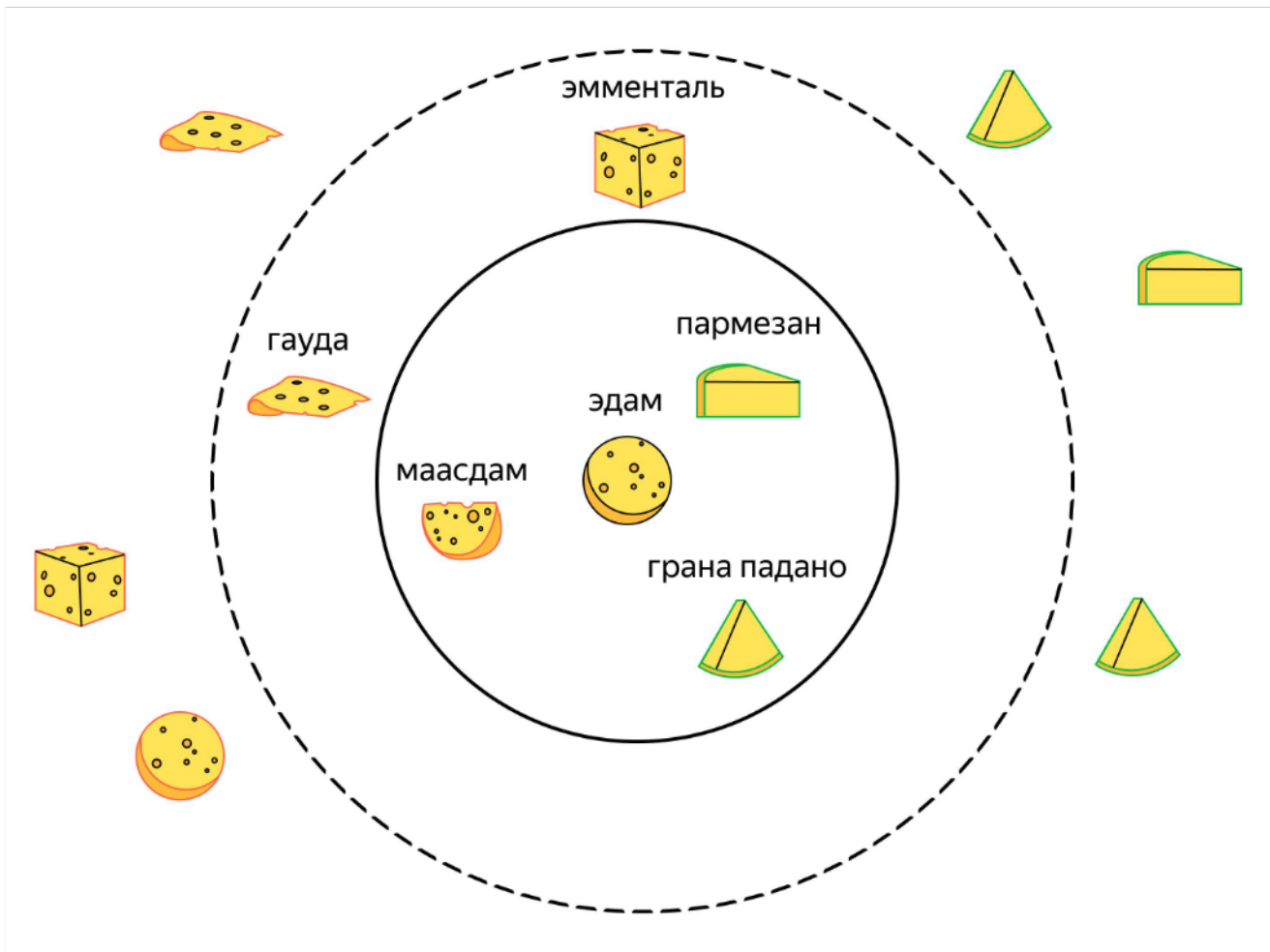


Метод k-ближайших соседей (англ. k-nearest neighbors, kNN) — модель для работы с нелинейными данными, её результаты легко интерпретировать.

На этапе обучения модель kNN сохраняет данные из тренировочной выборки, чтобы затем присваивать новым наблюдениям класс по аналогии. Какой класс у большинства соседей нового объекта — такой будет и у него.

Настройки модели kNN:

- **metric** — выбор метода для расчёта расстояния между объектами в тренировочной и тестовой выборках.
- **n_neighbors**, k — количество ближайших соседей, которых нужно учесть для присвоения класса новым объектам.



Выбор значения k

Нет точного способа выбрать значение k . Его можно выяснить только перебором значений. Есть только общие рекомендации:

- Если значение k слишком мало, то модель переобучится.
- Если значение k слишком велико, тогда модель рискует стать слишком грубой и не будет отражать локальные особенности датасета.

Выбор метода для расчёта расстояния между объектами

- Манхэттенское расстояние выбирают, когда в данных много выбросов, и их влияние на модель нужно минимизировать.

$$d(\vec{a}, \vec{b}) = \sum_{n=1}^n |a_i - b_i|$$

- Евклидово расстояние выбирают, когда выбросов нет или их влияние нужно учитывать.

$$d(\vec{a}, \vec{b}) = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$

Полный список мер для подсчёта расстояний доступен в документации [scipy](#) и [sklearn](#).

Пример использования модели kNN:

```
from sklearn.neighbors import KNeighborsClassifier

# обязательное масштабирование данных перед работой с моделью
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# объявляем классификатор
model = KNeighborsClassifier(
    n_neighbors=10, # количество ближайших соседей для предсказания
    metric='euclidean' # евклидово расстояние
)

model.fit(X_train_scaled, y_train) # обучаем модель

y_pred = model.predict(X_test_scaled) # получаем предсказания
```

Метрики для классификации:

F1-мера (F1-score) — среднее гармоническое между precision и recall, его значение лежит в диапазоне [0,1]. Используется для оценки модели по числу ошибок первого и второго родов. Если хотя бы одна из исходных метрик близка к нулю, то и F1 тоже стремится к нулю.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

```
from sklearn.metrics import f1_score

f1_score(
    y_test, # истинные значения классов
    preds, # предсказанные моделью значения классов
    pos_label='Jazz' # метка положительного класса, по умолчанию равна 1
)
```

Взвешенная F1-мера (F-beta-score) используется, когда одна из метрик более важна.

$$F_{\beta} = (\beta^2 + 1) * \frac{precision * recall}{\beta^2 * precision + recall}$$

Коэффициент β регулирует влияние метрик:

- $\beta > 1$ — больше внимания к recall,
- $0 < \beta < 1$ — больше внимания к precision.

```
from sklearn.metrics import fbeta_score

fbeta_score(
    y_test, # истинные значения классов
    preds, # предсказанные моделью значения классов
    beta=10, # распределение важности между precision и recall
    pos_label='Jazz' # метка положительного класса, по умолчанию равна 1
)
```

True Positive Rate, TPR — доля положительных объектов, правильно предсказанных положительными, или доля верно классифицированных объектов класса 1.

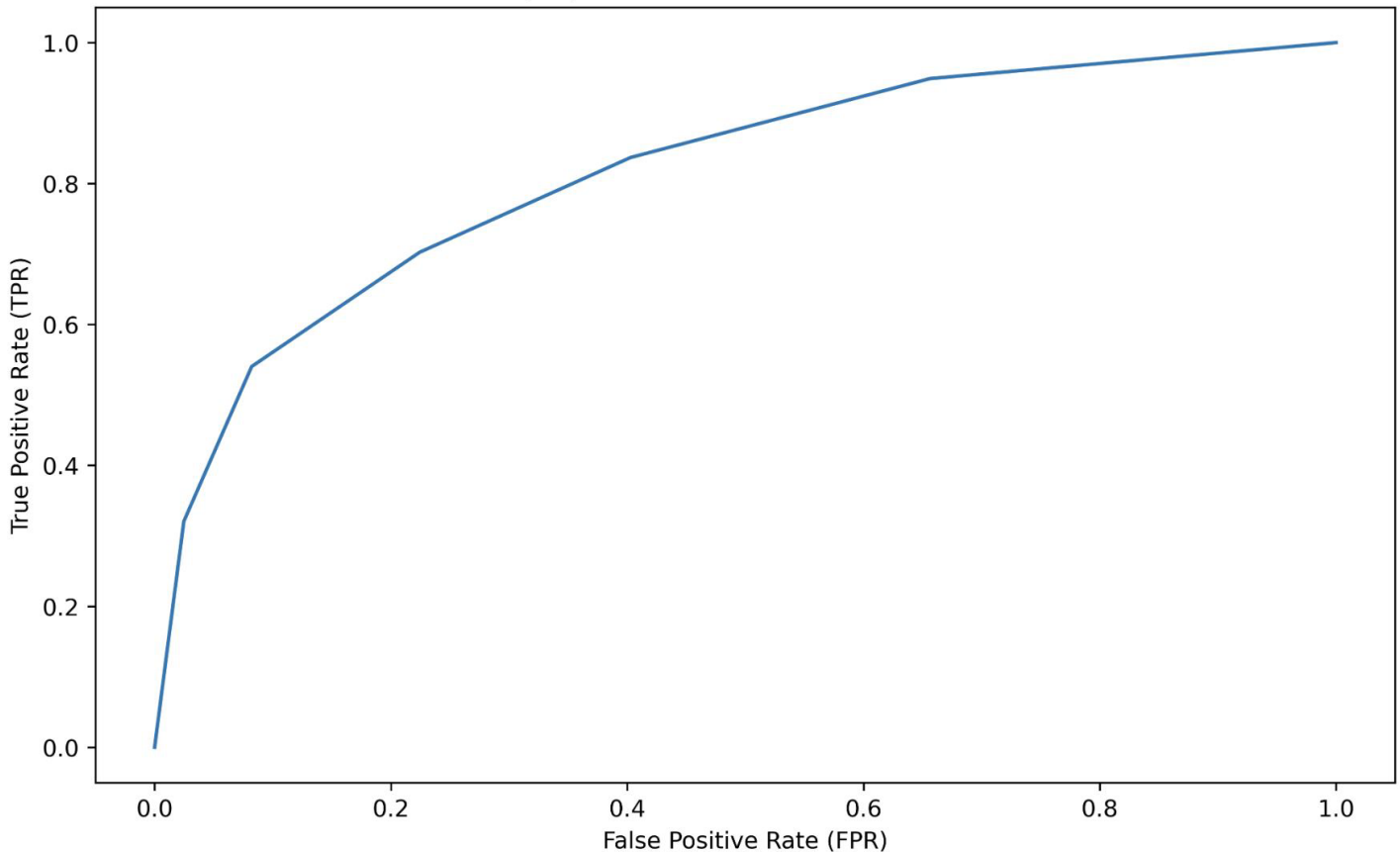
$$TPR = \frac{TP}{TP + FN}$$

False Positive Rate, FPR — доля отрицательных объектов, неверно предсказанных положительными, или доля неверно классифицированных объектов класса 0.

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

ROC-кривая (англ. Receiver Operation Characteristic) — это график, который отображает динамику качества классификации при разных порогах. Он показывает зависимость между TPR и FPR.

График зависимости FPR от TPR



```
from sklearn.metrics import roc_curve

roc_curve(
    y_test, # истинные значения классов
    preds[:,1], # предсказанные моделью вероятности старшего класса
    pos_label='Jazz' # метка положительного класса, по умолчанию равна 1
)
```

Функция создаст три списка — список выбранных функцией порогов вероятностей и списки метрик TPR и FPR для этих порогов. В качестве порогов `roc_curve()` выбирает все уникальные значения вероятностей принадлежности объектов к классу 1.

Также ROC-кривую строит функция `RocCurveDisplay()`. Она делает это двумя способами:

1. В качестве аргументов нужно передать модель и данные из тестовой выборки.

```
from sklearn.metrics import RocCurveDisplay

RocCurveDisplay.from_estimator(model, X_test, y_test)
```

2. В качестве аргументов нужно передать истинные метки классов и вероятности старшего класса, предсказанные моделью.

```
from sklearn.metrics import RocCurveDisplay

RocCurveDisplay.from_predictions(y_test_labels, preds[:,1])
```

ROC-AUC (англ. ROC — receiver operating characteristic, AUC — area under the curve, «площадь под кривой») — площадь под графиком ROC-кривой.

```
from sklearn.metrics import roc_auc_score

roc_auc_score(
    y_test, # истинные значения классов
    preds[:,1], # предсказанные моделью вероятности старшего класса
)
```