Procjena financijskog rizika defaulta

Bayesovski optimizirani gradient boosting i objašnjivost pomoću SHAP-a

Matej Čiček

23. siječnja 2026.

Sažetak. U radu je razvijen reproducibilan sustav za procjenu rizika financijskog neispunjenja obveza (default/bankruptcy) koji integrira heterogene izvore podataka (CSV i XLSX), provodi naprednu predobradu (MICE imputacija, robustan tretman outliera), trenira i optimizira modele (logistička regresija, random forest i XGBoost) te uvodi Bayesovu optimizaciju hiperparametara pomoću Optuna okvira. Sustav uključuje rigoroznu validaciju (nested cross-validation), kalibraciju vjerojatnosti (izotonička kalibracija), interpretabilnost SHAP metodom te segmentaciju uzoraka (K-means). Integrirani podaci i metrike pohranjuju se u SQLite bazu, a rezultati i predikcije izlažu se kroz REST API (FastAPI), što omogućuje jednostavnu integraciju u druge aplikacije.

Ključne riječi: kreditni rizik, default, XGBoost, Optuna, SHAP, kalibracija, K-means, reproducibilnost, SQLite, FastAPI.

Sadržaj

Popis slika

Popis tablica

Popis kratica

AUC	Area Under ROC Curve
PR-AUC	Area Under Precision–Recall Curve
CV	Cross-Validation
MICE	Multiple Imputation by Chained Equations
SHAP	SHapley Additive exPlanations
API	Application Programming Interface
DVC	Data Version Control
PCA	Principal Component Analysis

Poglavlje 1

Uvod

1.1 Motivacija

Procjena financijskog rizika neispunjenja obveza (defaulta) predstavlja središnji problem u području kreditiranja, upravljanja portfeljem i regulatorne usklađenosti financijskih institucija. Odluke o odobravanju kredita, određivanju kamatnih stopa i alokaciji kapitala izravno ovise o sposobnosti modela da precizno i pouzdano procijeni vjerojatnost nastupa defaulta. Za razliku od općih klasifikacijskih problema, u kreditnom riziku nije dovoljna samo binarna odluka (odobriti/odbiti), već je nužna probabilistička predikcija: model mora generirati dobro kalibrirane vjerojatnosti koje se mogu izravno koristiti za definiranje pragova odlučivanja, izračun očekivanog gubitka (Expected Loss) i cjenovno upravljanje rizikom.

U industrijskoj praksi linearni modeli, poput logističke regresije, i dalje se koriste zbog svoje jednostavnosti i interpretabilnosti, no često ne uspijevaju uhvatiti složene nelinearne odnose prisutne u stvarnim financijskim podacima. Zbog toga ansambl metode, osobito gradijentno pojačavanje stabala odluke, predstavljaju dominantan pristup u modernim sustavima procjene kreditnog rizika. XGBoost se u tom kontekstu ističe kao de facto standard zahvaljujući visokoj prediktivnoj učinkovitosti, skalabilnosti i fleksibilnosti u radu s neravnotežnim skupovima podataka.

Međutim, povećanje modelne složenosti donosi i značajan izazov: smanjenu transparentnost odluka. Regulatorni okviri i poslovna praksa zahtijevaju da odluke o kreditiranju budu objašnjive i auditabilne, kako bi se osiguralo povjerenje korisnika i usklađenost s propisima. U tom smislu, metode objašnjivosti poput SHAP-a omogućuju formalizirano tumačenje doprinosa pojedinih značajki na razini cijelog modela i pojedinačnih predikcija. Time se premošćuje jaz između visoke prediktivne snage suvremenih modela i potrebe za razumijevanjem i nadzorom njihovih odluka.

Motivacija ovog projekta proizlazi upravo iz potrebe za integracijom snažnih modela strojnog učenja s metodama objašnjivosti i rigorozne validacije, kako bi se izgradio sustav koji je istovremeno učinkovit, pouzdan i primjenjiv u realnom financijskom okruženju.

1.2 Ciljevi projekta

Projekt je definiran kao inženjerski i istraživački pipeline s ciljevima:

- prikupljanje najmanje dva heterogena skupa podataka i integracija u zajednički okvir,

- implementacija predobrade (imputacija, tretman outliera, kodiranje i skaliranje),
- usporedba baseline modela s Bayesovski optimiziranim XGBoost modelom,
- rigorozna validacija (nested CV) i optimizacija praga odluke,
- kalibracija vjerojatnosti i procjena kvalitete probabilističkih predikcija,
- interpretabilnost (globalna i lokalna) pomoću SHAP-a,
- segmentacija uzoraka (K-means) i analiza klastera,
- pohrana integriranog skupa, metrika i artefakata u bazu te izrada API sučelja.

1.3 Korišteni skupovi podataka i heterogenost

Korištena su najmanje dva izvora podataka:

- **Skup A (CSV)**: *Company Bankruptcy Prediction* [1] s ciljem `Bankrupt?`.
- **Skup B (XLSX)**: *Credit Risk Modelling Dataset* [2] s binarnom ciljanom varijablom (npr. `default`).

Skupovi su heterogeni po formatu (CSV i Excel), strukturi i semantici značajki.

1.4 Organizacija dokumenta

U nastavku se daje teorijska podloga, opis implementacije, prikaz rada aplikacije, kritički osvrt i zaključak, uz literaturu u IEEE stilu. Svaki relevantan isječak koda opisan je u tekstu prije ili poslije listinga.

Poglavlje 2

Glavni dio teme: formalizam, definicije i teorija

2.1 Binarna klasifikacija i probabilističko predviđanje

Neka su $\mathbf{x} \in \mathbb{R}^d$ značajke, a $y \in \{0, 1\}$ ciljana oznaka gdje $y = 1$ označava default/bankrot. Model procjenjuje vjerojatnost

$$p(y = 1 \mid \mathbf{x}) \in [0, 1]. \tag{2.1}$$

U kreditnom riziku takva vjerojatnost se često mapira u odluku preko praga τ:

$$\hat{y} = \mathbb{I}[p(y = 1 \mid \mathbf{x}) \geq \tau]. \tag{2.2}$$

2.2 Metrike evaluacije

Za neravnotežne probleme AUC i PR-AUC pružaju dopunske informacije. ROC-AUC mjeri rangiranje bez obzira na prag, dok PR-AUC bolje reflektira kvalitetu predikcije pozitivne klase. Za izabrani prag koristi se F1 mjera:

$$F1 = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}}. \tag{2.3}$$

Za probabilističku kvalitetu koristi se Brier score [6]:

$$\text{Brier} = \frac{1}{n} \sum_{i=1}^{n} (p_i - y_i)^2. \tag{2.4}$$

2.3 Logistička regresija

Logistička regresija modelira:

$$p(y = 1 \mid \mathbf{x}) = \sigma(\mathbf{w}^\top \mathbf{x} + b), \tag{2.5}$$

gdje je σ logistička funkcija. Prednost je interpretabilnost i stabilnost, no ograničenje je slabija sposobnost modeliranja nelinearnih odnosa.

2.4 Random Forest

Random Forest (RF) trenira ansambl stabala na bootstrap uzorcima, uz slučajni odabir značajki po splitu. RF je robustan i često dobar baseline, no može imati slabiju kalibraciju i slabije performanse u odnosu na specijalizirane metode gradijentnog pojačavanja.

2.5 XGBoost i gradijentno pojačavanje

XGBoost [3] je učinkovita implementacija gradijentnog pojačavanja stabala odluke s regularizacijom. U neravnotežnim problemima tipično se koristi `scale_pos_weight` kako bi se penalizirale pogreške na manjinskoj klasi.

2.6 Bayesova optimizacija hiperparametara (Optuna)

Umjesto grid/random pretrage koristi se Bayesova optimizacija hiperparametara s Optuna okvirom [4]. Optuna iterativno predlaže hiperparametre (trialove) prema prethodnim rezultatima i maksimalizira odabranu metriku (u ovom projektu PR-AUC).

2.7 Nested cross-validation

Nested CV razdvaja optimizaciju hiperparametara (inner loop) od procjene generalizacije (outer loop). Time se smanjuje optimistična pristranost pri procjeni performansi modela, posebno kada je prostor hiperparametara velik.

2.8 Imputacija: MICE

Za numeričke značajke koristi se IterativeImputer koji implementira MICE pristup [7]. Umjesto imputacije prosjekom/medijanom, MICE iterativno modelira svaku varijablu na temelju ostalih, često dajući realističnije imputacije.

2.9 Detekcija anomalija: Isolation Forest

Isolation Forest [8] izolira anomalije kroz slučajne particije prostora podataka. U projektu se koristi za izgradnju dodatne binarne značajke `outlier_flag` koja signalizira potencijalno atipične uzorke.

2.10 Kalibracija vjerojatnosti

Ansambl modeli mogu davati nekalibrirane vjerojatnosti. Izotonička kalibracija uči monotonu transformaciju koja usklađuje predikcije s empirijskim učestalostima [6]. Usporedba Brier score-a prije i poslije kalibracije daje jasan signal korisnosti.

2.11 Interpretabilnost: SHAP

SHAP [5] daje aditivna objašnjenja predikcija i povezuje doprinos značajki s formalizmom Shapleyjevih vrijednosti. Time se dobiva i globalna važnost značajki (summary plot) i lokalno objašnjenje za pojedini uzorak (waterfall).

2.12 Segmentacija: K-means

K-means [9] particionira uzorke u k klastera minimiziranjem unutar-klasterske varijance. Silhouette score se koristi za izbor k, a PCA projekcija omogućuje vizualnu provjeru separacije.

Poglavlje 3

Opis implementacije (komponente sustava)

3.1 Arhitektura rješenja

Sustav se sastoji od sljedećih komponenti:

1. Akvizicija podataka (Kaggle CLI) i validacija integriteta.
2. Predobrada i integracija heterogenih skupova u zajednički prostor značajki.
3. Modeliranje: baseline modeli i optimizirani XGBoost.
4. Validacija: standardna CV i nested CV.
5. Post-proces: threshold tuning, kalibracija, SHAP objašnjenja, K-means segmentacija.
6. Pohrana: SQLite baza za integrirane podatke i metrike.
7. Sučelje: REST API (FastAPI) za dohvat metrika i predikcije.

Slika 3.1: Arhitektura sustava (predložak; korisnik umeće vlastitu sliku).

3.2 Struktura repozitorija

Sljedeći isječak prikazuje preporučenu strukturu projekta. Odvajanje podataka, artefakata i izvornog koda olakšava održavanje i reproducibilnost.

```
default-risk-project/
  data/
    raw/
    interim/
    processed/
  artifacts/
    fig/
    models/
  src/
    api.py
```

```
    features.py
    train.py
  default_risk.db
  dvc.yaml
  .gitignore
  README.md
```

Listing 3.1: Primjer strukture projekta

3.3 Reproducibilnost: Git i DVC

Ovaj isječak demonstrira inicijalizaciju Git i DVC-a [10]. DVC prati velike datoteke i omogućuje vraćanje točne verzije skupa podataka korištene u eksperimentu.

```
git init
dvc init
dvc add data/raw/
git add .
git commit -m "Init: repo + DVC + raw data tracking"
```

Listing 3.2: Inicijalizacija Git i DVC

3.4 Akvizicija podataka (Kaggle)

U nastavku je kod koji provjerava dostupnost Kaggle tokena i preuzima dva skupa. Ovaj korak zadovoljava zahtjev projekta za dokumentiranje načina prikupljanja i citiranje izvora [1, 2].

```
from pathlib import Path
import os
import subprocess

def run(cmd):
    r = subprocess.run(cmd, shell=True, capture_output=True, text
       =True)
    return r.returncode, r.stdout.strip(), r.stderr.strip()

def kaggle_ready():
    p = Path.home() / ".kaggle" / "kaggle.json"
    if p.exists():
        return True
    if os.environ.get("KAGGLE_USERNAME") and os.environ.get("
       KAGGLE_KEY"):
        return True
    return False

def kaggle_download(dataset_slug, out_dir):
    out_dir = Path(out_dir)
    out_dir.mkdir(parents=True, exist_ok=True)
    cmd = f'kaggle datasets download -d "{dataset_slug}" -p "{
       out_dir.as_posix()}" --unzip -q'
```

```
    return run(cmd)

slug_a = "fedesoriano/company-bankruptcy-prediction"
slug_b = "abhirajmandal/credit-risk-modelling-dataset"

if kaggle_ready():
    kaggle_download(slug_a, "data/raw/bankruptcy_csv")
    kaggle_download(slug_b, "data/raw/credit_xlsx")
```

Listing 3.3: Preuzimanje heterogenih skupova (CSV i XLSX) s Kaggle-a

3.5 Učitavanje i validacija integriteta

Nakon preuzimanja, slijedi učitavanje i osnovna validacija (dimenzije, duplikati, udio nedostajućih). Ovaj korak služi kao kontrolna točka prije ulaska u predobradu.

```
import pandas as pd
import numpy as np

df_a_raw = pd.read_csv("data/raw/bankruptcy_csv/CompanyBankruptcy.csv")
df_b_raw = pd.read_excel("data/raw/credit_xlsx/case_study1.xlsx")

def basic_validation(df):
    return {
        "redaka": int(df.shape[0]),
        "stupaca": int(df.shape[1]),
        "duplikata": int(df.duplicated().sum()),
        "udio_nedostajucih": float(df.isna().mean().mean())
    }

va = basic_validation(df_a_raw)
vb = basic_validation(df_b_raw)
va, vb
```

Listing 3.4: Učitavanje i osnovna provjera integriteta

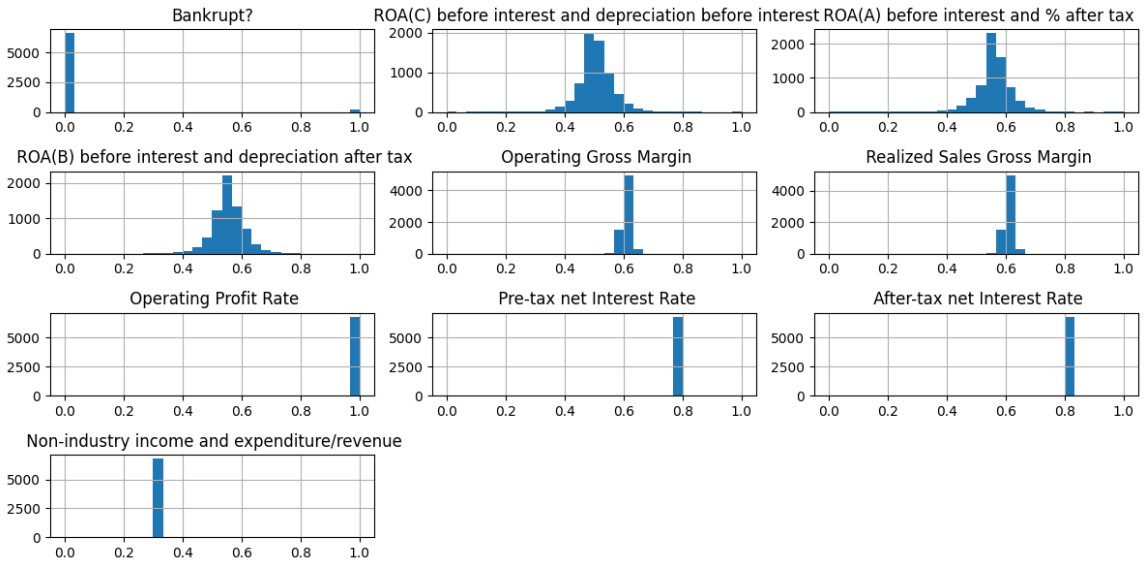
3.6 EDA i statistička analiza

Ovaj dio pokazuje primjer EDA postupaka: histogrami numeričkih varijabli i prikaz korelacija. Takvi grafovi pomažu identificirati ekstremne distribucije, potencijalne transformacije i multikolinearnost.

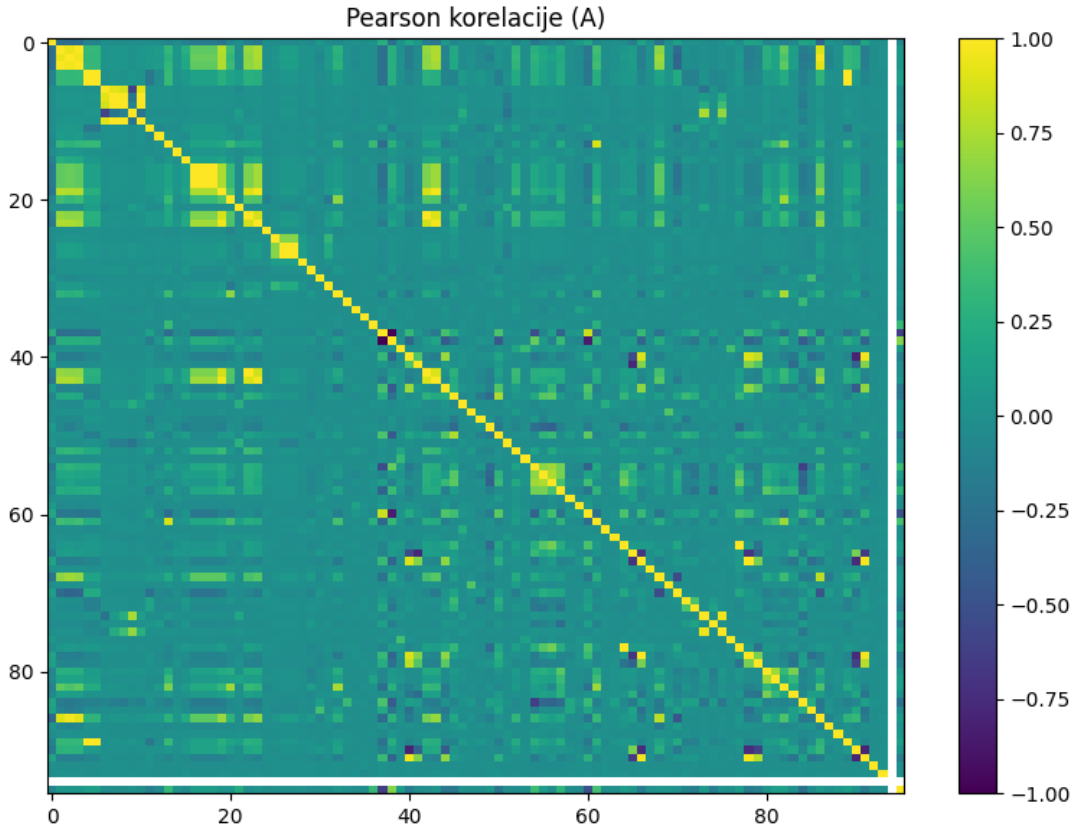
```
import matplotlib.pyplot as plt
import numpy as np

num_cols_a = df_a_raw.select_dtypes(include=[np.number]).columns.tolist()
df_a_raw[num_cols_a].iloc[:, :12].hist(bins=30, figsize=(12,6))
plt.tight_layout()
```

```
plt.show()

corr = df_a_raw[num_cols_a].corr(method="pearson")
plt.figure(figsize=(9,6))
plt.imshow(corr.values, aspect="auto")
plt.colorbar()
plt.title("Pearson korelacije (Skup A)")
plt.tight_layout()
plt.show()
```

Listing 3.5: Distribucije i korelacije (primjer)



Slika 3.2: Histogrami numeričkih varijabli (Skup A).



Slika 3.3: Matrica korelacija (Skup A).

3.7 Tretman outliera i anomalija

Sljedeći isječak implementira IQR winsorization (klipanje ekstremnih vrijednosti) i izgradnju `outlier_flag` značajke Isolation Forest metodom [8]. Ovakav signal može pomoći modelu u prepoznavanju atipičnih obrazaca.

```
import numpy as np
from sklearn.ensemble import IsolationForest

def iqr_clip_fit(X, num_cols, k=1.5):
    q1 = X[num_cols].quantile(0.25)
    q3 = X[num_cols].quantile(0.75)
    iqr = q3 - q1
    lo = q1 - k * iqr
    hi = q3 + k * iqr
    return lo, hi

def iqr_clip_transform(X, num_cols, lo, hi):
    X2 = X.copy()
    X2[num_cols] = X2[num_cols].clip(lo, hi, axis=1)
    return X2

def add_isoforest_flag(X, num_cols):
    Xn = X[num_cols].replace([np.inf, -np.inf], np.nan)
    Xn = Xn.fillna(Xn.median())
```

```
    iso = IsolationForest(n_estimators=200, contamination="auto",
        random_state=42)
    return (iso.fit_predict(Xn.values) == -1).astype(int)
```

Listing 3.6: IQR winsorization + Isolation Forest outlier flag

3.8 Predobrada: MICE imputacija, skaliranje i kodiranje

Ovaj preprocesor provodi MICE imputaciju [7] na numeričkim varijablama, standardizaciju, a za kategorijske varijable (ako postoje) imputaciju najčešćom vrijednošću i one-hot kodiranje. Implementacija je robusna na slučaj kada u skupu nema kategorijskih varijabli.

```
import numpy as np
from sklearn.pipeline import Pipeline
from sklearn.compose import ColumnTransformer
from sklearn.preprocessing import OneHotEncoder, StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer

def make_preprocessor(num_cols, cat_cols, seed=42):
    num_pipe = Pipeline([
        ("imp", IterativeImputer(random_state=seed, max_iter=20,
            sample_posterior=False)),
        ("sc", StandardScaler())
    ])
    if len(cat_cols) == 0:
        return ColumnTransformer([("num", num_pipe, num_cols)])
    cat_pipe = Pipeline([
        ("imp", SimpleImputer(strategy="most_frequent")),
        ("oh", OneHotEncoder(handle_unknown="ignore"))
    ])
    return ColumnTransformer([
        ("num", num_pipe, num_cols),
        ("cat", cat_pipe, cat_cols)
    ])

def to_feature_names(prep):
    num_cols = list(prep.transformers_[0][2])
    cat_cols = list(prep.transformers_[1][2]) if len(prep.
        transformers_) > 1 else []
    if len(cat_cols) == 0:
        return num_cols
    cat_pipe = prep.transformers_[1][1]
    oh = cat_pipe.named_steps["oh"]
    return num_cols + oh.get_feature_names_out(cat_cols).tolist()
```

Listing 3.7: Robustan ColumnTransformer preprocesor i dohvat naziva značajki

3.9 Integracija heterogenih skupova

Integracija se izvodi tako da se oba skupa zasebno preprocesiraju, a zatim se njihove matrice značajki spajaju u zajednički prostor značajki pomoću *outer* konkatenacije uz popunjavanje nedostajućih značajki nulama. Dodatno se čuva informacija o izvoru uzorka kroz `dataset` atribut.

```
import pandas as pd
import numpy as np

def split_types(df):
    num = df.select_dtypes(include=[np.number]).columns.tolist()
    cat = [c for c in df.columns if c not in num]
    return num, cat

Xa = df_a_raw.drop(columns=["Bankrupt?"])
ya = df_a_raw["Bankrupt?"].astype(int)

Xb = df_b_raw.drop(columns=["default"])
yb = df_b_raw["default"].astype(int)

num_a, cat_a = split_types(Xa)
num_b, cat_b = split_types(Xb)

prep_a = make_preprocessor(num_a, cat_a, seed=42)
prep_b = make_preprocessor(num_b, cat_b, seed=42)

Za = prep_a.fit_transform(Xa)
Zb = prep_b.fit_transform(Xb)

fa = to_feature_names(prep_a)
fb = to_feature_names(prep_b)

Xa_enc = pd.DataFrame(Za.toarray() if hasattr(Za,"toarray") else
   Za, columns=fa)
Xb_enc = pd.DataFrame(Zb.toarray() if hasattr(Zb,"toarray") else
   Zb, columns=fb)

Xa_enc["dataset"] = "A"
Xb_enc["dataset"] = "B"

X_all = pd.concat([Xa_enc, Xb_enc], axis=0, join="outer",
   ignore_index=True).fillna(0.0)
y_all = pd.concat([ya.reset_index(drop=True), yb.reset_index(drop
   =True)], ignore_index=True)
X_all.shape, y_all.mean()
```

Listing 3.8: Integracija A i B skupa u zajednički prostor značajki

3.10 Selekcija značajki i baseline modeli

Zbog potencijalno visoke dimenzionalnosti uvodi se selekcija značajki. RFE postupno uklanja najmanje važne značajke prema logističkoj regresiji te ostavlja podskup za treniranje modela. Nakon toga se treniraju baseline modeli (LR i RF) radi usporedbe.

```
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import roc_auc_score,
    average_precision_score

X = X_all.drop(columns=["dataset"]).astype(float).values
y = y_all.values

lr_base = LogisticRegression(max_iter=1000, solver="liblinear",
    class_weight="balanced", random_state=42)
rfe = RFE(estimator=lr_base, n_features_to_select=min(60, X.shape
    [1]//2), step=0.1)
rfe.fit(X, y)

mask = rfe.support_
X_sel = X[:, mask]

lr = LogisticRegression(max_iter=1000, solver="liblinear",
    class_weight="balanced", random_state=42)
rf = RandomForestClassifier(n_estimators=600, class_weight="
    balanced_subsample", random_state=42)

lr.fit(X_sel, y)
p_lr = lr.predict_proba(X_sel)[:,1]

rf.fit(X_sel, y)
p_rf = rf.predict_proba(X_sel)[:,1]

roc_auc_score(y, p_lr), average_precision_score(y, p_lr),
    roc_auc_score(y, p_rf)
```

Listing 3.9: RFE selekcija + baseline modeli (LR i RF)

3.11 Optuna + XGBoost optimizacija

Sljedeći isječak definira `objective` funkciju za Optuna: za svaki trial predlaže hiperparametre XGBoost modela, evaluira ih stratificiranom CV-om i vraća prosječni PR-AUC.

```
import optuna
from xgboost import XGBClassifier
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import average_precision_score
import numpy as np

```

```python
def objective(trial, X, y, seed=42, n_splits=5):
    params = {
        "n_estimators": trial.suggest_int("n_estimators", 200, 1200),
        "max_depth": trial.suggest_int("max_depth", 2, 8),
        "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.25, log=True),
        "subsample": trial.suggest_float("subsample", 0.6, 1.0),
        "colsample_bytree": trial.suggest_float("colsample_bytree", 0.6, 1.0),
        "min_child_weight": trial.suggest_float("min_child_weight", 1e-2, 20.0, log=True),
        "reg_lambda": trial.suggest_float("reg_lambda", 1e-3, 50.0, log=True),
        "reg_alpha": trial.suggest_float("reg_alpha", 1e-4, 10.0, log=True),
        "gamma": trial.suggest_float("gamma", 0.0, 5.0),
        "objective": "binary:logistic",
        "eval_metric": "aucpr",
        "random_state": seed,
        "n_jobs": -1
    }
    posw = float((y==0).sum() / max(1, (y==1).sum()))
    params["scale_pos_weight"] = posw

    cv = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=seed)
    scores = []
    for tr_idx, va_idx in cv.split(X, y):
        m = XGBClassifier(**params)
        m.fit(X[tr_idx], y[tr_idx])
        p = m.predict_proba(X[va_idx])[:,1]
        scores.append(average_precision_score(y[va_idx], p))
    return float(np.mean(scores))

study = optuna.create_study(direction="maximize")
study.optimize(lambda t: objective(t, X_sel, y, seed=42, n_splits=5), n_trials=40, show_progress_bar=False)

study.best_value, study.best_params
```

Listing 3.10: Optuna objective funkcija za XGBoost (cilj: PR-AUC)

3.12 Rigorozna validacija: nested CV

Nested CV implementacija osigurava da se hiperparametri odabiru isključivo unutar train dijela vanjskog folda, a evaluacija se radi na neviđenim vanjskim test foldovima.

```python
from sklearn.model_selection import StratifiedKFold
from sklearn.metrics import roc_auc_score, average_precision_score
```

```
from xgboost import XGBClassifier
import numpy as np
import optuna

outer = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
outer_rows = []

for fold, (tr_idx, te_idx) in enumerate(outer.split(X_sel, y), start=1):
    Xo_tr, Xo_te = X_sel[tr_idx], X_sel[te_idx]
    yo_tr, yo_te = y[tr_idx], y[te_idx]

    st = optuna.create_study(direction="maximize")
    st.optimize(lambda t: objective(t, Xo_tr, yo_tr, seed=42, n_splits=3), n_trials=15, show_progress_bar=False)

    bp = dict(st.best_params)
    bp.update({
        "objective": "binary:logistic",
        "eval_metric": "aucpr",
        "random_state": 42,
        "n_jobs": -1,
        "scale_pos_weight": float((yo_tr==0).sum() / max(1,(yo_tr==1).sum()))
    })

    m = XGBClassifier(**bp)
    m.fit(Xo_tr, yo_tr)
    p = m.predict_proba(Xo_te)[:,1]

    outer_rows.append({
        "fold": fold,
        "AUC": float(roc_auc_score(yo_te, p)),
        "PR_AUC": float(average_precision_score(yo_te, p))
    })

outer_rows, float(np.mean([r["AUC"] for r in outer_rows])), float(np.mean([r["PR_AUC"] for r in outer_rows]))
```

Listing 3.11: Nested CV (outer + inner Optuna)

3.13 Odabir praga odluke

Ovaj dio pokazuje kako se prag τ može optimizirati za F1 ili prema jednostavnoj profit funkciji (cost-sensitive pristup). U stvarnoj primjeni težine se određuju prema poslovnim troškovima FP i FN.

```
import numpy as np
from sklearn.metrics import precision_recall_curve, f1_score

```

```
p = m.predict_proba(X_sel)[:,1]
pr, rc, th = precision_recall_curve(y, p)

f1s = 2*pr*rc/(pr+rc+1e-12)
best_i = int(np.nanargmax(f1s[:-1])) if len(th) else 0
thr_f1 = float(th[best_i]) if len(th) else 0.5
yhat = (p >= thr_f1).astype(int)

thr_f1, f1_score(y, yhat)

C_TP, C_FP, C_FN, C_TN = 1.0, -0.2, -1.0, 0.0
def profit(y_true, y_pred):
    y_true = np.asarray(y_true); y_pred = np.asarray(y_pred)
    tp = ((y_true==1)&(y_pred==1)).sum()
    fp = ((y_true==0)&(y_pred==1)).sum()
    fn = ((y_true==1)&(y_pred==0)).sum()
    tn = ((y_true==0)&(y_pred==0)).sum()
    return C_TP*tp + C_FP*fp + C_FN*fn + C_TN*tn

profits = [(t, profit(y, (p>=t).astype(int))) for t in np.
   linspace(0.01,0.99,99)]
best_t_profit = max(profits, key=lambda x: x[1])[0]
best_t_profit
```

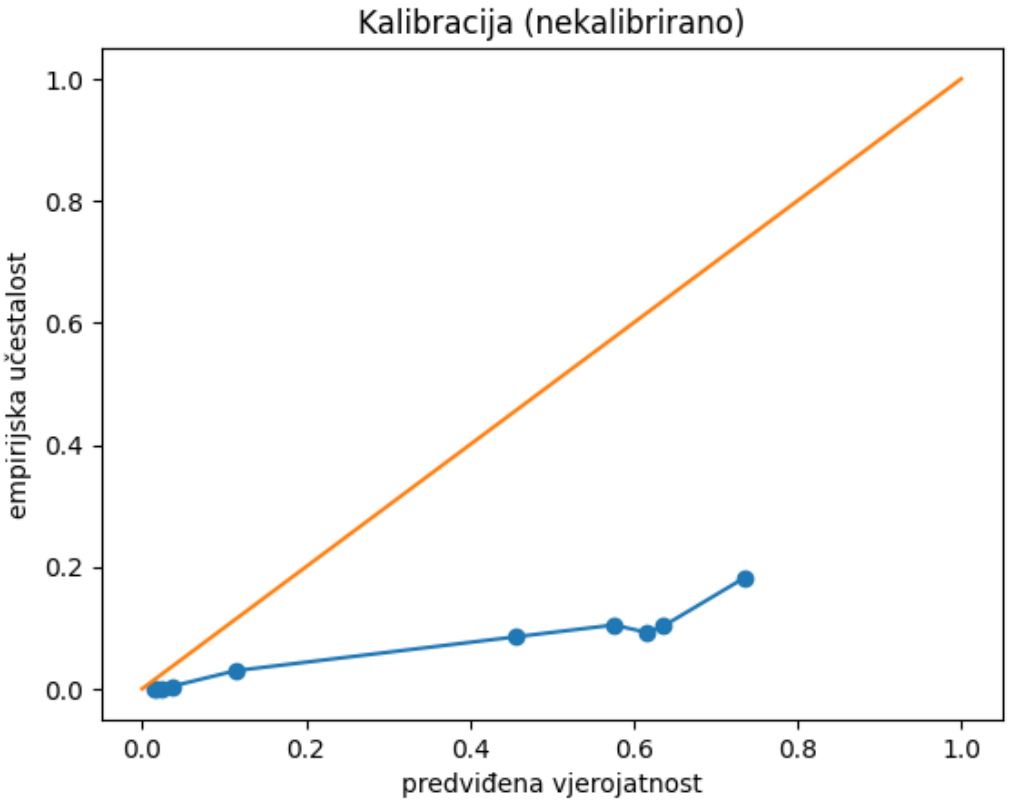
Listing 3.12: Threshold tuning: F1 i profit funkcija

3.14 Kalibracija vjerojatnosti

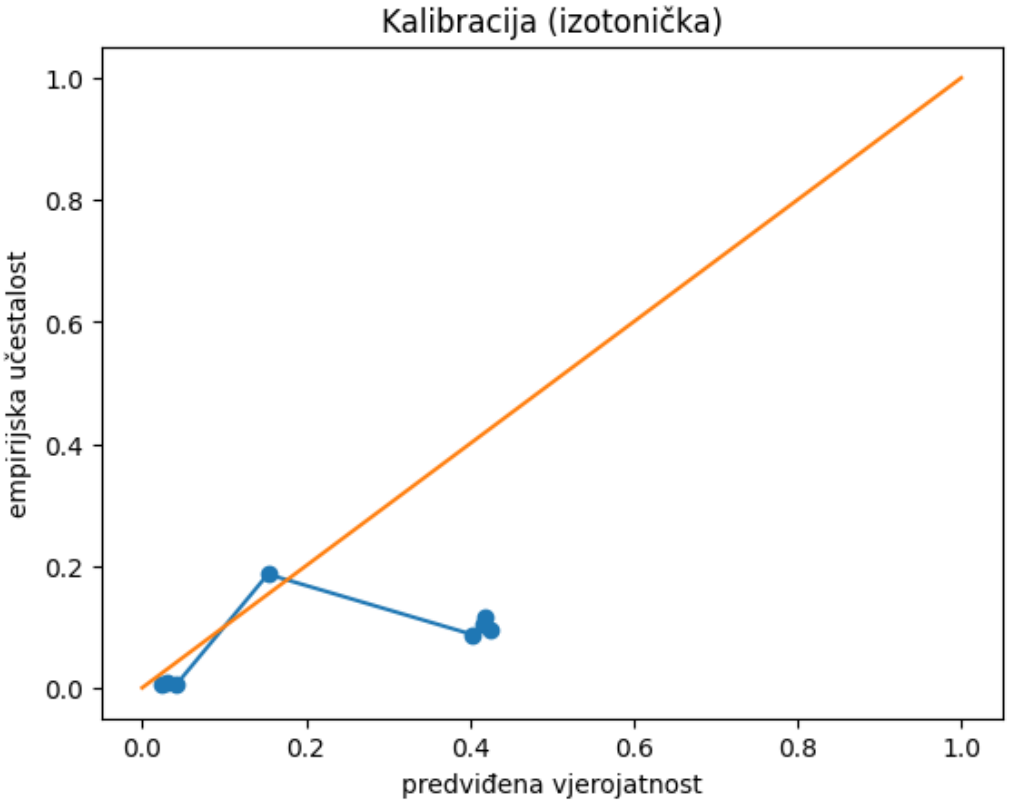
Sljedeći isječak provodi izotoničku kalibraciju i uspoređuje Brier score prije i poslije. U dokumentaciji se preporučuje uključiti i kalibracijske krivulje.

```
from sklearn.calibration import CalibratedClassifierCV
from sklearn.metrics import brier_score_loss

cal = CalibratedClassifierCV(m, method="isotonic", cv=3)
cal.fit(X_sel, y)

p_uncal = m.predict_proba(X_sel)[:,1]
p_cal = cal.predict_proba(X_sel)[:,1]

brier_score_loss(y, p_uncal), brier_score_loss(y, p_cal)
```

Listing 3.13: Izotonička kalibracija i Brier score



Slika 3.4: Kalibracijska krivulja prije kalibracije (predložak).



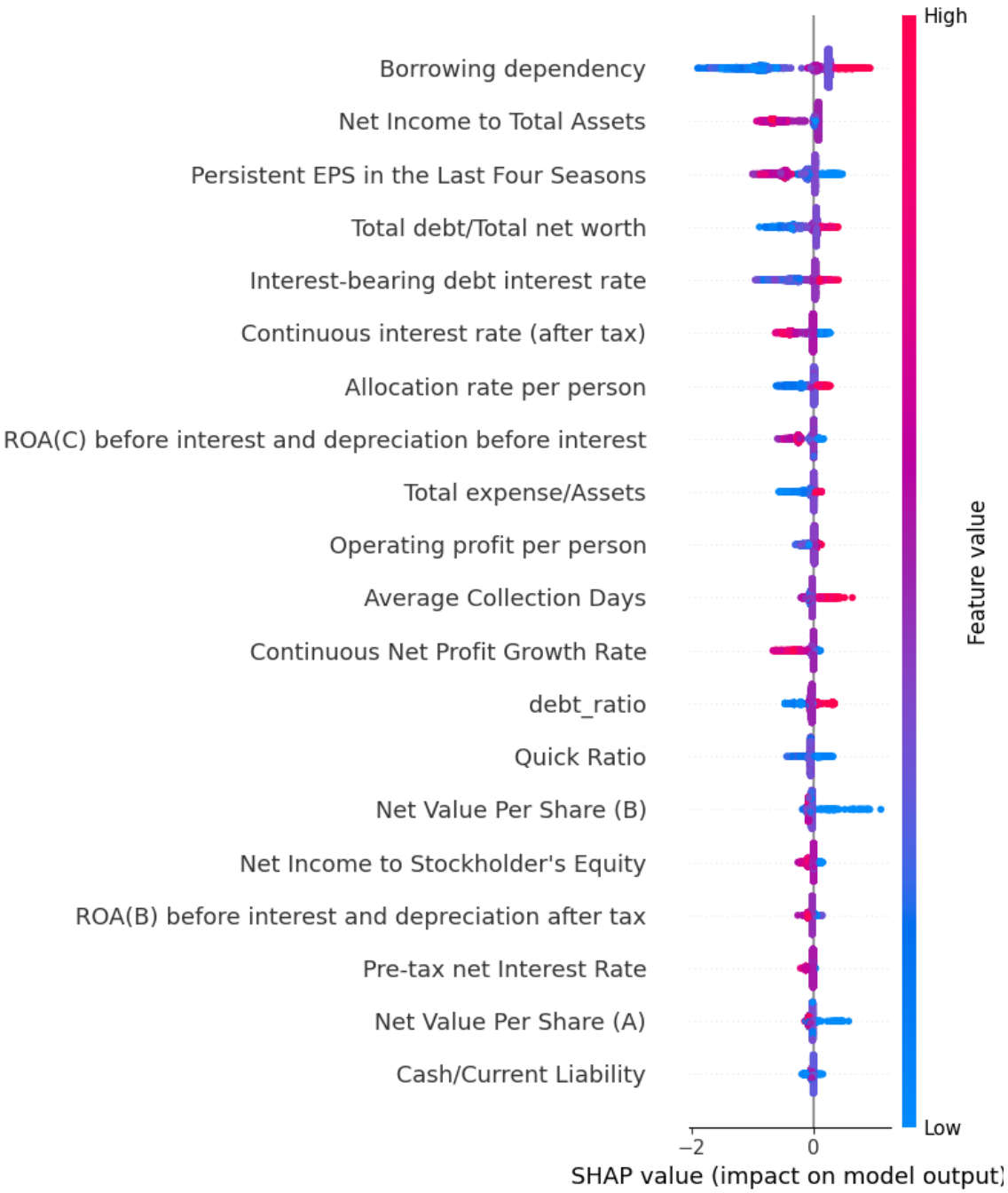
Slika 3.5: Kalibracijska krivulja nakon izotoničke kalibracije (predložak).

3.15 SHAP interpretabilnost

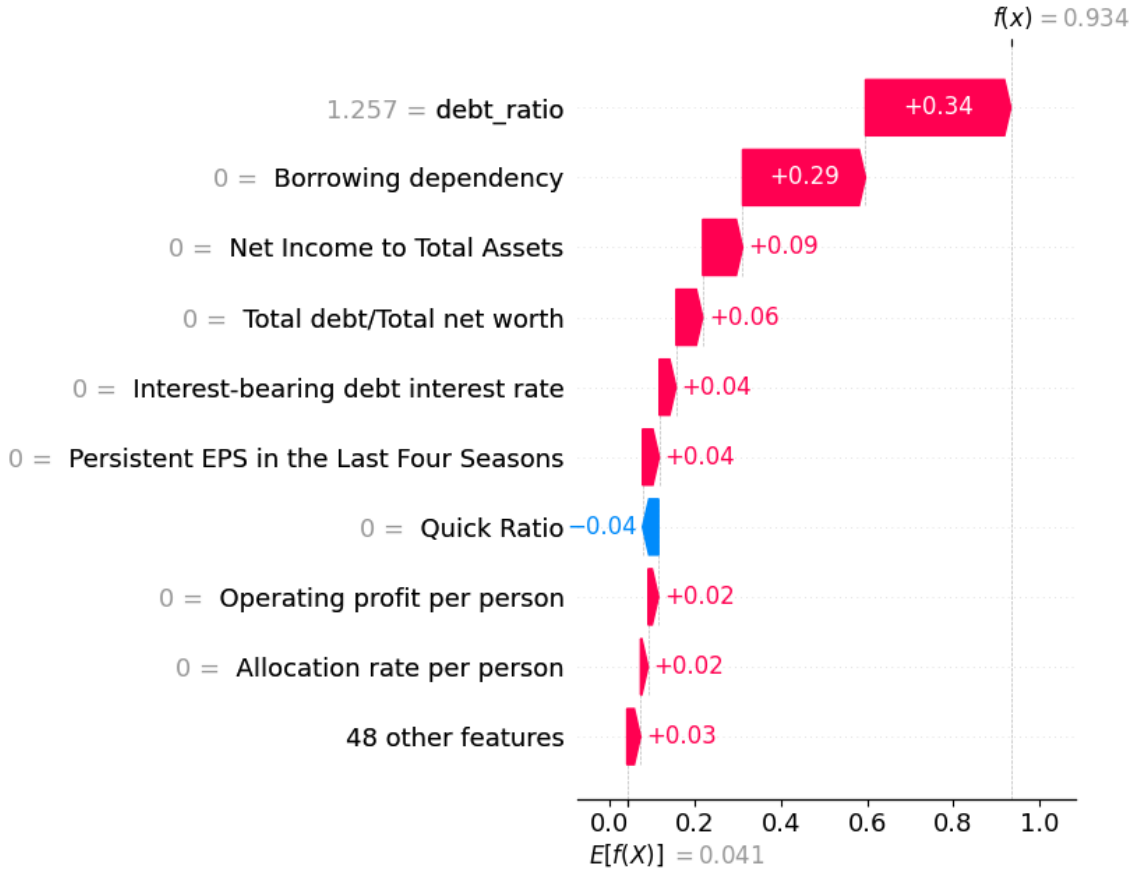
Ovaj dio računa SHAP vrijednosti i sprema grafove. Preporuka je koristiti uzorak podataka radi vremenske učinkovitosti, te navesti barem jedan globalni i jedan lokalni graf.

```
import shap
import matplotlib.pyplot as plt

expl = shap.TreeExplainer(m)
X_sample = X_sel[:2000]
sv = expl.shap_values(X_sample)

shap.summary_plot(sv, X_sample, show=False)
plt.savefig("artifacts/fig/shap_summary.png", dpi=200,
    bbox_inches="tight")
plt.close()

idx = 0
shap.plots._waterfall.waterfall_legacy(expl.expected_value, sv[
    idx], features=X_sample[idx], show=False)
plt.savefig("artifacts/fig/shap_waterfall.png", dpi=200,
    bbox_inches="tight")
plt.close()
```

Listing 3.14: SHAP TreeExplainer: globalno i lokalno objašnjenje



Slika 3.6: SHAP summary plot: globalna važnost značajki (predložak).



Slika 3.7: SHAP waterfall: lokalno objašnjenje za jedan uzorak (predložak).

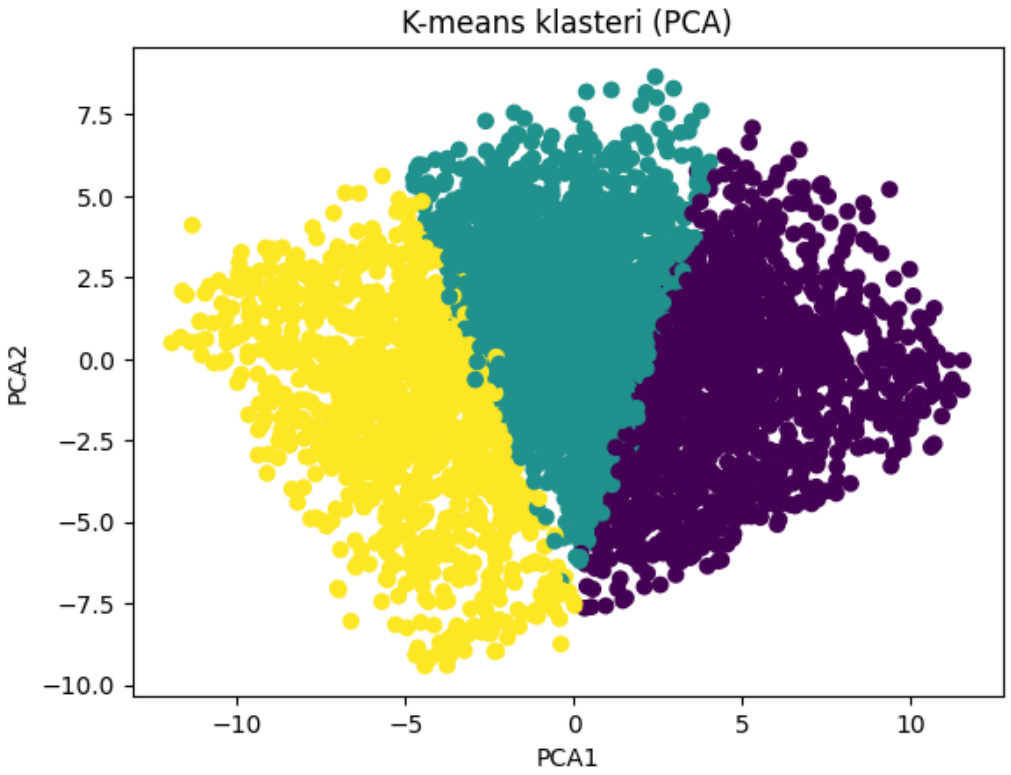
3.16 K-means segmentacija

U nastavku se bira broj klastera prema silhouette score-u i vizualizira segmentacija u PCA prostoru. Nakon toga se tipično izvodi analiza klastera: stopa defaulta po klasteru i prosjeci ključnih značajki.

```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.pyplot as plt

k_vals = range(2, 9)
sil = []
for k in k_vals:
    km = KMeans(n_clusters=k, random_state=42, n_init=10)
    lab = km.fit_predict(X_sel)
    sil.append(silhouette_score(X_sel, lab))

k_best = int(list(k_vals)[int(np.argmax(sil))])

km = KMeans(n_clusters=k_best, random_state=42, n_init=10)
lab = km.fit_predict(X_sel)
```

```
18
19 pca = PCA(n_components=2, random_state=42)
20 Z = pca.fit_transform(X_sel)
21
22 plt.figure()
23 plt.scatter(Z[:,0], Z[:,1], c=lab)
24 plt.savefig("artifacts/fig/kmeans_pca.png", dpi=200, bbox_inches=
       "tight")
25 plt.close()
26
27 k_best, float(np.max(sil))
```

Listing 3.15: K-means: izbor k, učenje i PCA vizualizacija



Slika 3.8: K-means klasteri u PCA projekciji (predložak).

3.17 Pohrana integriranih podataka i metrika u SQLite

Pohrana u SQLite omogućuje ponovnu uporabu rezultata bez retreniranja te jednostavan dohvat kroz API sloj. Sljedeći isječak sprema odabrane značajke i ciljne oznake, te primjer tablice metrika.

```
1 import sqlite3
2 import pandas as pd
3
```

```
DB = "default_risk.db"
con = sqlite3.connect(DB)

pd.DataFrame(X_sel).assign(y=y).to_sql("dataset_selected", con,
    index=False, if_exists="replace")

metrics = pd.DataFrame([
    {"model":"LR", "AUC": None, "PR_AUC": None, "F1": None, "thr"
        : None},
    {"model":"RF", "AUC": None, "PR_AUC": None, "F1": None, "thr"
        : None},
    {"model":"XGB", "AUC": None, "PR_AUC": None, "F1": None, "thr
        ": None}
])
metrics.to_sql("metrics", con, index=False, if_exists="replace")

con.close()
```

Listing 3.16: SQLite: spremanje procesiranih podataka i metrika

3.17.1 Shema baze

Sljedeći SQL primjer opisuje minimalnu shemu tablica. U praksi se tablice generiraju iz pandas DataFrame objekata, ali eksplicitna shema je korisna za dokumentaciju.

```
CREATE TABLE dataset_selected (
  f0 REAL, f1 REAL, ...,
  y INTEGER
);

CREATE TABLE metrics (
  model TEXT,
  AUC REAL,
  PR_AUC REAL,
  F1 REAL,
  thr REAL
);
```

Listing 3.17: Primjer sheme (konceptualno)

3.18 REST API sučelje (FastAPI)

API sloj omogućuje standardiziran pristup metrikama i predikcijama. U nastavku je minimalna implementacija: `/health` za provjeru stanja, `/metrics` za dohvat metrika i `/predict` za predikciju vjerojatnosti defaulta.

```
from fastapi import FastAPI
from pydantic import BaseModel
import numpy as np
import sqlite3
import pandas as pd
```

```

app = FastAPI()

class PredictIn(BaseModel):
    features: dict

def read_sql(table, db_path="default_risk.db"):
    con = sqlite3.connect(db_path)
    try:
        return pd.read_sql_query(f"SELECT * FROM {table}", con)
    finally:
        con.close()

@app.get("/health")
def health():
    return {"status": "ok"}

@app.get("/metrics")
def metrics():
    df = read_sql("metrics")
    return df.to_dict(orient="records")

@app.post("/predict")
def predict(inp: PredictIn):
    x = np.zeros(X_sel.shape[1], dtype=float)
    for k, v in inp.features.items():
        idx = int(k)
        x[idx] = float(v)
    p = float(cal.predict_proba(x.reshape(1,-1))[:,1][0])
    return {"p_default": p}
```

Listing 3.18: FastAPI: endpointi health, metrics i predict

3.19 Pokretanje servisa

Ovaj isječak prikazuje tipični način pokretanja FastAPI aplikacije pomoću Uvicorn poslužitelja.

```
uvicorn src.api:app --host 127.0.0.1 --port 8000 --reload
```

Listing 3.19: Pokretanje FastAPI aplikacije

Poglavlje 4

Prikaz rada aplikacije

4.1 Pipeline: od podataka do modela

Prikaz rada aplikacije organiziran je kroz korake:

1. preuzimanje podataka i validacija,
2. EDA i statistička analiza,
3. predobrada i integracija,
4. treniranje baseline modela,
5. Optuna optimizacija XGBoost-a,
6. nested CV evaluacija,
7. threshold tuning i kalibracija,
8. SHAP i K-means analize,
9. pohrana u SQLite,
10. izlaganje kroz API i testiranje endpointa.

4.2 Testiranje API-ja

Sljedeći primjeri pokazuju provjeru stanja servisa i dohvat metrika.

```
curl http://127.0.0.1:8000/health
curl http://127.0.0.1:8000/metrics
```

Listing 4.1: Testiranje /health i /metrics endpointa

4.3 Primjer predikcije

U praksi se šalje JSON koji sadrži vrijednosti značajki. Ovaj primjer pretpostavlja da se u `features` šalju indeksi značajki (jednostavniji oblik), dok je produkcijski pristup tipično mapiranje po nazivima.

```
curl -X POST "http://127.0.0.1:8000/predict" \
  -H "Content-Type: application/json" \
  -d "{\"features\": {\"0\": 0.12, \"1\": -1.03, \"2\": 0.44}}"
```

Listing 4.2: Primjer poziva /predict endpointa

Slika 4.1: Primjer prikaza rada API-ja (predložak; korisnik umeće sliku).

Poglavlje 5

Rezultati i vizualizacije

5.1 Tablice metrika

U tablicu se unose rezultati iz izvršenog notebooka. Ako se metrike razlikuju od navedenih, potrebno je ažurirati vrijednosti u tablici.

Tablica 5.1: Usporedba modela (primjer popunjavanja).

Model	AUC	PR-AUC	F1	Prag
Logistička regresija	0.7909	0.2164	0.2566	0.7294
Random Forest	0.6256	0.1189	0.2564	0.1417
XGBoost (Optuna)	0.7921	0.2464	0.2673	0.7202

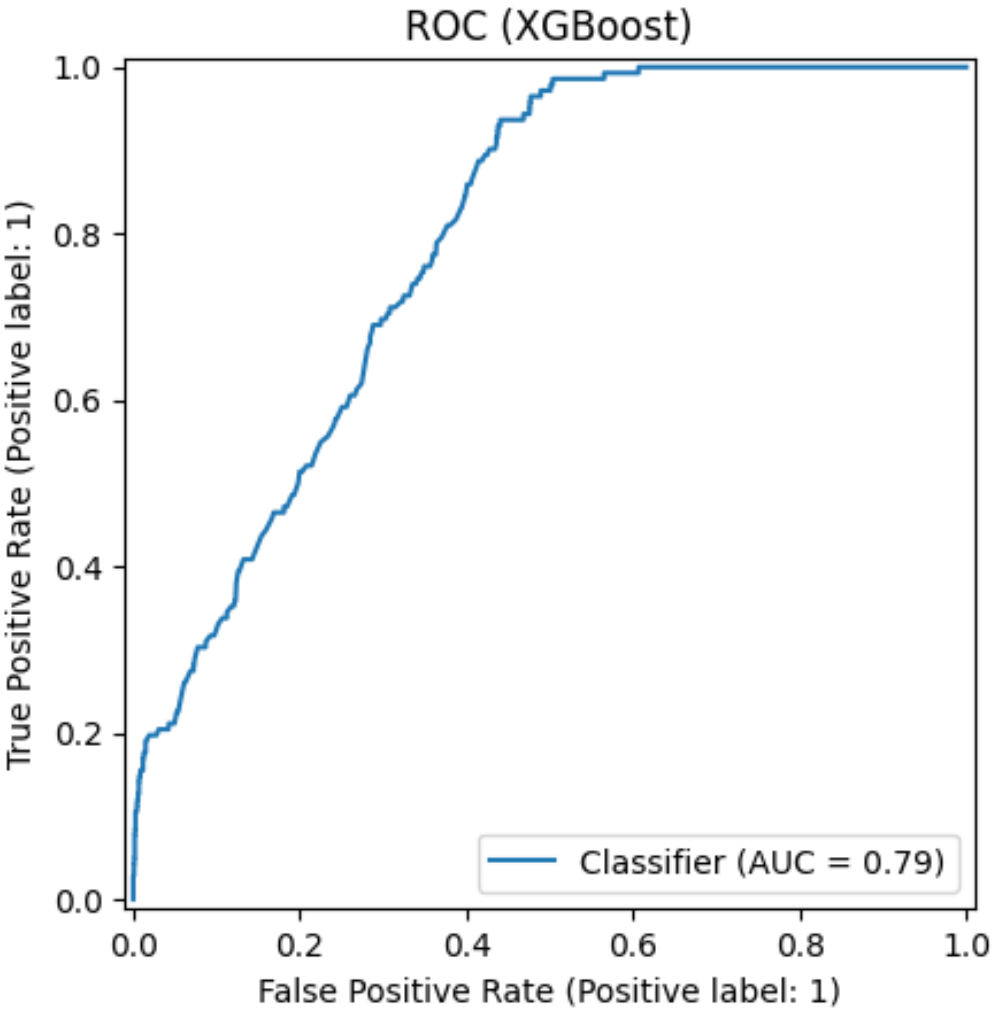
5.2 Nested CV rezultati

Tablica 5.2: Nested CV rezultati (primjer).

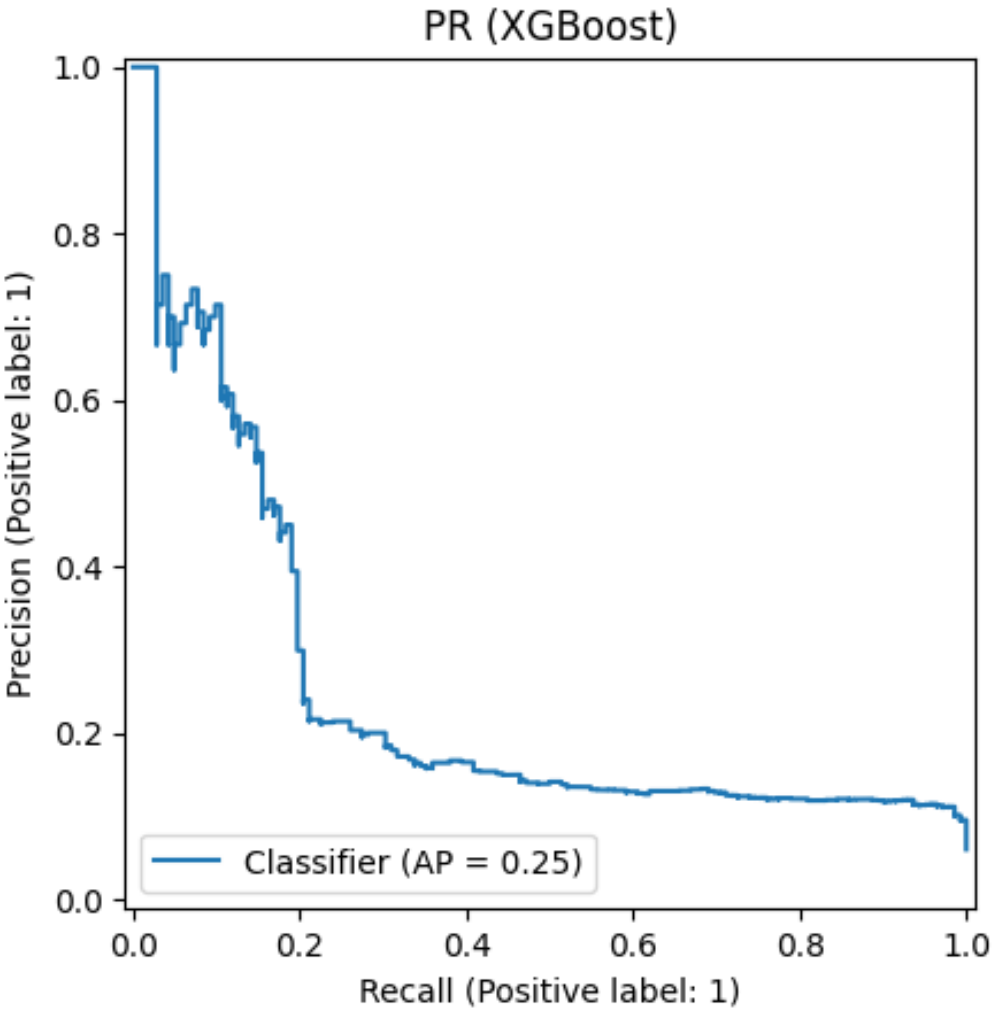
Fold	AUC	PR-AUC
1	0.8171	0.2520
2	0.8097	0.2014
3	0.7742	0.1702
4	0.8159	0.2890
5	0.8059	0.2671
Prosjek	0.8046	0.2359

5.3 Ključni grafovi

U nastavku su predlošci za umetanje grafova iz notebooka. Korisnik treba spremiti slike u mapu `fig/`.



Slika 5.1: ROC krivulja za XGBoost model (predložak).



Slika 5.2: Precision–Recall krivulja za XGBoost model (predložak).

Poglavlje 6

Kritički osvrt

6.1 Praktična izvedivost

Razvijeni sustav pokazuje visoku razinu praktične izvedivosti zahvaljujući modularnoj arhitekturi i oslanjanju na široko prihvaćene, stabilne Python biblioteke kao što su scikit-learn, xgboost, optuna, shap i fastapi. Takav tehnološki odabir omogućuje jednostavno održavanje, nadogradnju pojedinih komponenti i razumijevanje sustava od strane drugih inženjera ili analitičara.

Posebna prednost sustava jest korištenje SQLite baze podataka za pohranu integriranih podataka, metrika i rezultata modeliranja. Time se značajno smanjuje infrastrukturna složenost u usporedbi s distribucijskim bazama podataka, što je osobito pogodno za prototipe, istraživačke projekte i manje produkcijske sustave. Unatoč svojoj jednostavnosti, SQLite omogućuje pouzdanu i konzistentnu pohranu rezultata te njihovu ponovnu uporabu bez potrebe za ponovnim treniranjem modela.

Izlaganje funkcionalnosti sustava putem REST API-ja dodatno povećava njegovu praktičnu vrijednost. API sloj omogućuje jednostavnu integraciju modela procjene default rizika u postojeće informacijske sustave, web aplikacije ili automatizirane procese odlučivanja. Takav pristup odvaja modeliranje od korisničkog sučelja i poslovne logike, što je u skladu s modernim principima dizajna softverskih sustava.

Konačno, naglasak na reproducibilnosti kroz jasno definirani pipeline, verzioniranje podataka i strogu validaciju (nested cross-validation) čini sustav pogodnim ne samo za demonstraciju koncepta, već i kao temelj za daljnji razvoj prema produkcijskom okruženju. Time se potvrđuje da predloženo rješenje nije isključivo akademska vježba, već realno primjenjiv okvir za procjenu kreditnog rizika.

6.2 Primjena i ograničenja

Prednosti:

- XGBoost uz Bayesovu optimizaciju često daje dobar kompromis između performansi i stabilnosti [3, 4].
- Kalibracija poboljšava uporabljivost vjerojatnosti za poslovna pravila [6].
- SHAP omogućuje auditabilnost i objašnjenja prema regulatornim i poslovnim potrebama [5].

- Segmentacija K-means može otkriti profile rizika i podržati strategije upravljanja portfeljem [9].

Ograničenja:

- Integracija heterogenih skupova kroz union prostora značajki može rezultirati sparsnom i visokodimenzionalnom matricom, što povećava potrebu za selekcijom i regularizacijom.
- Nested CV i Optuna povećavaju računalni trošak te zahtijevaju balans između rigoroznosti i vremena.
- K-means pretpostavlja sferne klastere i osjetljiv je na skaliranje; interpretacija klastera zahtijeva dodatnu analizu.
- SHAP može biti računalno zahtjevan na velikim uzorcima pa se preporučuje uzorkovanje.

6.3 Moguća poboljšanja

- Uvođenje monitoringa (data drift, concept drift) i periodičnog retreniranja.
- Uvođenje cost-sensitive optimizacije praga prema realnim troškovima i prihodima.
- Povećanje transparentnosti kroz modelne kartice i automatsku generaciju izvještaja.
- Razmatranje alternative segmentacije (Gaussian Mixture Models, HDBSCAN) i usporedba.

Poglavlje 7

Zaključak

U ovom radu razvijen je cjelovit i reproducibilan pipeline za procjenu financijskog rizika neispunjenja obveza koji obuhvaća sve ključne faze modernog sustava strojnog učenja: od heterogene akvizicije i integracije podataka, preko napredne predobrade i usporedbe baseline modela, do Bayesovski optimiziranog XGBoost modela uz rigoroznu nested cross-validation evaluaciju. Dodatne komponente, poput optimizacije praga odluke i kalibracije vjerojatnosti, osiguravaju da su izlazi modela praktično uporabivi u kontekstu stvarnih poslovnih odluka.

Poseban doprinos rada očituje se u integraciji metoda interpretabilnosti i analize podataka. Primjena SHAP metode omogućuje transparentno i formalno objašnjenje odluka složenog ansambl modela, čime se postiže auditabilnost nužna za regulatorne i poslovne zahtjeve. Segmentacija uzoraka K-means metodom dodatno proširuje analitičku vrijednost sustava, omogućujući identifikaciju različitih profila rizika unutar populacije.

Rezultati rada pokazuju da kombinacija snažnih modela strojnog učenja, rigorozne validacije i metoda objašnjivosti može rezultirati sustavom koji je istovremeno prediktivno učinkovit, transparentan i reproducibilan. Takav sustav predstavlja čvrst temelj za daljnji razvoj prema produkcijskim rješenjima u području kreditnog rizika te potvrđuje da suvremeni pristupi strojnog učenja mogu zadovoljiti i tehničke i regulatorne zahtjeve financijske industrije.

Bibliografija

[1] Kaggle, "Company Bankruptcy Prediction," dataset, pristupljeno: 23. siječnja 2026..

[2] Kaggle, "Credit Risk Modelling Dataset," dataset, pristupljeno: 23. siječnja 2026..

[3] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2016, pp. 785–794.

[4] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A Next-generation Hyperparameter Optimization Framework," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2019.

[5] S. M. Lundberg and S.-I. Lee, "A Unified Approach to Interpreting Model Predictions," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[6] A. Niculescu-Mizil and R. Caruana, "Predicting Good Probabilities with Supervised Learning," in *Proceedings of the 22nd International Conference on Machine Learning (ICML)*, 2005, pp. 625–632.

[7] S. van Buuren and K. Groothuis-Oudshoorn, "mice: Multivariate Imputation by Chained Equations in R," *Journal of Statistical Software*, vol. 45, no. 3, pp. 1–67, 2011.

[8] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation Forest," in *2008 Eighth IEEE International Conference on Data Mining*, 2008, pp. 413–422.

[9] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations," in *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, 1967, pp. 281–297.

[10] DVC, "Data Version Control (DVC) Documentation," službena dokumentacija, pristupljeno: 23. siječnja 2026..

Dodatak A

Dodatci

A.1 Popis slika koje je potrebno umetnuti

U nastavku je popis predloženih slika koje ovaj dokument referencira. Potrebno ih je spremiti u mapu `fig/`:

- `architecture.png` – arhitektura sustava.
- `eda_hist_a.png` – histogrami skupa A.
- `eda_corr_a.png` – korelacije skupa A.
- `roc_xgb.png` – ROC krivulja.
- `pr_xgb.png` – PR krivulja.
- `calibration_uncalibrated.png` – kalibracija prije.
- `calibration_isotonic.png` – kalibracija poslije.
- `shap_summary.png` – SHAP summary.
- `shap_waterfall.png` – SHAP waterfall.
- `silhouette.png` – silhouette graf.
- `kmeans_pca.png` – PCA vizualizacija klastera.
- `api_demo.png` – prikaz rada API-ja (screenshot).