Group Lasso Transformer: Design and Implementation

Author: Sheng Wang a1903948
Team: RAIL-PG-2

1.Introduction

In our railway safety prediction project, the dataset combines diverse sources of information, including infrastructure characteristics, operational records (e.g., train speed and tonnage), and sensor-based measurements such as vibration and acceleration. While this diversity is valuable, it also brings challenges: redundant or noisy features may increase computation and reduce prediction accuracy.

Based on Sprint 1, I focus on implementing Embedded Methods for feature selection in Sprint 2. Regularization-based models are chosen for their ability to handle high-dimensional datasets with correlated features. Although standard Lasso regression is linear and limited in capturing nonlinear patterns, this limitation can be addressed by applying L1 or Group Lasso penalties within Transformer neural network architectures.

The next section introduces the implementation of the proposed approach, reports its performance on the Titan leaderboard, and highlights directions for future hyperparameter tuning.

2.Design

2.1 Basic Implementation Design

- Training Data Preparation + Feature Process & Engineering

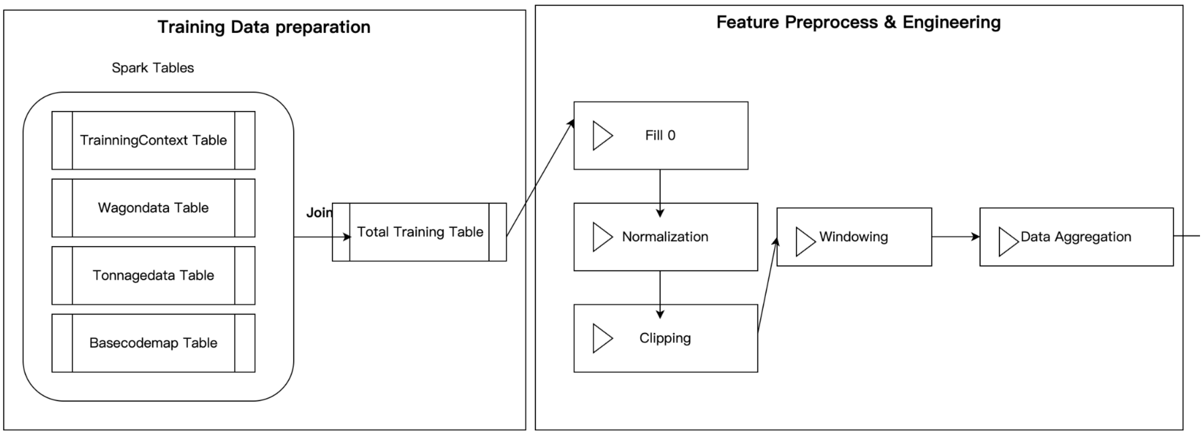


Figure1. Data Preparation & Feature Engineering Design

- Model Training + Output

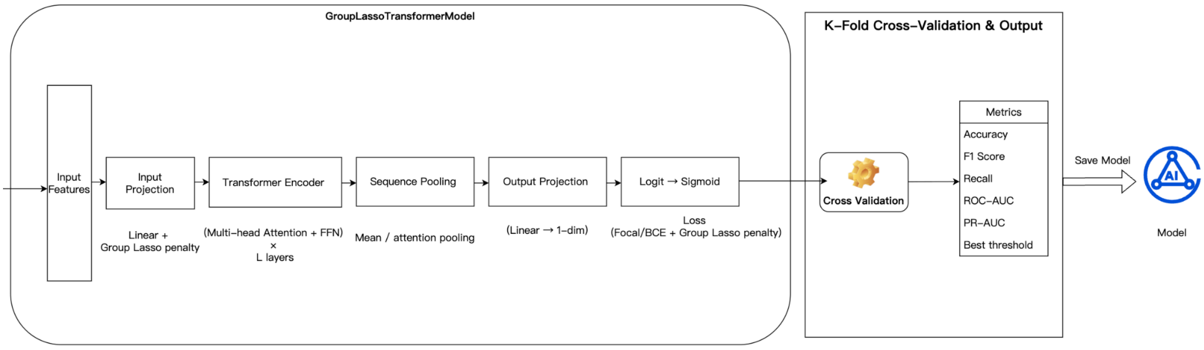


Figure2. Model Training and Output Design

2.2 InsightFactory Pipeline Building

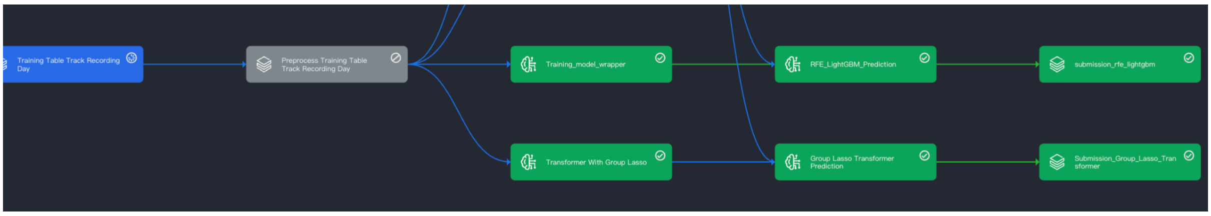

Training Table Track Recording Day
Preprocess Training Table Track Recording Day
Training_model_wrapper
RFE_LightGBM_Prediction
submission_rfe_lightgbm
Transformer With Group Lasso
Group Lasso Transformer Prediction
Submission_Group_Lasso_Transformer

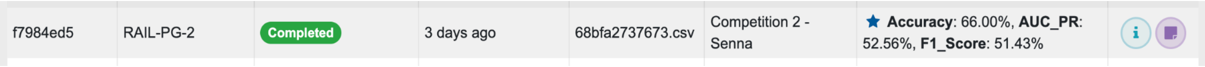
Figure3. Pipeline Design

The overall pipeline has been decomposed into the following key modules, each representing a specific stage in data preparation, model training, prediction, and submission:

1. Training Table Track Recording Task
2. Preprocess Training Table Track Recording Day
3. Transformer With Group Lasso
4. Group Lasso Transformer Prediction
5. Submission of Group Lasso Transformer

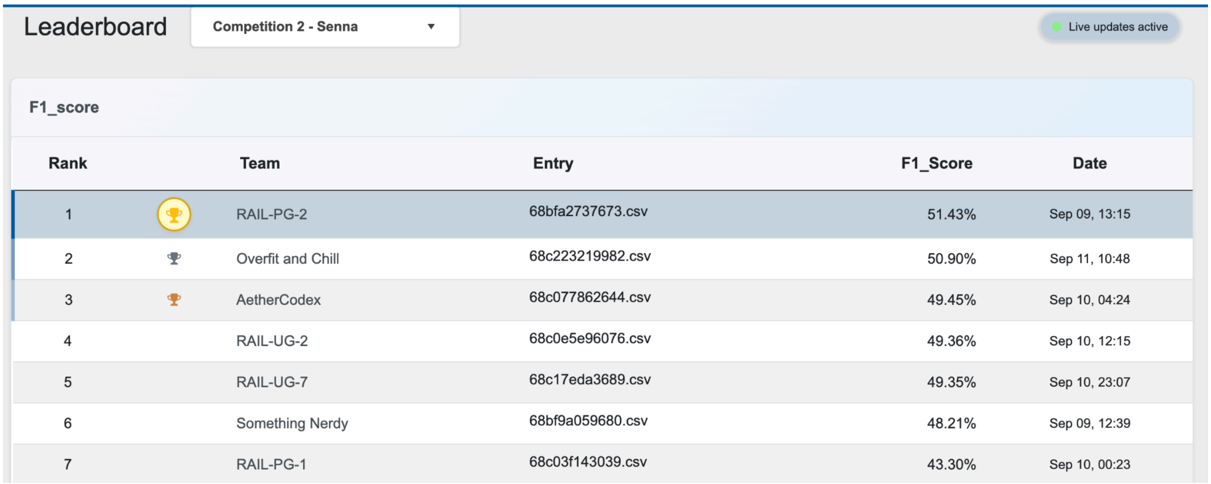
3.Results

Based on the first version of the basic implementation, model training and prediction were carried out, and the results were submitted to the leaderboard：



f7984ed5	RAIL-PG-2	Completed	3 days ago	68bfa2737673.csv	Competition 2 - Senna	★ **Accuracy**: 66.00%, **AUC_PR**: 52.56%, **F1_Score**: 51.43%	

Figure4. Model training scores



F1_score

Rank		Team	Entry	F1_Score	Date
1	🏆	RAIL-PG-2	68bfa2737673.csv	51.43%	Sep 09, 13:15
2	🏆	Overfit and Chill	68c223219982.csv	50.90%	Sep 11, 10:48
3	🏆	AetherCodex	68c077862644.csv	49.45%	Sep 10, 04:24
4		RAIL-UG-2	68c0e5e96076.csv	49.36%	Sep 10, 12:15
5		RAIL-UG-7	68c17eda3689.csv	49.35%	Sep 10, 23:07
6		Something Nerdy	68bf9a059680.csv	48.21%	Sep 09, 12:39
7		RAIL-PG-1	68c03f143039.csv	43.30%	Sep 10, 00:23

Figure5. Leaderboard results

Submitted version parameter settings:

- Time Window Aggregation:
  Training Data: -30days ~ training recording date
  Prediction Data: -30days ~ Prediction recording date

- Feature Grouping and Regularization Penalties:
  Feature Grouping: No grouping, each feature treated as an individual group
  Regularization Penalties: No penalty applied, L1 = L2 = 0

- Clipping Strategy Comparison:
  percentile=(0.5, 99.5)

- Handling Imbalanced Data:
  None

- Model Architecture and Hyperparameter:
  1. **TransformerEncoderLayer**

```
encoder_layer = nn.TransformerEncoderLayer(
            d_model=96,
            nhead=4,
            dim_feedforward=192,
            dropout=0.1,
            activation='relu',
            batch_first=True,
            norm_first=True
```

```
        )
```

2. **Transformer**

```
  self.transformer = nn.TransformerEncoder(
      encoder_layer,
      num_layers=1
  )
```

3. **Output_head**

```
d_model=96
dropout=0.1
        output_head = nn.Sequential(
                nn.Linear(d_model, d_model // 2),
                nn.LayerNorm(d_model // 2),
                nn.GELU(),
                nn.Dropout(dropout),
                nn.Linear(d_model // 2, 1)
        )
```

- Threshold Tuning Strategies:
  F1 maximization

4.Next plan

To further optimize model performance and enhance prediction robustness, the next step will involve parameter tuning across data preprocessing, feature engineering, feature selection, model architecture, and evaluation strategies. The detailed plan is as follows:

4.1 Time Window Aggregation

- Objective: Compare the performance of training and prediction under different time window lengths (e.g., 7 days, 30 days).
- Stragegies:
  1. Compare different Time Window Lengths Strategies

4.2 Feature Grouping and Regularization Penalties

- Objective: Explore different feature grouping schemes(prefix-based,domain-knowledge-based) under Group Lasso regularization.

- Stragegies:
  1. Effect of L1/L2 regularization factors (λ1, λ2) on feature sparsity
  2. Preservation or elimination of key feature groups

4.3 Clipping Strategy Comparison

- Objective:Investigate the impact of different clipping strategies on feature distributions
- Parameters to explore:
  1. Percentile clipping (e.g., [0.5, 99.5])
  2. Fixed clipping ([-10, 10])
  3. None (no clipping)

4.4 Handling Imbalanced Data

- Objective: Address datasets imbalance
- Methods to explore:
  1. Oversampling/undersampling
  2. Weighted loss functions (Weighted BCE, Focal Loss)

4.5 Model Architecture and Hyperparameter Optimization

- Objective: Optimize model architecture within the Transformer and Group Lasso framework.
- Parameters to explore:
  1. Number of layers (num_layers)
  2. Number of attention heads (nhead)
  3. Hidden dimension (d_model)
  4. Dropout ratio
  5. Epochs and batch size

4.6 Threshold Tuning Strategies

- Object: Investigate the effect of different threshold selection policies on classification performance.
- Strategies to Explore:
  1. F1 maximization
  2. Quantile-based thresholds
  3. Precision-at / Recall-at thresholds

4. Youden index
5. Cost-sensitive thresholds (considering FP/FN costs)