

Baseline Model

Цель данного шага — построить модель бинарной классификации для оценки кредитного риска клиента.

Модель должна определять, относится ли клиент к:

- **низкому кредитному риску (0)**
- **высокому кредитному риску (1)**

В рамках данного ноутбука рассматривается **baseline-модель**, которая используется как отправная точка для дальнейших улучшений и сравнения с более сложными алгоритмами.

```
In [ ]: import joblib
from sklearn.linear_model import LogisticRegression
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import os
from sklearn.metrics import (
    roc_auc_score,
    f1_score,
    confusion_matrix
)
```

Загрузка данных для обучение модели

```
In [2]: data = joblib.load(
    "C:\\Users\\nurs\\OneDrive\\Рабочий стол\\Credit scoring\\lr_preprocessing.joblib"
)
X_train = data["X_train"]
X_test = data["X_test"]
y_train = data["y_train"]
y_test = data["y_test"]

preprocessor = data["preprocessor"]
```

```
In [3]: def evaluate_classification_model(
    y_true,
    y_proba,
    threshold=0.5,
    model_name="Model"
):
    """
    Универсальная функция оценки бинарной классификации

    Parameters:
    y_true : array-like
        Истинные метки (0/1)
    y_proba : array-like
        Вероятности класса 1
    threshold : float, default=0.5
    """
```

```

        Порог классификации
model_name : str
        Название модели (для визуализаций)

Returns:
dict with ROC-AUC and F1-score
"""
y_pred = (y_proba >= threshold).astype(int)
roc_auc = roc_auc_score(y_true, y_proba)
f1 = f1_score(y_true, y_pred)
print(f"{model_name}")
print("-" * len(model_name))
print(f"ROC-AUC : {roc_auc:.4f}")
print(f"F1-score: {f1:.4f}")
print(f"Threshold: {threshold}")
cm = confusion_matrix(y_true, y_pred)
plt.figure(figsize=(5, 4))
sns.heatmap(
    cm,
    annot=True,
    fmt="d",
    cmap="Blues",
    cbar=False
)
plt.xlabel("Predicted label")
plt.ylabel("True label")
plt.title(f"Confusion Matrix - {model_name}")
plt.show()
return {
    "model": model_name,
    "roc_auc": roc_auc,
    "f1_score": f1,
    "threshold": threshold
}

```

Перед обучением модели данные были:

- разделены на обучающую и тестовую выборки
- предварительно обработаны с помощью пайплайна предобработки

```

In [4]: X_train_lr = preprocessor.fit_transform(X_train)
        X_test_lr  = preprocessor.transform(X_test)

```

Baseline модель

В качестве baseline-модели была выбрана **логистическая регрессия**, так как:

- она проста в интерпретации
- хорошо работает как базовый классификатор
- позволяет оценить качество признаков

```

In [5]: lr_baseline = LogisticRegression(
        max_iter=3000,
        class_weight="balanced",
        solver="liblinear",
        random_state=42

```

```
)  
lr_baseline.fit(X_train_lr, y_train)
```

Out[5]:

▼ **LogisticRegression** ⓘ ⓘ

► Parameters

In [6]: `y_proba = lr_baseline.predict_proba(X_test_lr)[:, 1]`

Метрики оценки качества

In [7]:

```
results_lr = evaluate_classification_model(  
    y_true=y_test,  
    y_proba=y_proba,  
    threshold=0.5,  
    model_name="Logistic Regression (Baseline)"  
)
```

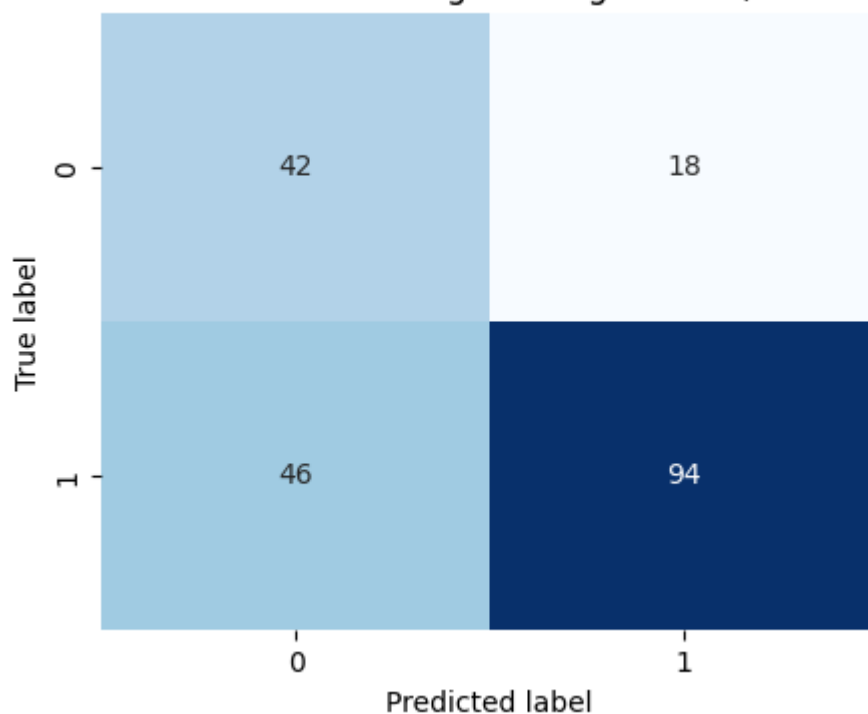
Logistic Regression (Baseline)

ROC-AUC : 0.7589

F1-score: 0.7460

Threshold: 0.5

Confusion Matrix — Logistic Regression (Baseline)



По результатам тестирования baseline-модели были получены следующие значения:

- **ROC-AUC:** 0.7589
- **F1-score:** 0.7460
- **Порог классификации:** 0.5

Полученные результаты демонстрируют, что модель уже на базовом уровне способна достаточно хорошо разделять клиентов по уровню кредитного риска.

Сохранение модели

Для дальнейшего использования и воспроизводимости эксперимента baseline-модель была сохранена вместе с:

- обученной моделью
- объектом предобработки
- метриками качества

```
In [8]: lr_baseline_artifact = {
        "model": lr_baseline,
        "preprocessor": preprocessor,
        "metrics": {
            "roc_auc": 0.7589,
            "f1_score": 0.7460,
            "threshold": 0.5
        }
    }
    joblib.dump(
        lr_baseline_artifact,
        "lr_baseline_model.joblib"
    )
```

```
Out[8]: ['lr_baseline_model.joblib']
```

```
In [9]: os.makedirs("models", exist_ok=True)

    joblib.dump(
        lr_baseline_artifact,
        "models/lr_baseline_model.joblib"
    )
```

```
Out[9]: ['models/lr_baseline_model.joblib']
```

Вывод

Baseline-модель логистической регрессии показала стабильное качество и может быть использована как отправная точка для дальнейшего улучшения.