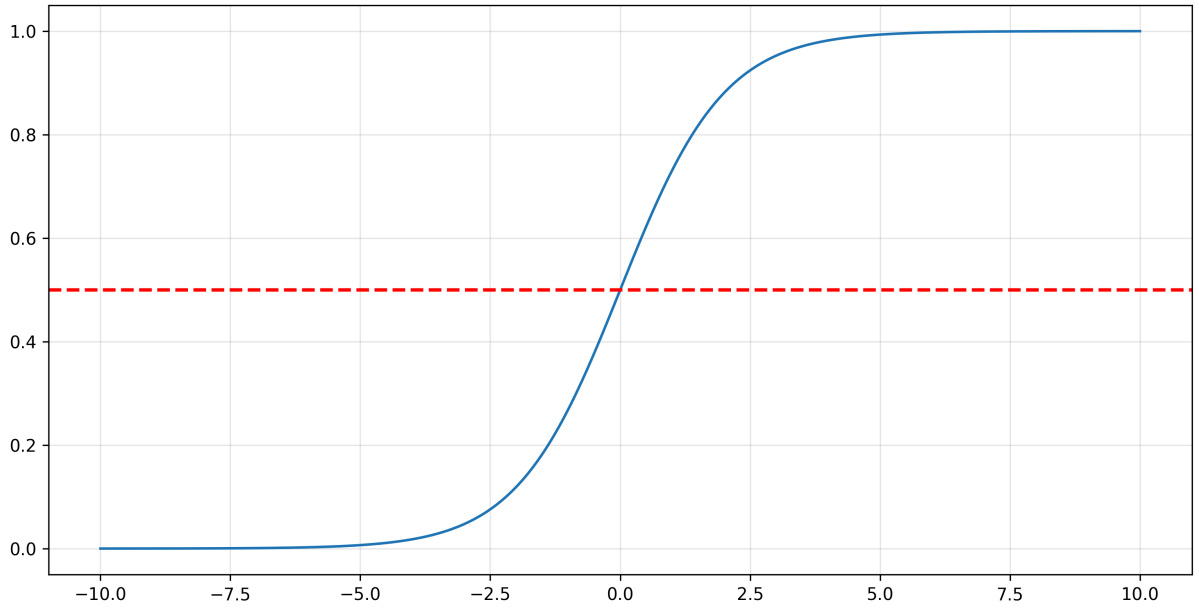
Логистическая регрессия

Логистическая регрессия — статистическая модель, используемая для прогнозирования вероятности возникновения некоторого события путём его сравнения с логистической кривой. Эта регрессия выдаёт ответ в виде вероятности бинарного события (1 или 0).

сигмоида:

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$



Подставим линейную модель $\hat{y} = \beta^\top x + \beta_o$ в сигмоиду, тогда получим

$$\sigma(\beta^\top x + \beta_o) = \frac{1}{1 + e^{-(\beta^\top x + \beta_o)}}$$

Как вероятность

$$P(y = 1 \mid x, \beta) = \sigma(\beta^\top x + \beta_o)$$

$$P(y = 0 \mid x, \beta) = 1 - \sigma(\beta^\top x + \beta_o)$$

Функция правдоподобия (likelihood)

Для одного объекта вероятность наблюдения y_i

$$P(y_i \mid x_i, \beta) = \left[\sigma(\beta^\top x_i)\right]^{y_i} \left[1 - \sigma(\beta^\top x_i)\right]^{1-y_i}$$

если y_i = 1, то вторая скобка исчезает, если y_i = 0, аналогично исчезает первая скобка.

Для всего датасета это:

$$\mathcal{L}(\beta) = \prod_{i=1}^{m} \left[\sigma(\beta^\top x_i)\right]^{y_i} \left[1 - \sigma(\beta^\top x_i)\right]^{1-y_i}$$

(функция правдободобия), мы должны ее **МАКСИМИЗИРОВАТЬ.** То есть $\max_\beta \ \mathcal{L}(\beta)$

Функция потерь (log-loss)

Функция потерь выводится из функции правдоподобия. Логарифм не меняет точку максимума, но превращает произведение в сумму:

$$\ell(\beta) = log\mathcal{L}(\beta)$$

предположим, что $y_i \mid x_i \sim \text{Bernoulli}(p_i)$, где $p_i = \sigma(\beta^\top x_i)$,
тогда $\mathcal{L}(\beta) = \prod_{i=1}^{m} p_i^{y_i} \left[1 - p_i\right]^{1-y_i}$

$$\ell(\beta) = log\mathcal{L}(\beta)$$
$$\ell(\beta) = log(\prod_{i=1}^{m} p_i^{y_i} \left[1 - p_i\right]^{1-y_i})$$

Так как логарифм произведения это сумма логарифмов, то

$$\ell(\beta) = \sum_{i=1}^{m} \left(y_i \log(p_i) + (1 - y_i) \log(1 - p_i)\right)$$
$$\ell(\beta) = \sum_{i=1}^{m} \left(y_i \log \sigma(\beta^\top x_i) + (1 - y_i) \log\left(1 - \sigma(\beta^\top x_i)\right)\right)$$

Однако, мы должны минимизировать функцию потерь, а не максимизировать, тогда

$$J(\beta) = -\ell(\beta)$$

$J(\beta)$ (аналогично $L(\beta)$) - функция потерь

$$J(\beta) = -\frac{1}{m} \sum_{i=1}^{m} \left(y_i \log \sigma(\beta^\top x_i) + (1 - y_i) \log\left(1 - \sigma(\beta^\top x_i)\right)\right)$$

(можно заметить, что $\min J(\beta) \iff \max \ell(\beta)$)

Градиент функции потерь

$$\nabla_\beta \mathcal{L}(\beta) = \sum_{i=1}^{n} \left(\sigma(x_i^\top \beta) - y_i\right) x_i$$

Шаг градиентного спуска

$$\beta^{(t+1)} = \beta^{(t)} - \eta \nabla_\beta \mathcal{L}(\beta^{(t)})$$

η - learning rate

Метрики

Матрица ошибок (Confusion matrix)

Confusion Matrix (матрица ошибок) - это таблица, которая показывает, как модель классификации ошибается и где именно.

Как выглядит:

	Предсказал 0	**Предсказал 1**
Истина = 0	TN (True Negative)	FP (False Positive)
Истина = 1	FN (False Negative)	TP (True Positive)

True / False - угадал модель или нет

Positive (1) / Negative (0) - что модель сказала, а не что на самом деле

Аналогия:

ML	Statistics
FP	Type I error
FN	Type II error

Из матрицы ошибок, выводятся метрики:

Accuracy = $\frac{TN+TP}{\text{total}}$ (доля всех предсказаний, которые модель сделала правильно)

Precision (точность) = $\frac{TP}{\text{TP} + \text{FP}}$ (из всех объектов, которые модель назвала положительными, какая доля действительно положительная)

Recall (**полнота**) = $\frac{TP}{\mathrm{TP}+\mathrm{FN}}$ (из всех реальных положительных объектов, какую долю модель смогла найти)

F1-Score = $2 \cdot \frac{\mathrm{Precision} \cdot \mathrm{Recall}}{\mathrm{Precision}+\mathrm{Recall}}$ (баланс между precision и recall, их гармоническое среднее)

Регуляризация

В логистической регрессии мы минимизируем log loss + штраф:

$$J(\beta) = \underbrace{-\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log \sigma(\beta^\top x_i) + (1-y_i)\log\left(1-\sigma(\beta^\top x_i)\right)\right)}_{\text{loss}} + \underbrace{\lambda \Omega(\beta)}_{\text{regularization}}$$

L2 - регуляризация (Ridge)

$$\Omega(\beta) = \|\beta\|_2^2 = \beta^\top \beta = \sum_{j=1}^{p} \beta_j^2$$

тогда

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log \sigma(\beta^\top x_i) + (1-y_i)\log\left(1-\sigma(\beta^\top x_i)\right)\right) + \lambda \sum_{j=1}^{p} \beta_j^2$$

L1 - регуляризация (Lasso)

$$\Omega(\beta) = \|\beta\|_1 = \sum_{j=1}^{p} |\beta_j|$$

тогда

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log \sigma(\beta^\top x_i) + (1-y_i)\log\left(1-\sigma(\beta^\top x_i)\right)\right) + \lambda \sum_{j=1}^{p} |\beta_j|$$

ElasticNet регуляризация

$$\Omega(\beta) = \alpha\|\beta\|_1 + (1-\alpha)\|\beta\|_2^2 = \alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2, \quad \alpha \in [0,1]$$

тогда

$$J(\beta) = -\frac{1}{m}\sum_{i=1}^{m}\left(y_i \log \sigma(\beta^\top x_i) + (1-y_i)\log\left(1-\sigma(\beta^\top x_i)\right)\right) + \lambda(\alpha \sum_{j=1}^{p} |\beta_j| + (1-\alpha) \sum_{j=1}^{p} \beta_j^2)$$

Sklearn

```
# отделяем признаки и целевую переменную

X = df.drop('test_results', axis = 1)
y = df['test_results']


# мастабирование признаков

from sklearn.model_selection import train_test_split
from sklearn.preproccesing import StandardScaler
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1, random_state=101)

scaler = StandardScaler()
scaled_X_train = scaler.fit_transform(X_train)
scaled_X_test = scaler.transform(X_test)

# модель логистической регрессии

from sklearn.linear_model import LogisticRegression
log_model = LogisticRegression()
log_model.fit(scaled_X_train, y_train) # обучение

y_pred = log_model.predict(scaled_X_test) # 0 и 1
y_pred_prob = log_model.predict_proba(scaled_X_test) # вероятности принадлежности к классам 0 и 1

# метрики

from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
from sklearn.metrics import ConfusionMatrixDisplay
accuracy_score(y_test, y_pred) # вернет accuracy, например 0.93
confusion_matrix(y_test, y_pred) # вернет матрицу 2x2

ConfusionMatrixDisplay.from_estimator(log_model,scaled_X_test,y_test) # нарисует уже типа heatmap для confusion matrix

print(classification_report(y_test, y_pred)) # вернет таблицу recall, f1, precision..

from sklearn.metrics import precision_score, recall_score

precision_score(y_test, y_pred)
recall_score(y_test, y_pred)
```