Explainable AI for Financial Models: SHAP & LIME Integration

Your Name

July 2, 2025

1. Motivation: Interpretability in Finance

Financial decision models (e.g. credit approval, risk scoring) require *transparency* for:

- **Regulatory compliance:** Legal frameworks (e.g. GDPR, Basel III) mandate explainable decisions.
- **Stakeholder trust:** Loan officers and customers need human-readable rationales.
- **Bias detection:** Reveal and mitigate unfair treatment across demographics.

2. SHAP & LIME for Local Explanations

SHAP (SHapley Additive exPlanations) Derives from Shapley values in cooperative game theory. Computes each feature's average contribution to a prediction over all coalitions:

$$\phi_j = \sum_{S \subseteq F \setminus \{j\}} \frac{|S|!\,(|F| - |S| - 1)!}{|F|!} \big[f_{S \cup \{j\}}(\mathbf{x}) - f_S(\mathbf{x}) \big].$$

- *TreeSHAP* yields exact, fast values for tree-based models.
- Guarantees *local accuracy* and *consistency*.

LIME (Local Interpretable Model-agnostic Explanations) Fits a sparse surrogate (e.g. linear) model around each instance by perturbation:

$$\arg\min_{g \in G} \mathcal{L}\big(f,\, g,\, \pi_x\big) + \Omega(g),$$

where π_x weighs locality and $\Omega(g)$ enforces simplicity.

3. Algorithm Outline

1. **Data preparation:** Standardize numeric features (Income, Debt, Credit_Score, ...).
2. **Model training:** Fit a black-box (e.g. Random Forest).
3. **Explainer selection:**
   - `shap.TreeExplainer` for tree models or `shap.KernelExplainer` otherwise.
   - `lime.lime_tabular.LimeTabularExplainer` for surrogate fits.
4. **Compute explanations:**
   - *SHAP:* `explainer = shap.Explainer(model, X_background)`
     `shap_exp = explainer(X_new)` yields ϕ per feature.
   - *LIME:* `explainer = LimeTabularExplainer(...)`
     `lime_exp = explainer.explain_instance(...)`.
5. **Visualize:** Beeswarm/summary plots, force plots, bar charts.

4. Python Integration Example

```
import numpy as np
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import shap
import matplotlib.pyplot as plt

# 1) Synthetic financial dataset
np.random.seed(42)
N = 1000
df = pd.DataFrame({
    "Age": np.random.randint(18, 70, N),
    "Income": np.random.normal(60000, 15000, N),
    "Debt": np.random.normal(10000, 5000, N),
    "Credit_Score": np.random.randint(300, 850, N),
    "Years_Employed": np.random.randint(0, 40, N),
    "Num_Accounts": np.random.randint(1, 10, N),
})
# Approval rule: high score and low debt ratio
df["Approved"] = (
    (df["Credit_Score"] > 600) &
    (df["Debt"] / (df["Income"] + 1) < 0.5)
).astype(int)

# 2) Train/Test split
X = df.drop("Approved", axis=1)
y = df["Approved"]
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# 3) Train classifier
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
print(f"Train␣accuracy:␣{model.score(X_train,y_train):.3f}")
print(f"Test␣accuracy:␣␣{model.score(X_test,y_test):.3f}")

# 4) SHAP Explainer & values
explainer = shap.Explainer(model, X_train, seed=42)
shap_exp = explainer(X_test)              # (n_test, n_feat, 2)
# select positive class attributions
shap_pos = shap_exp.values[:,:,1]         # (n_test, n_feat)

# 5) Global summary plot
plt.figure(figsize=(8,6))
shap.summary_plot(
    shap_pos,
    X_test,
    feature_names=X_test.columns,
    show=False
)
plt.tight_layout()
plt.savefig("shap_summary.png", dpi=150)
plt.close()

# 6) Local force plot for sample #0
```

```
shap.initjs()
force_fig = shap.plots.force(
    explainer.expected_value[1],
    shap_pos[0],
    X_test.iloc[0],
    feature_names=X_test.columns,
    matplotlib=False
)
shap.save_html("force_plot.html", force_fig)
print("Saved: shap_summary.png and force_plot.html")
```