Используя данные датасета реализовать задачу кластеризации. Необходимо выделить страны экспортеры и страны импортеры пшеницы. Выделить лидеров рынка по производству и по внутреннему потреблению. Оценить процент импорта/экспорта относительно внутреннего потребления.

Подготовьте данные - оставьте только страны

Выберите метод кластеризации:

- Иерархическая кластеризация

- DBSCAN

- k-Means

Создайте матрицу ошибок

Оценить качество полученной модели используя precision, recall и F-меру

Оценить модель в целом, не привязываясь к конкретному порогу:

- AUC-ROC (или ROC AUC) — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve )

- Logistic Loss

Оформить результаты.

В процессе работы этапы иллюстрировать графиками графиками.

Дополнительные материалы:

Матрица ошибок - это матрица размером N x N, используемая для оценки эффективности модели классификации, где N - количество целевых классов. Матрица сравнивает фактические целевые значения с предсказанными моделью машинного обучения.

Матрица ошибок  —  это простой способ визуально оценить, насколько часто предсказания классификатора оказываются верными.

Precision, recall и F-мера

Для оценки качества работы алгоритма на каждом из классов по отдельности введем метрики precision (точность) и recall (полнота).

$\large precision = \frac{TP}{TP + FP}$

$\large recall = \frac{TP}{TP + FN}$

Precision можно интерпретировать как долю объектов, названных классификатором положительными и при этом действительно являющимися положительными, а recall показывает, какую долю объектов положительного класса из всех объектов положительного класса нашел алгоритм.

Именно введение precision не позволяет записывать все объекты в один класс, так как в этом случае получается рост уровня False Positive. Recall демонстрирует способность алгоритма обнаруживать данный класс вообще, а precision — способность отличать этот класс от других классов.

Ошибки классификации бывают двух видов: False Positive и False Negative. В статистике первый вид ошибок называют ошибкой I-го рода, а второй — ошибкой II-го рода.

Precision и recall не зависят от соотношения классов и потому применимы в условиях несбалансированных выборок.

F-мера (в общем случае $\ F\_\beta$) — среднее гармоническое precision и recall :

$\large \ F\_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$

$\beta$ в данном случае определяет вес точности в метрике, и при $\beta = 1$ это среднее гармоническое (с множителем 2, чтобы в случае precision = 1 и recall = 1 иметь $\ F\_1 = 1$)

F-мера достигает максимума при полноте и точности, равными единице, и близка к нулю, если один из аргументов близок к нулю.

В sklearn есть удобная функция \_metrics.classificationreport, возвращающая recall, precision и F-меру для каждого из классов, а также количество экземпляров каждого класса.

AUC-ROC и AUC-PR

При конвертации вещественного ответа алгоритма в бинарную метку, необходимо выбрать какой-либо порог, при котором 0 становится 1. Естественным и близким кажется порог, равный 0.5, но он не всегда оказывается оптимальным, например, при отсутствии баланса классов.

Одним из способов оценить модель в целом, не привязываясь к конкретному порогу, является AUC-ROC (или ROC AUC) — площадь (Area Under Curve) под кривой ошибок (Receiver Operating Characteristic curve ). Данная кривая представляет из себя линию от (0,0) до (1,1) в координатах True Positive Rate (TPR) и False Positive Rate (FPR):

$\large TPR = \frac{TP}{TP + FN}$

$\large FPR = \frac{FP}{FP + TN}$

TPR полнота, а FPR показывает, какую долю из объектов negative класса алгоритм предсказал неверно. В идеальном случае, когда классификатор не делает ошибок (FPR = 0, TPR = 1) получается площадь под кривой, равная единице; в противном случае, когда классификатор случайно выдает вероятности классов, AUC-ROC будет стремиться к 0.5, так как классификатор будет выдавать одинаковое количество TP и FP.

Каждая точка на графике соответствует выбору некоторого порога. Площадь под кривой в данном случае показывает качество алгоритма (больше — лучше), кроме этого, важной является крутизна самой кривой — максимизировать TPR, минимизируя FPR, а значит, кривая в идеале должна стремиться к точке (0,1).

Код отрисовки ROC-кривой

Критерий AUC-ROC устойчив к несбалансированным классам и может быть интерпретирован как вероятность того, что случайно выбранный positive объект будет проранжирован классификатором выше (будет иметь более высокую вероятность быть positive), чем случайно выбранный negative объект.

Logistic Loss

$\large logloss = - \frac{1}{l} \cdot \sum\_{i=1}^l (y\_i \cdot log(\hat y\_i) + (1 - y\_i) \cdot log(1 - \hat y\_i))$

здесь $\hat y$ — это ответ алгоритма на $i$-ом объекте, $y$ — истинная метка класса на $i$-ом объекте, а $l$ размер выборки.

Интуитивно можно представить минимизацию logloss как задачу максимизации accuracy путем штрафа за неверные предсказания. Однако необходимо отметить, что logloss крайне сильно штрафует за уверенность классификатора в неверном ответе.

k-Means

Самый известный и часто используемый метод кластеризации называется k-Means (метод k средних). Число k в названии говорит о том, что алгоритм находит k кластеров по данным. Схема работы алгоритма довольно проста. Каждый кластер (группа клиентов, которую мы хотим выделить) задается центром — неким типичным клиентом этой группы.

Описанный алгоритм k-Means очень хорошо работает на несложных данных, например на данных, изображенных ниже. Если же данные сложнее, например требуется выделить кластеры с несколькими «центрами», то этот метод может не очень хорошо справиться с задачей. Кроме того, метод не очень хорошо работает, если число кластеров задано неправильно: тогда он будет разбивать кластеры на более мелкие или объединять несколько кластеров в один. Опишем последовательность работы:

В начале работы алгоритма мы назначаем эти центры случайно, например, выбираем произвольных клиентов в качестве центров.

Затем мы выполняем кластеризацию: каждого клиента из данных записываем в тот кластер, на центр которого он больше всего похож. Получается k групп (кластеров) клиентов.

Теперь, когда клиенты распределены по кластерам, мы находим новые центры кластеров: в каждой группе находим самого репрезентативного клиента.

И далее все повторяется снова: заново распределяем клиентов по кластерам, используя новые центры, заново ищем центры и т.д.

Применение k-Means для кластеризации точек на плоскости. Исходные кластеры (первое изображение) неправильные, но после выбора новых центров (черные звездочки, второе изображение) и перераспределения точек (третье) изображение кластеризация стала информативной.

Описанный алгоритм k-Means очень хорошо работает на несложных данных. Если же данные сложнее, например требуется выделить кластеры с несколькими «центрами», то этот метод может не очень хорошо справиться с задачей. Кроме того, метод не очень хорошо работает, если число кластеров задано неправильно: тогда он будет разбивать кластеры на более мелкие или объединять несколько кластеров в один.

DBSCAN

Другой известный метод — DBSCAN (Density-Based Spatial Clustering of Applications with Noise) — требует задания минимального числа объектов в кластере и минимальной схожести объектов в кластере и затем сам определяет число кластеров. Интересной особенностью DBSCAN является то, что некоторые объекты он называет шумовыми и не относит их ни к одному кластеру. Пример данных, для которых хорошо работает DBSCAN (шумовые точки отмечены черным) и плохо работает k-Means:

Кластеризация точек на два кластера (оранжевый и голубой). Слева метод KMeans, справа DBSCAN. Черный цвет отмечает шумовые точки.

Иерархическая кластеризация

Еще один метод кластеризации, иерархическая кластеризация (Agglomerative clustering), находит вложенные кластеры: например, в кластере «клиенты-студенты» могут быть выделены подкластеры «работающие студенты», «студенты с большими тратами в индустрии развлечений» и «иногородние студенты». Иерархическая кластеризация строит диаграммы следующего вида:

Здесь по вертикали отмечены объекты, и в самом начале работы алгоритма каждый кластер состоит из одного объекта. На каждом шаге алгоритм объединяет два кластера в один: это отмечается соединяющей скобкой. Самая последняя (самая большая) скобка означает объединение всех объектов в один кластер. Любое отсечение этой диаграммы задает одну кластеризацию, например черная линия пересекает семь отрезков и поэтому задает кластеризацию на семь кластеров.