Name: Justin Bucsa

E-mail: Justin.Bucsa@gmail.com

LinkedIn: https://www.linkedin.com/in/justin-bucsa/  
GitHub: https://github.com/jbucsa

Applicant for: HFT Quantitative Researcher/ Trader

Applicant at: Eqvilent

Brief Title: Report on Errors and Rectifications for pipeline.py

The following are key errors found when reviewing the pipeline.py file. This report does not include all the comments within the pipeline\_Bucsa\_Justin\_Submission.py file. Opening the pipeline\_Bucsa\_Justin\_Submission.py files and reviewing each comment will provide a more detailed explanation of the errors below followed by solutions to rectify the error in the code, respectively. This report is just a brief review of the errors found in the pipeline.py file.

**1. Logical Errors:**

* **Issue:** In print\_price\_std(df), \_high[\_high == \_low] = np.nan replaces values where high equals low with NaN, causing unintended skewing.
* **Solution:** Instead of replacing these values, drop rows where both high and low are missing to maintain the integrity of the dataset.

**2. Data Preprocessing Errors:**

* **Issue:** df[base\_cols] = df[base\_cols].fillna(0).astype(np.float32) replaces missing OHLCV values with zero, which distorts the data.
* **Solution:** Use forward fill (df[base\_cols] = df[base\_cols].ffill().astype(np.float32)) or drop rows with NaN values.
* **Issue:** df["y"] = df["close"].shift(100).fillna(0) shifts the target variable by 100 periods and fills missing values with 0, potentially introducing artificial values.
* **Solution:** Use df["y"] = df["close"].shift(1).fillna(method="bfill") to maintain accuracy.
* **Issue:** df["year"] = df["date"].dt.year % 10 reduces years to a single digit (e.g., 2023 becomes 3), leading to loss of valuable information.
* **Solution:** Extract full-year values as df["year"] = df["date"].dt.year.

**3. Feature Engineering Issues:**

* **Issue:** df[f"{name}\_diff\_{period}"] = df[name].shift(period) / df[name] can cause division by zero.
* **Solution:** Replace zero values with NaN before division: df[f"{name}\_diff\_{period}"] = df[name].shift(period) / df[name].replace(0, np.nan).
* **Issue:** df[f"{name}\_rank"] = df[name].rank() applies ranking without normalization.
* **Solution:** Normalize ranks within a specific window.
* **Issue:** MinMax scaling (df[name] = (df[name] - df[name].min()) / (df[name].max() - df[name].min())) may not be ideal.
* **Solution:** Use StandardScaler from sklearn.preprocessing.

**4. Machine Learning Model Issues:**

* **Issue:** Data is split twice (train\_test\_split used twice), which can lead to data leakage.
* **Solution:** Avoid redundant splits and verify necessity through a code review.
* **Issue:** "loss": "mape" is used in LightGBM, but MAPE is not always the best metric.
* **Solution:** Rename "loss" to "metric" to properly configure LightGBM.
* **Issue:** "learning\_rate": 1 is too high, leading to overfitting.
* **Solution:** Lower learning rate to values between 0.05 and 0.3.
* **Issue:** model.feature\_importance(importance\_type="split") provides less informative results.
* **Solution:** Use importance\_type="gain" for a more insightful feature importance analysis.

**5. Output Handling Issues:**

* **Issue:** Predictions are saved as pred\_to\_prod.csv without an index, making analysis difficult.
* **Solution:** Add an index column to the CSV output and handle potential NaN values:

pred\_test = pd.Series(pred\_test).interpolate().fillna(method='bfill').fillna(method='ffill').values

output\_df = pd.DataFrame({"index": range(len(pred\_test)), "prediction": pred\_test})

output\_df.to\_csv("pred\_to\_prod.csv", index=False)

For any additional details or information, please feel free to contact Justin directly at his e-mail address.