[3]:
```
# 🍣 Elite Ensemble: Blending Logistic + XGBoost Stackers

import os
import glob
import joblib
import pandas as pd
import numpy as np
from sklearn.metrics import average_precision_score

# 📂 Paths
logi_folder = "champion_stacks"
xgb_folder = "xgboost_stacks"
data_folder = "market_shock_synthetic_datasets"
base_model_folder = "champion_packages"

# 📦 Load All Stackers
logi_models = {os.path.basename(f).replace("_stacked.pkl", ""): joblib.load(f)
               for f in glob.glob(os.path.join(logi_folder, "*.pkl"))}

xgb_models = {os.path.basename(f).replace("_xgb_stacked.pkl", ""): joblib.load(f)
              for f in glob.glob(os.path.join(xgb_folder, "*.pkl"))}

# 📊 Load All Datasets
datasets = {os.path.basename(p).replace(".csv", ""): pd.read_csv(p)
            for p in glob.glob(os.path.join(data_folder, "*.csv"))}

# 📦 Load base models for meta-feature reconstruction
base_model_paths = [p for p in glob.glob(os.path.join(base_model_folder, "*.pkl")) if "_meta" not in p]
base_models = [joblib.load(p) for p in base_model_paths]

# ⚖️ Weighted Blend Config
weights = {"logi": 0.4, "xgb": 0.6}  # Feel free to tune

results = []
for name, df in datasets.items():
    if name in logi_models and name in xgb_models:
        X = df.drop("rare_event", axis=1)
        y = df["rare_event"]

        # 🍣 Rebuild meta-features for this dataset
        meta_features = []
        for model in base_models:
            try:
                meta_features.append(model.predict_proba(X)[:, 1])
            except AttributeError:
                meta_features.append(model.decision_function(X))

        X_meta = np.vstack(meta_features).T

        # Predict from each stacker
        logi_pred = logi_models[name].predict_proba(X_meta)[:, 1]
        xgb_pred = xgb_models[name].predict_proba(X_meta)[:, 1]
        blended = weights["logi"] * logi_pred + weights["xgb"] * xgb_pred

        pr_auc = average_precision_score(y, blended)
        baseline = y.mean()

        results.append({
            "Dataset": name,
            "PR AUC": round(pr_auc, 3),
            "Baseline": round(baseline, 3),
            "Lift": round(pr_auc - baseline, 3),
            "Liftoff": "🚀" if pr_auc >= 0.6 else "🍣"
        })

# 💾 Save leaderboard
results_df = pd.DataFrame(results).sort_values(by="Lift", ascending=False)
results_df.to_csv("ensemble_blended_lift_results.csv", index=False)
print(results_df)
```

```
                  Dataset  PR AUC  Baseline   Lift Liftoff
12     marketshock_easy_s2   0.890     0.136  0.755       🚀
11     marketshock_easy_s3   0.886     0.133  0.753       🚀
13     marketshock_easy_s1   0.853     0.141  0.712       🚀
9    marketshock_medium_s3   0.775     0.140  0.635       🚀
4          market_shock_sim   0.785     0.165  0.620       🚀
6      marketshock_hard_s3   0.767     0.155  0.612       🚀
5      marketshock_hard_s2   0.761     0.164  0.598       🚀
8    marketshock_medium_s1   0.738     0.141  0.597       🚀
3      marketshock_hard_s1   0.758     0.177  0.581       🚀
10   marketshock_medium_s2   0.720     0.144  0.576       🚀
1    marketshock_extreme_s3  0.802     0.264  0.538       🚀
0    marketshock_extreme_s2  0.789     0.255  0.534       🚀
2    marketshock_extreme_s1  0.780     0.261  0.519       🚀
7    stock_prediction_clean  0.964     0.502  0.462       🚀
```

[ ]: