FirstHome Advisor

# Klasy

##### UserData

Opis: reprezentuje dane finansowe i osobiste wprowadzone przez użytkownika.

Atrybuty:

* age: int
* employment\_type: str
* monthly\_income: float
* monthly\_expenses: float
* existing\_loans: float
* own\_contribution: float
* property\_value: float
* dependents: int

Metody:

to\_dataframe() -> pd.DataFrame – przygotowanie danych do klasyfikatora

##### RiskClassifier

Opis: opakowanie modelu ML do oceny ryzyka użytkownika.

Atrybuty:

* model – np. obiekt scikit-learn
* preprocessor – pipeline / scaler / encoder

Metody:

predict(user\_data: UserData) -> str – zwraca klasyfikację (np. "gotowy")

predict\_proba(user\_data: UserData) -> float – zwraca prawdopodobieństwo ryzyka

##### CreditCalculator

Opis: wykonuje obliczenia pomocnicze na podstawie danych użytkownika.

Metody:

calculate\_dti(user\_data: UserData) -> float

calculate\_max\_credit(user\_data: UserData) -> float

recommendations(user\_data: UserData) -> list[str]

##### AssessmentResult

Opis: struktura wyniku oceny.

Atrybuty:

* risk\_category: str
* risk\_score: float
* max\_loan: float
* dti: float
* recommendations: list[str]

##### AppController

Opis: koordynator logiki aplikacji, spina komponenty i zwraca gotowy wynik do UI.

Metody:

run\_assessment(user\_data: UserData) -> AssessmentResult

##### MainUI

Opis: warstwa interfejsu

Metody:

render\_form() -> UserData

display\_result(result: AssessmentResult)

# Wzorzec architektoniczny: MVC z logiką ML w serwisie

Model – UserData, AssessmentResult, dane ML (RiskClassifier)

Service – CreditCalculator, AppController

View/Interface – MainUI

# Przepływ danych

Użytkownik wypełnia formularz → MainUI tworzy UserData

AppController.run\_assessment():

używa RiskClassifier do predykcji

używa CreditCalculator do obliczeń

tworzy AssessmentResult

MainUI prezentuje wynik użytkownikowi

# Diagram UML

Obraz zawierający tekst, diagram, Czcionka, paragon

Opis wygenerowany automatycznie