-learn: SGDClassifier и DecisionTreeClassifier.

SGDClassifier

Стохастический градиентный спуск (SGD) — это простой, но очень эффективный подход к подгонке линейных классификаторов и регрессоров под выпуклые функции потерь, такие как (линейный) метод опорных векторов и логистическая регрессия.

Машинное обучение рассматривают задачу минимизации целевой функции, имеющей форму суммы $Q(w) = \frac{1}{n}\sum_{k=1}^{n} Q_i(w)$, где параметр w минимизирующий $Q(w)$ следует оценить. Каждый член суммы Q_i обычно ассоциируется с *i*-ым наблюдением в наборе данных, использованном для обучения.

Задача минимизации суммы возникает также при минимизации эмпирического риска. В этом случае Q_i(w) является значением функции потерь на *i*-ом примере, а Q(w) является эмпирическим риском. При использовании для минимизации вышеприведённой функции стандартный метод градиентного спуска осуществляет следующие итерации:

$w = w - \alpha * \nabla Q(w)$, где α является размером шага, называемым скоростью обучения в машинном обучении.

В стохастическом градиентном спуске истинный градиент Q(w) аппроксимируется градиентом одного тренировочного примера $w = w - \alpha * \nabla Q_i(w)$, пробегая через тренировочное множество, алгоритм осуществляет приведённый выше пересчёт для каждого тренировочного примера. Для достижения сходимости алгоритма может потребоваться несколько проходов по тренировочному набору данных. Перед каждым новым проходом данные в наборе перемешиваются для устранения возможности зацикливания алгоритма. Типичные реализации могут использовать адаптивную скорость обучения для улучшения сходимости.

Для рассматриваемого метода я сравнивал следующие значения функции потерь (loss functions):

- hinge – средняя hinge loss.

В случае двоичного класса, предполагая, что метки в y_true закодированы как +1 и -1, при ошибке прогнозирования значение margin = y_true * pred_decision всегда отрицательное (поскольку знаки не совпадают), что означает, что 1 - margin всегда больше 1. Таким образом, суммарная hinge loss является верхней границей количества ошибок, допущенных классификатором.

При многоклассовой классификации поле для мультикласса рассчитывается в соответствии с методом Краммера-Сингера. Как и в бинарном случае, суммарная hinge loss является верхней границей количества ошибок, допущенных классификатором.

- squared_hinge – квадратичная версия предыдущей функции потерь.
- log_loss - Логарифмическая потеря, она же логистическая потеря или кросс-энтропийная потеря.

Это функция потерь, используемая в логистической регрессии и ее расширениях, таких как нейронные сети, определяемая как отрицательное логарифмическое правдоподобие логистической модели, которая возвращает y_pred вероятности для своих обучающих данных y_true. Для одного образца с истинной меткой $y \in \{0, 1\}$ и оценкой вероятности p = Pr (y = 1) логарифмическая потеря равна:

$$L_{log}(y, p) = -(y \log p + (1 - y) \log(1 - p))$$

- perceptron - линейные потеря, используемая алгоритмом персептрона.
- squared_error – функция потерь, применяемая в методах регрессии, но иногда применимая к задачам классификации, вычисляема по формуле Squared_error (y, ŷ) = $\frac{1}{n_{samples}} \sum_{i=1}^{n_{samples}} (y_i - \hat{y}_i)^2$, где ŷ - прогнозируемое значение i-го образца, а y - соответствующее истинное значение.
- huber - функция потерь, используемая в регрессии

L (t, y) = max (0, 1 - ty)^2 , если, ty > -1, и L(t, y) = -4ty иначе.

- modified_huber – модифицированная функция потерь huber, используемая для классификации.
- epsilon_insensitive

Epsilon_insensitive функция потерь может быть математически выражены как:

$L(y_i, f(x_i)) = \max(0, |y_i - f(x_i)| - \varepsilon)$, где y_i - оценка классификатора, а $f(x_i)$ - фактический результат (класс), требуемый от классификатора.

Значение ε определяет расстояние, в пределах которого ошибки считаются равными нулю. Функция потерь игнорирует ошибки, которые меньше или равны значению epsilon, относя их к нулю. Таким образом, функция потерь эффективно заставляет оптимизатор находить такую гиперплоскость, чтобы труба шириной эпсилон вокруг этой гиперплоскости содержала все точки данных.

- squared_epsilon_insensitive – квадратичная версия epsilon_insensitive функции потерь.

DecisionTreeClassifier

Деревья решений используются в повседневной жизни в самых разных областях человеческой деятельности, порой и очень далеких от машинного обучения. Деревом решений можно назвать наглядную инструкцию, что делать в какой ситуации.

Дерево решений как алгоритм машинного обучения – объединение логических правил вида "Значение признака a меньше x и значение признака b меньше y… => Класс 1" в структуру данных "Дерево". Огромное преимущество деревьев решений в том, что они легко интерпретируемы, понятны человеку.

Для рассматриваемого метода для перебора параметров по сетке я выбрал 2 параметра: criterion и min_samples_split.

- criterion - функция для измерения качества разделения.

Были рассмотрены значения:

  - entropy

    Энтропия — это поддающийся количественной оценке физический признак и научное понятие, которое часто ассоциируется с беспорядком, непредсказуемостью или неопределенностью.

    Энтропия помогает построить подходящее дерево решений для выбора наилучшего разделителя. Энтропия может быть определена как мера чистоты разделения. Энтропия всегда находится в диапазоне от 0 до 1. Энтропия любого разделения может быть рассчитана по этой формуле:

    Entropy = - $\sum_{i=1}^{n} p_i * \log_2 p_i$, где p_i - доля точек данных, относящихся к классу i, а n - количество классов.

  - gini – соответствует критерию Джини.

    Внутренняя работа Gini impurity также в некоторой степени похожа на работу энтропии в Дереве решений. В алгоритме дерева решений оба метода используются для построения дерева путем разбиения по соответствующим признакам, но в вычислениях обоих методов есть существенная разница. Коэффициент Джини примеси признаков после разделения может быть рассчитан с помощью этой формулы.

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

  - log_loss – была описана ранее в пункте SGDCLassifier.
- min_samples_split - минимальное количество выборок, необходимое для разделения внутреннего узла. Рассмотрены целочисленные значения в пределах [2, 30].

GridSearchCV

Данный метод библиотеки scikit-learn позволяет провести перебор параметров по сетке для заданного метода для определения лучших параметров, соответствующих лучшему значению целевой метрики.

Для сравнения были рассмотрены следующие метрики:

- jaccard_weighted - оценка коэффициента сходства Жаккарда, которую можно вычислить по формуле: J (A, B) = $\frac{|A \cap B|}{|A \cup B|}$, для 2х заданных множеств A и B(даны как пример для вычисления).
- f1_weighted – может быть интерпретирована как среднее гармоническое значение точности и запоминания, при котором показатель F1 достигает своего наилучшего значения при 1, а наихудшего - при 0. Относительный вклад precision и recall в показатель F1 равен. Формула для определения показателя F1 имеет следующий вид: F1 = $\frac{2*TP}{2*TP+FP+FN}$, где TP - количество истинно положительных результатов, FN - количество ложноотрицательных результатов и FP - количество ложноположительных результатов.
- top_k_accuracy – этот показатель вычисляет количество случаев, когда правильная метка входила в число k лучших предсказанных меток.
- precision_weighted - определяется как количество истинных срабатываний (TP), превышающее количество истинных срабатываний плюс количество ложных срабатываний (FP): Precision = $\frac{TP}{TP+FP}$
- recall_weighted - определяется как количество истинных срабатываний (TP), превышающее количество истинных срабатываний плюс количество ложноотрицательных результатов (FN): Recall = $\frac{TP}{TP+FN}$
- roc_auc_ovr - вычисляет площадь под кривой ROC AUC на основе прогнозируемых значений.
- roc_auc_ovr_weighted – модифицированный аналог предыдущей метрики.

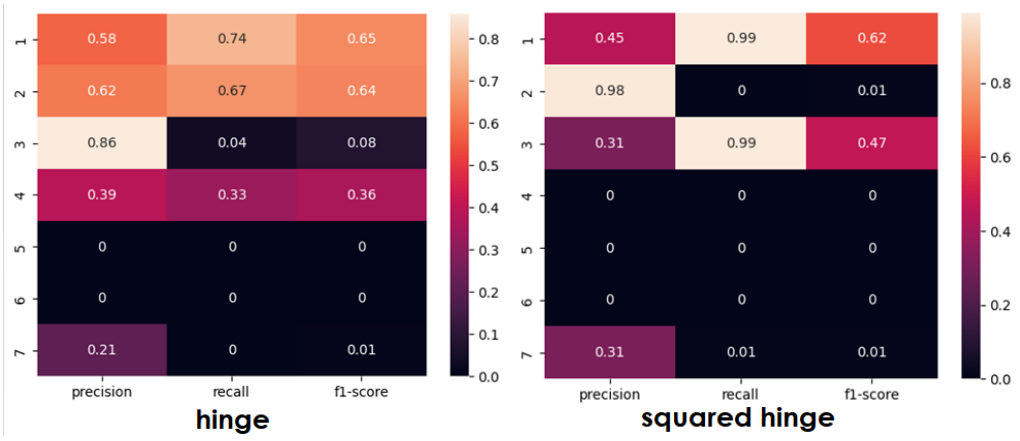
Замечание: постфикс weighted обозначает изменение алгоритма вычисления от стандартного к средневзвешенному значению.

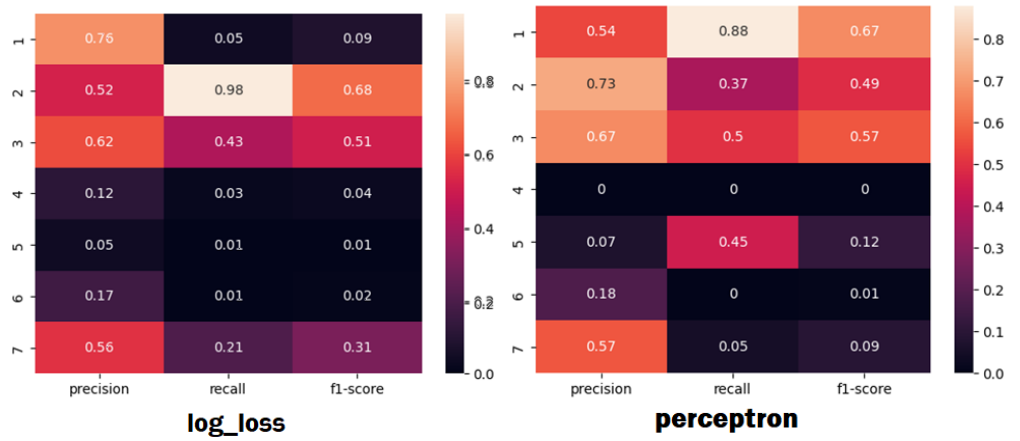
Результаты

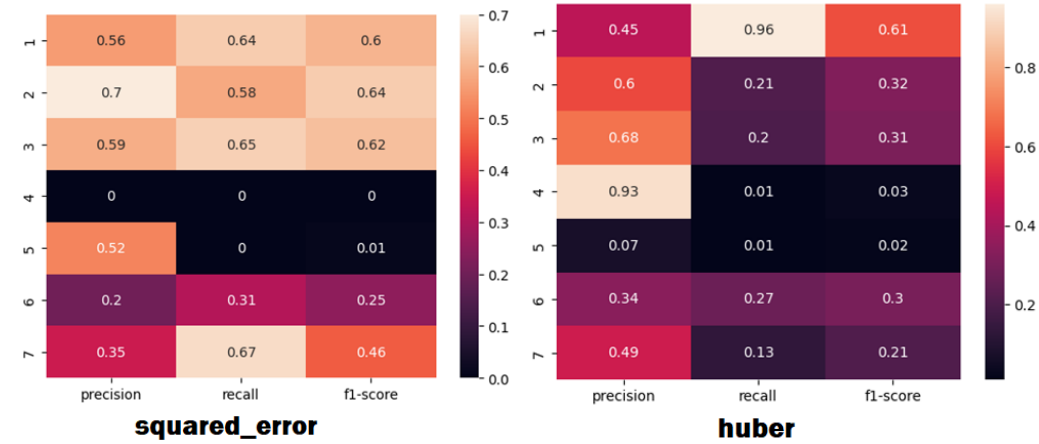
Результаты представлены графически при помощи функции classification_report () из библиотеки scikit-lean.

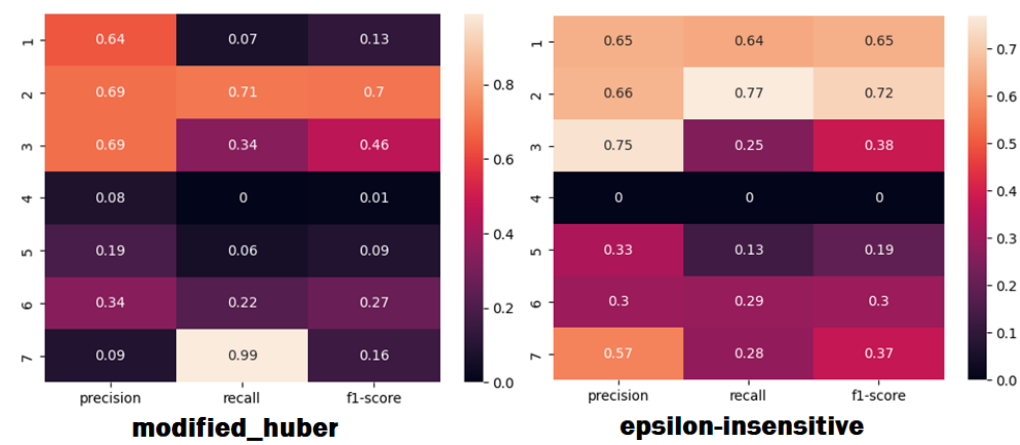
Результаты для SGDClassifier метода

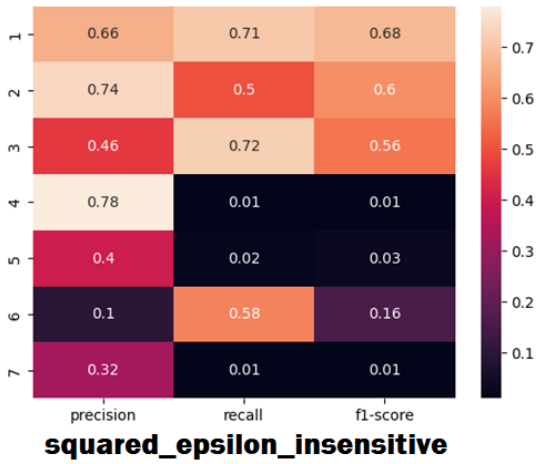
Результаты classification_report () для каждой из функций потерь:





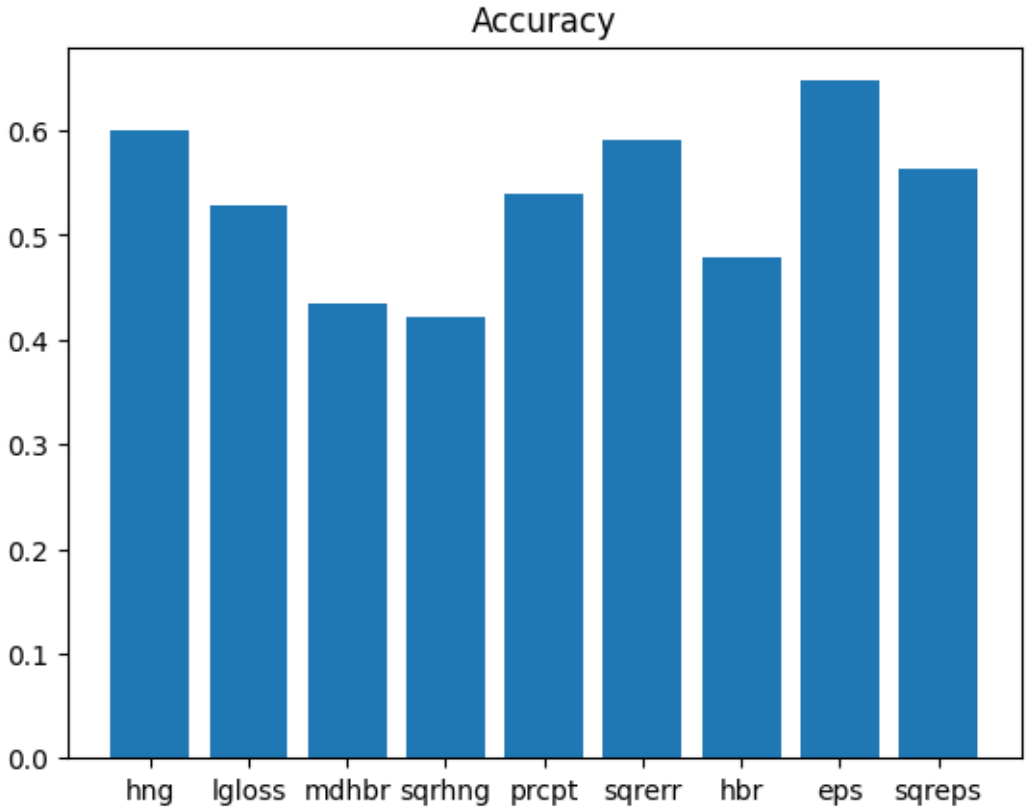






Сравнений моделей по целевой метрике

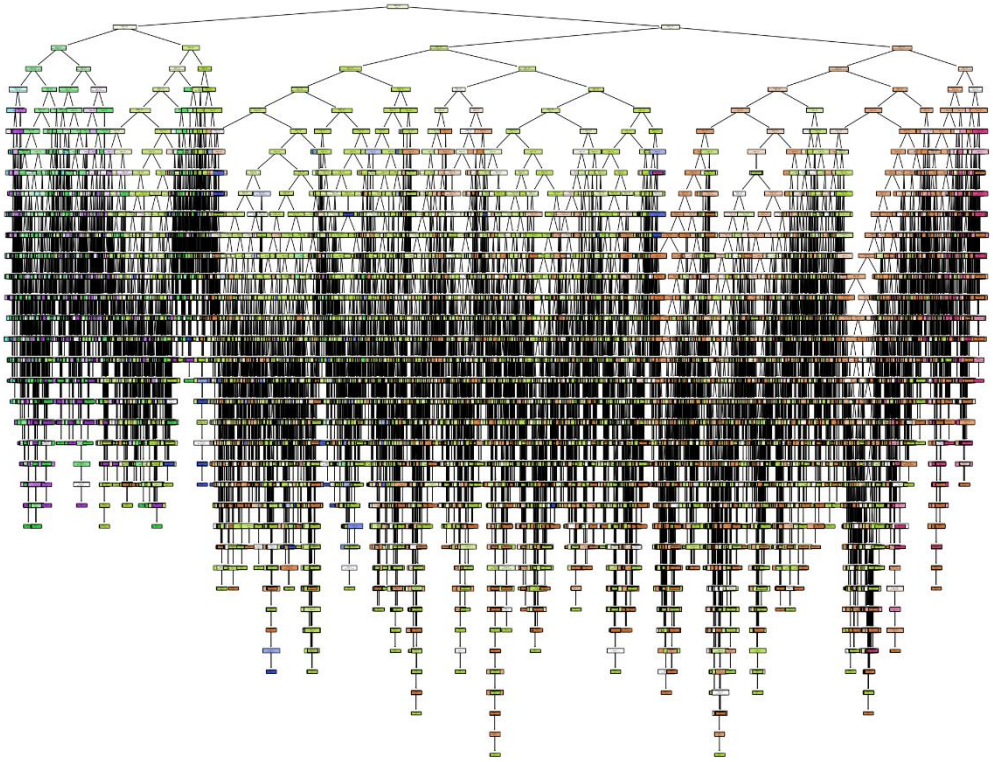
Сравнение производилось по одинаковой тестовой выборке, созданной при помощи метода train_test_split () из библиотеки scikit-lean.



Loss function	Сокращение	Значение
hinge	hng	0.6
squared_hinge	sqrhng	0.42
log_loss	lgloss	0.53
perceptron	prcpt	0.54
squared_error	sqerr	0.59
huber	hbr	0.48
modified_huber	eps	0.43
epsilon_insensitive	eps	0.65
squared_epsilon_insensitive	sqreps	0.56

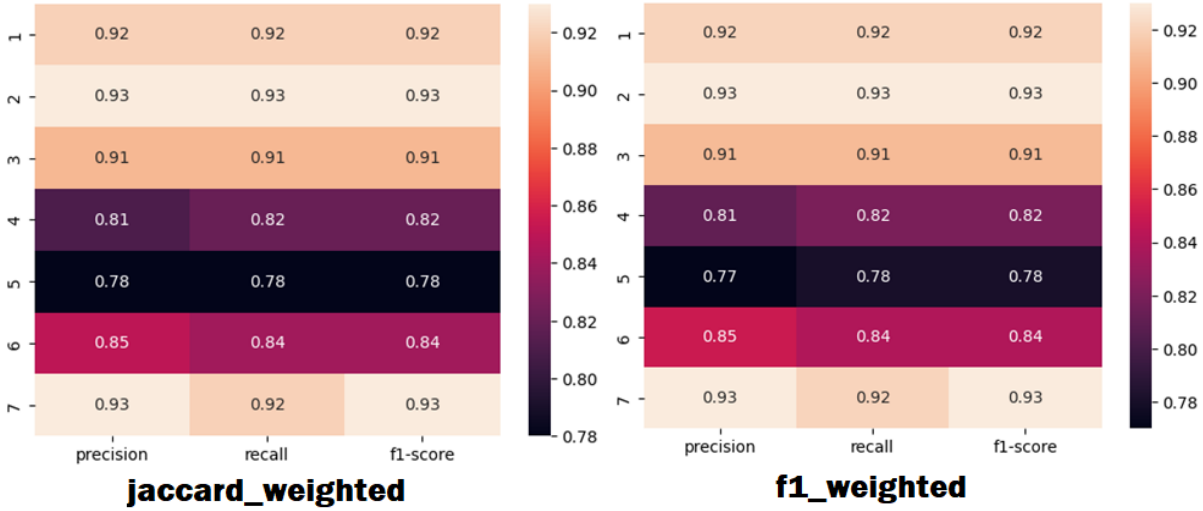
Результаты для перебора по сетке для DecisionTreeClassifier метода

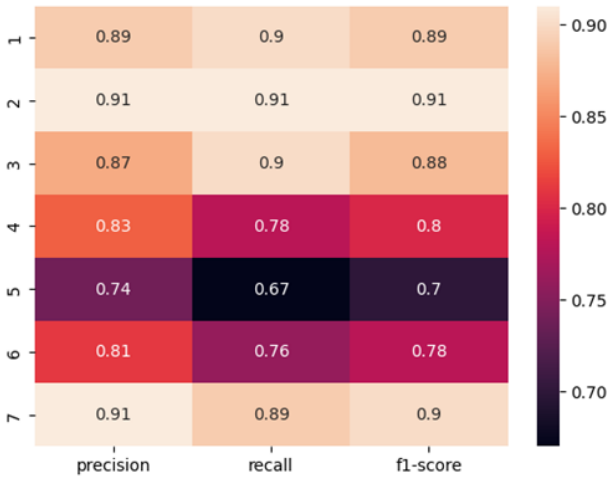
Примерный вид дерева, созданный при обучении алгоритма:



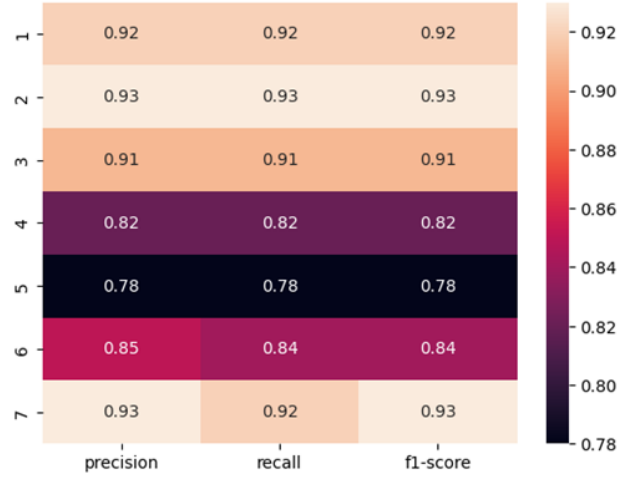
Более подробный вид представлен в прикрепленном файле DecisionTree_overview.pdf

Результат для каждой из метрик, использованных как scoring для GridSearchCV:

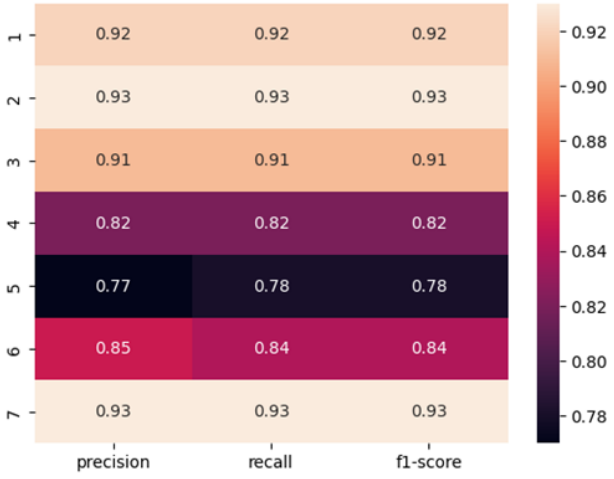




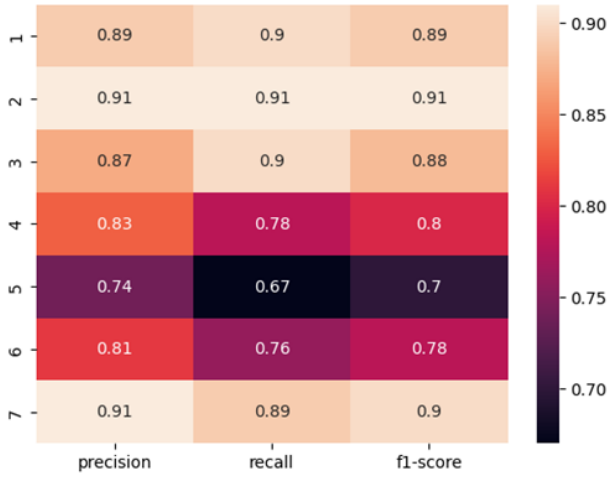
top_k_accuracy



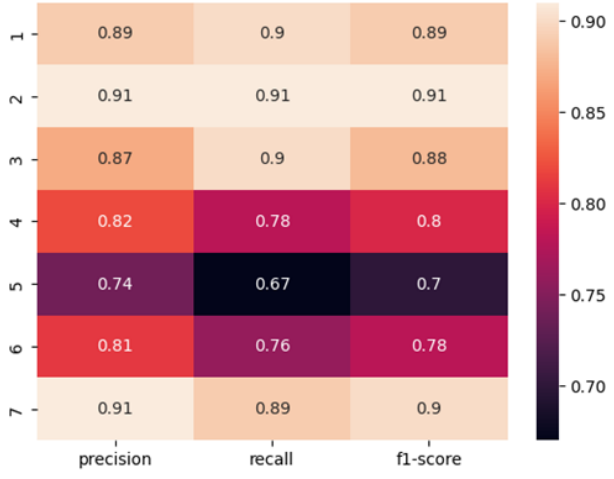
precision_weighted



recall_weighted



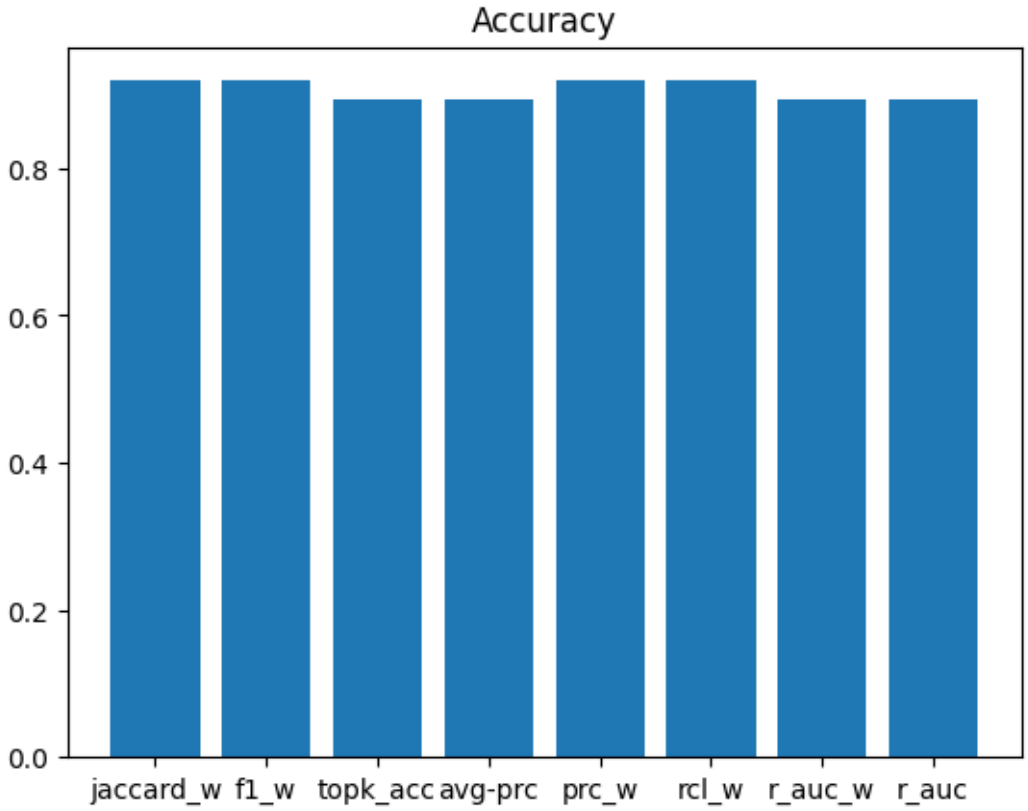
roc_auc_ovr



roc_auc_ovr_weighted

Сравнений моделей по целевой метрике и best fit parameters

Ниже расположены сравнения значения целевой метрики и наилучших параметров, подобранных GrindSearchCV, для каждой из scoring – метрик.



Scoring function	Сокращение	Значение	criterion	min_samples_split
jaccard_weighted	jaccard_w	0.9188141391106043	entropy	2
f1_weighted	f1_w	0.9188055334437727	entropy	2
top_k_accuracy	topk_acc	0.8946537294808632	log_loss	30
precision_weighted	prc_w	0.9191368516167897	log_loss	2
recall_weighted	rcl_w	0.9187280824422882	log_loss	2
roc_auc_ovr	r_auc	0.8946408209806157	log_loss	30
roc_auc_ovr_weighted	r_auc_ovr	0.8943998623093307	entropy	30

Выводы

1. Метрика accuracy_score () является хорошей для выполнения и решения задачи, т. к. она соответствует задаче – наиболее удовлетворяет условию о точности определения классов для данного набора классов, т.е. процент положительно ложно и правдиво определённых результатов относительно всего количества.
2. Функция потерь, обеспечившая наибольшую точность – epsilon_insensitive (). Данный результат стал результатом сравнения с другими функциями потерь, возможно вследствие того, что данная функция в отличие других использует не четкое разделение между классами (например, линиями, если представлять графически), а через определенный зазор, определяемый параметром epsilon, т. е. предоставляя более четкое разделение классов, что положительнее сказывается на результат целевой метрики качества.
3. В задании с перебором параметров по сетке наиболее результативной показала функция precision_weighted (), которая обеспечивает значение целевой метрики качества около 92%, что является высоким результатом и большим относительно метода SGDClassifier, что показывает лучший результат выбранной метрики качества, в следствие, чего имеется возможность предположить, что scoring-метрика была выбрана удачно.