# Credit Scoring API — Документация

**Ансамблевая система кредитного скоринга с интерпретацией решений (SHAP) и генерацией PDF-отчётов**

## 📌 Описание

Это **аналитическая система кредитного скоринга**, построенная на **ансамблевой модели машинного обучения** (Random Forest + XGBoost + CatBoost). Проект реализован на Python с использованием FastAPI, Streamlit, SHAP и scikit-learn.

Система позволяет:

* Прогнозировать риск дефолта
* Объяснять решения с помощью SHAP
* Генерировать PDF-отчёты
* Дообучать модель на основе обратной связи
* Сравнивать производительность моделей

## 🚀 Структура проекта

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

credit\_scoring/

├── app/

│ ├── main.py # FastAPI-сервер

│ └── services/

│ ├── model\_training.py # Обучение моделей

│ ├── retrain.py # Дообучение

│ └── utils.py # Прогноз и объяснение

├── frontend/

│ └── app.py # Streamlit-интерфейс

├── models/ # Сохранённые модели

├── reports/ # PDF-отчёты

├── data/

│ ├── credit\_risk\_dataset.csv # Исходные данные

│ └── feedback.jsonl # Обратная связь

├── shared/

│ ├── config.py # Глобальные пути и настройки

│ ├── data\_processing.py # Предобработка

│ ├── models.py # Pydantic-модели

│ └── auth.py # Авторизация

└── README.md

## 🔐 Авторизация

Все эндпоинты защищены базовой аутентификацией:

* **Логин:** admin
* **Пароль:** password123

## 🌐 API Endpoints

### 1. GET / — Главная страница

**Описание:** Проверка работоспособности API  
**URL:** http://localhost:8000/

bash

1

curl -u admin:password123 http://localhost:8000/

**Ответ:**

json

1

2

3

⌄

{

"message": "Добро пожаловать в Credit Scoring API"

}

### 2. POST /predict — Прогноз

**Описание:** Предсказывает, вернёт ли клиент кредит  
**URL:** http://localhost:8000/predict

**Пример запроса:**

bash

1

2

3

4

5

6

7

8

9

10

11

12

13

curl -u admin:password123 -X POST http://localhost:8000/predict -H "Content-Type: application/json" -d '{

"person\_age": 35,

"person\_income": 75000,

"person\_home\_ownership": "RENT",

"person\_emp\_length": 5.0,

"loan\_intent": "DEBTCONSOLIDATION",

"loan\_grade": "B",

"loan\_amnt": 20000,

"loan\_int\_rate": 9.5,

"loan\_percent\_income": 0.27,

"cb\_person\_default\_on\_file": "N",

"cb\_person\_cred\_hist\_length": 4

}'

**Ответ:**

json

1

2

3

4

5

6

7

⌄

{

"prediction": 0,

"status": "repaid",

"decision": "approve",

"probability\_repaid": 0.88,

"probability\_default": 0.12

}

### 3. POST /explain — Объяснение с SHAP

**Описание:** Возвращает прогноз + объяснение решения  
**URL:** http://localhost:8000/explain

**Пример запроса:**

bash

1

2

3

4

5

6

7

8

9

10

11

12

13

curl -u admin:password123 -X POST http://localhost:8000/explain -H "Content-Type: application/json" -d '{

"person\_age": 35,

"person\_income": 75000,

"person\_home\_ownership": "RENT",

"person\_emp\_length": 5.0,

"loan\_intent": "DEBTCONSOLIDATION",

"loan\_grade": "B",

"loan\_amnt": 20000,

"loan\_int\_rate": 9.5,

"loan\_percent\_income": 0.27,

"cb\_person\_default\_on\_file": "N",

"cb\_person\_cred\_hist\_length": 4

}'

**Ответ:**

json

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

⌄

⌄

⌄

⌄

{

"prediction": 0,

"status": "repaid",

"decision": "approve",

"probability\_repaid": 0.88,

"probability\_default": 0.12,

"explanation": {

"base\_value": 0.12,

"shap\_values": [

{"feature": "loan\_grade\_B", "value": -0.15},

{"feature": "person\_income", "value": -0.12},

{"feature": "loan\_int\_rate", "value": 0.08}

],

"summary": [

"loan\_grade\_B: ↓ риск (-0.150)",

"person\_income: ↓ риск (-0.120)",

"loan\_int\_rate: ↑ риск (+0.080)"

],

"shap\_image\_base64": "iVBORw0KGgoAAAANSUhEUgAAA..."

}

}

### 4. POST /report — Генерация PDF-отчёта

**Описание:** Создаёт PDF-отчёт с графиком SHAP и объяснением  
**URL:** http://localhost:8000/report

**Пример запроса:**

bash

1

2

3

4

5

6

7

8

9

10

11

12

13

curl -u admin:password123 -X POST http://localhost:8000/report -H "Content-Type: application/json" -d '{

"person\_age": 35,

"person\_income": 75000,

"person\_home\_ownership": "RENT",

"person\_emp\_length": 5.0,

"loan\_intent": "DEBTCONSOLIDATION",

"loan\_grade": "B",

"loan\_amnt": 20000,

"loan\_int\_rate": 9.5,

"loan\_percent\_income": 0.27,

"cb\_person\_default\_on\_file": "N",

"cb\_person\_cred\_hist\_length": 4

}'

**Ответ:**

json

1

2

3

⌄

{

"report\_path": "reports/explanation\_report.pdf"

}

### 5. POST /train-final — Обучение ансамбля

**Описание:** Обучает ансамблевую модель (RF + XGBoost + CatBoost)  
**URL:** http://localhost:8000/train-final

bash

1

curl -u admin:password123 -X POST http://localhost:8000/train-final

**Ответ:**

json

1

2

3

4

⌄

{

"model": "Ensemble (RF + XGBoost + CatBoost)",

"accuracy": 0.934

}

### 6. GET /compare — Сравнение моделей

**Описание:** Возвращает точность всех доступных моделей  
**URL:** http://localhost:8000/compare

bash

1

curl -u admin:password123 http://localhost:8000/compare

**Ответ:**

json

1

2

3

4

5

6

⌄

⌄

{

"models": [

{"model": "RandomForest", "accuracy": 0.912},

{"model": "Ensemble", "accuracy": 0.934}

]

}

### 7. POST /feedback — Обратная связь

**Описание:** Сохраняет фактический результат кредита для дообучения  
**URL:** http://localhost:8000/feedback

bash

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

curl -u admin:password123 -X POST http://localhost:8000/feedback -H "Content-Type: application/json" -d '{

"person\_age": 35,

"person\_income": 75000,

"person\_home\_ownership": "RENT",

"person\_emp\_length": 5.0,

"loan\_intent": "DEBTCONSOLIDATION",

"loan\_grade": "B",

"loan\_amnt": 20000,

"loan\_int\_rate": 9.5,

"loan\_percent\_income": 0.27,

"cb\_person\_default\_on\_file": "N",

"cb\_person\_cred\_hist\_length": 4,

"predicted\_status": 0,

"actual\_status": 1

}'

**Ответ:**

json

1

2

3

4

⌄

{

"status": "success",

"message": "Обратная связь сохранена"

}

### 8. POST /retrain — Дообучение модели

**Описание:** Переобучает модель на основе фидбэков  
**URL:** http://localhost:8000/retrain

bash

1

curl -u admin:password123 -X POST http://localhost:8000/retrain

**Ответ:**

json

1

2

3

4

5

6

7

⌄

{

"status": "retrained",

"samples\_used": 12,

"model\_path": "models/ensemble\_model.pkl",

"accuracy\_on\_feedback": 0.917,

"feature\_count": 27

}

## 🧠 Модули и функции

### shared/models.py

**Описание:** Pydantic-модели для валидации данных

* LoanRequest — входные данные заемщика
* FeedbackRequest — обратная связь (расширяет LoanRequest)

python

1

2

3

4

5

⌄

class LoanRequest(BaseModel):

person\_age: int

person\_income: int

person\_home\_ownership: str

...

### shared/data\_processing.py

**Описание:** Предобработка данных

* feature\_engineering() — добавляет loan\_to\_income\_ratio
* preprocess\_data\_for\_prediction() — OHE + выравнивание колонок
* CATEGORIES — словарь для OHE
* EXPECTED\_FEATURES — список фичей после OHE

### shared/config.py

**Описание:** Глобальные пути и настройки

python

1

2

3

4

5

ROOT\_DIR = Path(\_\_file\_\_).parent.parent

MODELS\_DIR = ROOT\_DIR / "models"

REPORTS\_DIR = ROOT\_DIR / "reports"

DATA\_SOURCE = DATA\_DIR / "credit\_risk\_dataset.csv"

API\_AUTH = ("admin", "password123")

### shared/auth.py

**Описание:** Авторизация через HTTP Basic

python

1

2

3

4

⌄

⌄

def verify\_credentials(credentials: HTTPBasicCredentials = Depends(security)):

if credentials.username != "admin" or credentials.password != "password123":

raise HTTPException(401, "Неверные учётные данные")

return credentials.username

### services/model\_training.py

**Описание:** Обучение ансамблевой модели

* train\_ensemble\_model(X, y) — создаёт VotingClassifier
* Сохраняет модель, feature\_names.pkl, background\_data.pkl

### services/retrain.py

**Описание:** Дообучение модели на фидбэках

* Читает feedback.jsonl
* Валидирует данные
* Дообучает модель
* Сохраняет обновлённую модель

### services/utils.py

**Описание:** Ядро прогнозирования и объяснения

* explain\_prediction() — использует shap.Explainer для ансамбля
* Обработка shap\_values и генерация waterfall-графика
* Возврат shap\_image\_base64 для вставки в PDF

### services/reporting.py

**Описание:** Генерация PDF-отчётов

* generate\_explanation\_pdf() — использует weasyprint и Jinja2
* Встраивает base64-изображение SHAP
* Поддерживает русский язык

### frontend/app.py

**Описание:** Streamlit-интерфейс

* 3 вкладки: Прогноз, Обучение, Сравнение
* Авторизация
* Генерация PDF
* Обратная связь
* Дообучение

## 🛠 Установка и запуск

bash

1

2

3

4

5

6

7

8

# 1. Установка зависимостей

pip install fastapi uvicorn streamlit shap weasyprint jinja2 joblib scikit-learn xgboost catboost

# 2. Запуск FastAPI

cd app && uvicorn main:app --reload

# 3. Запуск Streamlit

cd frontend && streamlit run app.py

## 📎 Требования

* Python 3.8+
* FastAPI
* Streamlit
* SHAP
* scikit-learn
* XGBoost, CatBoost
* weasyprint (для PDF)

## 🎓 Дипломный проект

Реализовано:

* Ансамблевые модели
* Интерпретируемость (SHAP)
* PDF-отчёты
* Обратная связь и дообучение
* Полноценный API и UI

© 2025 | Дипломный проект по машинному обучению