作业2：基于XGBoost的高频交易噪声过滤与多空信号识别（15%）

1. 数据集

本实验以 Kaggle Jane Street 数据集为基础

1. 任务目标
2. 掌握高频交易数据中使用 XGBoost 进行回归建模的基本流程；
3. 构建动态加权机制，引导模型更加关注极端下跌样本，提高下行风险预测能力；（也可自行设计）
4. 完成回归模型的参数搜索、性能评估及可视化分析；
5. 分析模型在市场噪声过滤中的表现及其实际应用潜力
6. 作业内容：
7. 数据加载与特征选择：通过调用 load\_data 和 preprocess\_data 完成初始数据处理与特征工程，设定 responder\_6 作为预测目标；
8. 构造权重策略：设计三级加权机制，对普通下跌和极端下跌样本分别赋予不同权重，引导模型加大对下行风险的关注；
9. 参数搜索与模型训练：结合 RandomizedSearchCV 实现 XGBoost 参数优化，关注特征采样、树深度、正则化等超参数的组合效果；
10. 模型性能评估：引入标准 RMSE、加权 RMSE、上下行误差对比等多项指标，综合衡量模型预测精度与风险识别能力。

参考代码示例：

def asymmetric\_weight(y\_true):

*# 动态加权机制*

    weights = np.ones\_like(y\_true)

*# 普通下跌样本*

    downside\_mask = y\_true < 0

    weights[downside\_mask] = 2.5

*# 极端下跌样本（底部10%分位数）*

    extreme\_down = y\_true < np.percentile(y\_true[y\_true<0], 10)

    weights[extreme\_down] = 4.0

    return weights

train\_weights = asymmetric\_weight(y\_train.values)

*# 参数搜索空间（可基于结果优化）*

param\_dist = {

    'n\_estimators': [200, 250, 300],

    'learning\_rate': [0.05, 0.08, 0.1],

    'max\_depth': [5, 6],

    'subsample': [0.7, 0.8],

    'colsample\_bytree': [0.7, 0.8],

    'gamma': [0, 0.05, 0.1],

    'reg\_alpha': [0, 0.1, 0.5],

    'reg\_lambda': [0.1, 0.5, 1]

}

*# 初始化模型*

xgb\_model = XGBRegressor(

    objective='reg:squarederror',

    random\_state=42,

    tree\_method='hist',

    early\_stopping\_rounds=10

)

*# 随机搜索*

random\_search = RandomizedSearchCV(

    estimator=xgb\_model,

    param\_distributions=param\_dist,

    n\_iter=15,

    scoring='neg\_root\_mean\_squared\_error',

    cv=2,

    verbose=2,

    n\_jobs=4

)

* 1. 数据集及预处理要求

1. 数据集来源为 Jane Street 的高频交易记录，可复用作业一、二中以实现的清洗方法；
2. 特征值选取在features.csv和responder.csv中有所体现
3. 标签列为 responder\_6，可视为未来6步收益率或方向信号；
4. 预处理结果包括输入特征矩阵 X 和目标向量 y，并按照 8:2 划分训练集与测试集。
   1. 实验分析与报告

使用 XGBRegressor，结合 RandomizedSearchCV 进行 15 次迭代超参数搜索，优化目标为负均方根误差（neg RMSE），并设置早停机制防止过拟合。（以参数优化为关注点）

* + 1. 量化评估指标
* 标准RMSE：反映整体预测误差；
* 加权RMSE：综合考虑下行风险后的误差表现；
* 上行误差 vs 下行误差：用于对比模型在多空两个方向的识别能力。
* 上行误差降低率：衡量动态加权策略对极端负样本的建模贡献。

参考代码示例：

try:

    # 执行训练

    with parallel\_backend('threading', n\_jobs=4):

        random\_search.fit(X\_train, y\_train,

                         sample\_weight=train\_weights,

                         eval\_set=[(X\_test, y\_test)],

                         verbose=False)

    # 预测和评估

    best\_model = random\_search.best\_estimator\_

    y\_pred = best\_model.predict(X\_test)

    test\_weights = asymmetric\_weight(y\_test.values)

    # 计算指标

    standard\_rmse = np.sqrt(mean\_squared\_error(y\_test, y\_pred))

    weighted\_rmse = np.sqrt(np.mean(test\_weights \* (y\_test - y\_pred)\*\*2))

    upward\_error = np.mean(np.abs(y\_pred[y\_test>=0] - y\_test[y\_test>=0]))

    downward\_error = np.mean(np.abs(y\_pred[y\_test<0] - y\_test[y\_test<0]))

    print("\n=== Final Results ===")

    print(f"Best Parameters: {random\_search.best\_params\_}")

    print(f"Standard RMSE: {standard\_rmse:.4f}")

    print(f"Weighted RMSE: {weighted\_rmse:.4f}")

    print(f"Upside Error: {upward\_error:.4f}")

    print(f"Downside Error: {downward\_error:.4f}")

    print(f"Downside Reduction: {(1-downward\_error/upward\_error):.2%}")

    # 可视化分析

    plt.figure(figsize=(15,5))

    # 误差分布

    plt.subplot(1,3,1)

    plt.hist(y\_pred[y\_test<0] - y\_test[y\_test<0], bins=50, alpha=0.7, label='Downside')

    plt.hist(y\_pred[y\_test>=0] - y\_test[y\_test>=0], bins=50, alpha=0.7, label='Upside')

    plt.legend()

    plt.title("Error Distribution")

    # 预测vs实际

    plt.subplot(1,3,2)

    plt.scatter(y\_test, y\_pred, alpha=0.1)

    plt.plot([y\_test.min(), y\_test.max()], [y\_test.min(), y\_test.max()], 'r--')

    plt.xlabel("True Values")

    plt.ylabel("Predictions")

    plt.title("Prediction vs Actual")

    # 分位数误差

    plt.subplot(1,3,3)

    quantiles = np.linspace(0,1,11)

    q\_errors = [np.quantile(np.abs(y\_test - y\_pred), q) for q in quantiles]

    plt.plot(quantiles, q\_errors, 'bo-')

    plt.xlabel("Quantile")

    plt.ylabel("Absolute Error")

    plt.title("Quantile Error Analysis")

    plt.tight\_layout()

    plt.show()

    # 特征重要性

    plt.figure(figsize=(10,6))

    sorted\_idx = best\_model.feature\_importances\_.argsort()

    plt.barh(np.array(feature\_cols)[sorted\_idx], best\_model.feature\_importances\_[sorted\_idx])

    plt.title("Feature Importance")

    plt.show()

except Exception as e:

    print("Error occurred:", str(e))

    print(f"Data Shape - X\_train: {X\_train.shape}, y\_train: {y\_train.shape}")

    print(f"NaN Values - X\_train: {np.isnan(X\_train.values).sum()}, y\_train: {np.isnan(y\_train.values).sum()}")

* + 1. 实验分析
* 可视化分析：误差分布图：展示多空方向预测偏差分布；预测 vs 实际散点图：观察模型拟合度；分位误差分析图：揭示不同误差分布的鲁棒性；特征重要性柱状图：辅助解释模型偏好与输入特征间的关系。
* 模型总结：加权机制明显改善了模型在极端下行情况下的预测表现，具有一定的市场风险识别潜力；相较于标准RMSE，使用加权RMSE更能反映模型在实盘交易中对下跌风险的响应；实验过程中发现部分特征对模型贡献显著，后续可在特征筛选与维度压缩上进一步优化。

1. 作业提交要求（word或pdf）

代码：使用 Scikit-learn、XGBoost 等工具包等；

报告包括：引言、设计思想、流程图（模块介绍）、实现代码、实验结果、结果分析、总结、参考资料

本实验指导书只是提供思路指引，有能力的同学也可以自行定义分析方法并结合实际金融场景进行分析，只要有完整合理的说明和对应的结果截图即可。