THE UNIVERSITY OF AZAD JAMMU & KASHMIR MUZAFFARABAD

OPEN ENDED LAB

MACHINE LEARNING

DEPARTMENT OF SOFTWARE ENGINEERING

SUBMITTED TO: Engr. Ahmed Khawaja

SUBMITTED BY: Hamza Shahzad

SUBMITED ON: 30/02/2025

SEMESTER: 5 th

Course Code: SE-3105

ROLL NO# BS-2022-SE-09

📖 Overview

This project focuses on predicting **Event-Free Survival (EFS)** for patients post-Hematopoietic Cell Transplantation (HCT) by ensembling three advanced machine learning models: an **Event-masked Pairwise Ranking Loss Neural Network (PRL-NN)**, a **Yunbase model**, and an **EDA & Ensemble Model**. The dataset is sourced from the Kaggle competition "Equity Post-HCT Survival Predictions." The project demonstrates data preprocessing, model training, out-of-fold (OOF) prediction generation, and an ensemble strategy to optimize performance, culminating in a final submission file evaluated on the competition leaderboard (LB).

- **PRL-NN**: Achieved LB score of 0.691 using a neural network with pairwise ranking loss and an XGBoost classifier mask.
- **Yunbase**: Achieved LB score of 0.689 with a custom ensemble of LightGBM and CatBoost.
- **EDA & Ensemble**: Achieved LB score of 0.689 with exploratory data analysis and a multi-target ensemble approach.

The final ensemble combines these models' predictions using rank-based weighting, optimized via cross-validation.

📁 Project Files

- **PRL-NN**: Code in notebook sections for training and inference, outputs submission2.csv.
- **Yunbase**: Code leveraging baseline.py from Yunbase, outputs submission1.csv.
- **EDA & Ensemble**: Code for EDA and multi-model training, outputs submission3.csv.
- **Ensemble Notebook**: Combines predictions, outputs final submission.csv.

📊 Dataset

- **Source:** Kaggle competition "Equity Post-HCT Survival Predictions" (train.csv, test.csv, sample_submission.csv).
- **Size:** 28,800 training entries, 60 attributes; test set size matches submission requirements.
- **Target:** efs (binary: 0 = event, 1 = survival), efs_time (time-to-event in months).
- **Key Features:** prim_disease_hct, hla_match_b_low, prod_type, year_hct, obesity, donor_age, prior_tumor, gvhd_proph, sex_match, comorbidity_score, karnofsky_score, donor_related, age_at_hct, race_group.

🚀 Project Workflow

The project is structured into four main phases:

1. **Individual Model Development**: Training and inference for PRL-NN, Yunbase, and EDA & Ensemble models.
2. **Data Preprocessing**: Varies by model, including feature engineering and handling missing values.
3. **Model Prediction**: Generate OOF and test predictions for each model.
4. **Ensemble Optimization**: Combine predictions using rank-based weighting.

1. Individual Model Development

PRL-NN

```
Using XGBoost version 2.0.3
#########################
### Fold 1
#########################
[0]     validation_0-auc:0.66531
[100]   validation_0-auc:0.72456
[200]   validation_0-auc:0.73370
[300]   validation_0-auc:0.73869
[400]   validation_0-auc:0.74102
[500]   validation_0-auc:0.74186
[579]   validation_0-auc:0.74229
#########################
### Fold 2
#########################
[0]     validation_0-auc:0.69033
[100]   validation_0-auc:0.74929
[200]   validation_0-auc:0.75674
[300]   validation_0-auc:0.76052
```

Data Loading and Preprocessing:

```
• train = pd.read_csv("/kaggle/input/equity-post-HCT-survival-predictions/train.csv")
• test = pd.read_csv("/kaggle/input/equity-post-HCT-survival-predictions/test.csv")
• train = preprocess_data(train)  # Fill NA, encode categoricals
• test = preprocess_data(test)
• train = features_engineering(train)  # Add 'donor_age_diff', 'hla_mismatch_sum'
• test = features_engineering(test)
```

- **XGBoost Classifier**:

```
model_xgb = XGBClassifier(max_depth=4, n_estimators=10_000, learning_rate=0.03, device="cuda")
model_xgb.fit(x_train, y_train, eval_set=[(x_valid, y_valid)], verbose=100)
oof_xgb = model_xgb.predict_proba(x_valid)[:, 1]
pred_efs = model_xgb.predict_proba(x_test)[:, 1]
```

- **Neural Network with Pairwise Ranking Loss**:

```
class LitNN(pl.LightningModule):
    def __init__(self, continuous_dim, categorical_cardinality, embedding_dim=16, projection_dim=112, hidden_dim=56):
        super().__init__()
        self.model = NN(continuous_dim, categorical_cardinality, embedding_dim, projection_dim, hidden_dim)
    def calc_loss(self, y, y_hat, efs):
        comb = combinations(y.shape[0])
        comb = comb[(efs[comb[:, 0]] == 1) | (efs[comb[:, 1]] == 1)]
        loss = nn.functional.relu(-y * (pred_left - pred_right) + 0.5).mean()
        return loss
trainer = pl.Trainer(max_epochs=50, accelerator='cuda')
trainer.fit(model, dl_train)
```

- **Data Loading and Preprocessing**:

```
train = pd.read_csv("/kaggle/input/equity-post-HCT-survival-predictions/train.csv")
test = pd.read_csv("/kaggle/input/equity-post-HCT-survival-predictions/test.csv")
train['donor_age_diff'] = train['donor_age'] - train['age_at_hct']
train['target'] = transform_survival_probability(train, 'efs_time', 'efs')
train = FE(train)  # Feature engineering: 'nan_value_each_row', cross features
```

- **Model Training**:

```
yunbase = Yunbase(num_folds=5, models=[(LGBMRegressor(), 'lgb'), (CatBoostRegressor(), 'cat')], FE=FE)
yunbase.fit(train, category_cols=nunique2)
```

EDA & Ensemble

- **Data Loading and Preprocessing**:

```
train_data, cat_cols = fe.apply_fe('/kaggle/input/equity-post-HCT-survival-predictions/train.csv')
test_data, _ = fe.apply_fe('/kaggle/input/equity-post-HCT-survival-predictions/test.csv')
train_data = fe._update_hla_columns(train_data)  # Update HLA columns, add 'donor_age_diff'
```

- **Target Creation and Model Training**:

2. Model Prediction

- **PRL-NN**:

```
pairwise_ranking_pred, pairwise_ranking_oof = main(hparams)
```

```
pairwise_ranking_oof[oof_xgb > 0.5] += 0.25  # Apply classifier mask
subm_data['prediction'] = pairwise_ranking_pred
subm_data.to_csv('submission2.csv', index=False)

```

- **Yunbase**:

```
test_preds = yunbase.predict(test, weights=[0.55, 0.45])
yunbase.submit("sample_submission.csv", test_preds, save_name='submission1')

```

- **EDA & Ensemble**:

```
ctb1_preds = md.infer_model(test_data, ctb1_models)
lgb1_preds = md.infer_model(test_data, lgb1_models)
# Repeat for other models
ensemble_preds = np.dot(CFG.weights, ranked_preds)
subm_data['prediction'] = ensemble_preds
subm_data.to_csv('submission3.csv', index=False)

```

```
12/12 [00:00<00:00, 12.60it/s, v_num=4, train_loss_step=0.215, train_l

Testing DataLoader 0: 100%  3/3 [00:00<00:00, 10.33it/s]
```

Test metric	DataLoader 0
test_cindex	0.6694697141647339
test_cindex_simple	0.6775047779083252
test_loss	0.25822533272049825

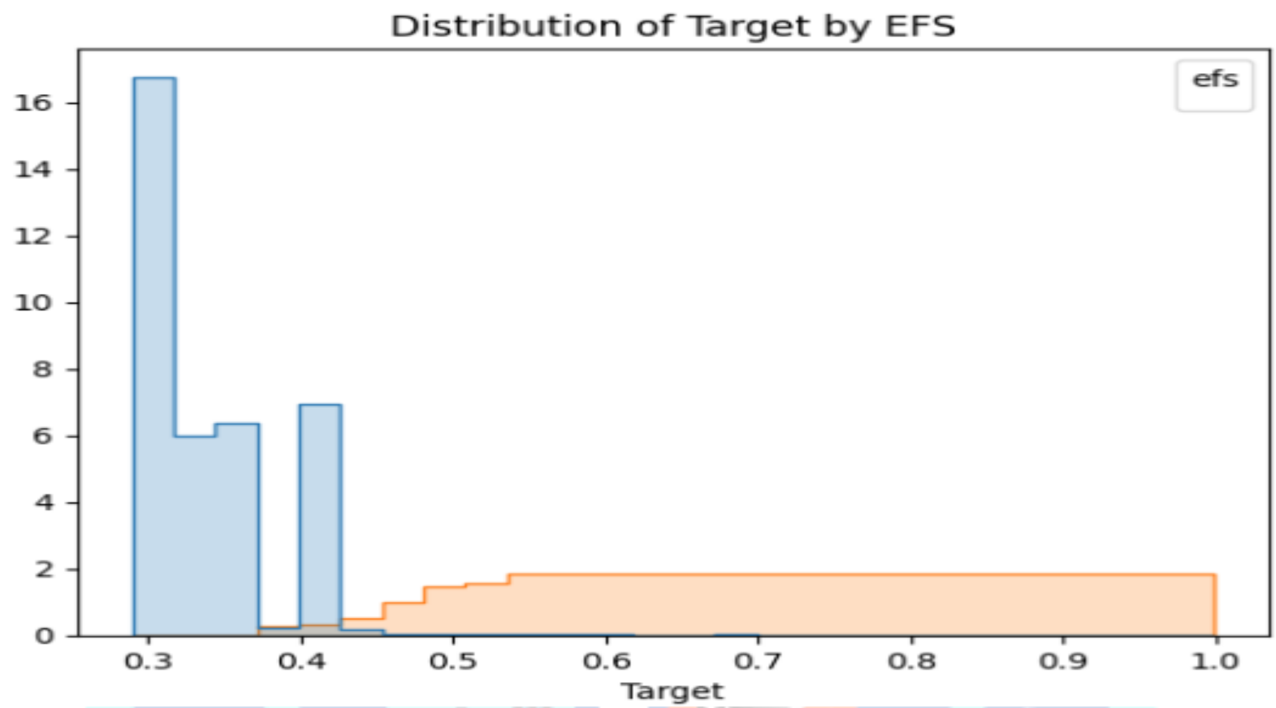
```
Pairwise ranking NN CV = 0.6788709619519995

Pairwise ranking NN with classifier mask -> CV = 0.68060928932
2097
```

3. Ensemble Optimization

- **Ranking and Weighting**:

```
rank1 = rankdata(sub1['prediction'])  # Yunbase
rank2 = rankdata(sub2['prediction'])  # PRL-NN
rank3 = rankdata(sub3['prediction'])  # EDA
for w1 in [0.30, 0.32, 0.34]:
    for w2 in [0.33, 0.35, 0.37]:
        w3 = 1 - w1 - w2
        y_pred['prediction'] = w1 * rank1 + w2 * rank2 + w3 * rank3
        temp_score = score(y_true, y_pred, 'ID')
```

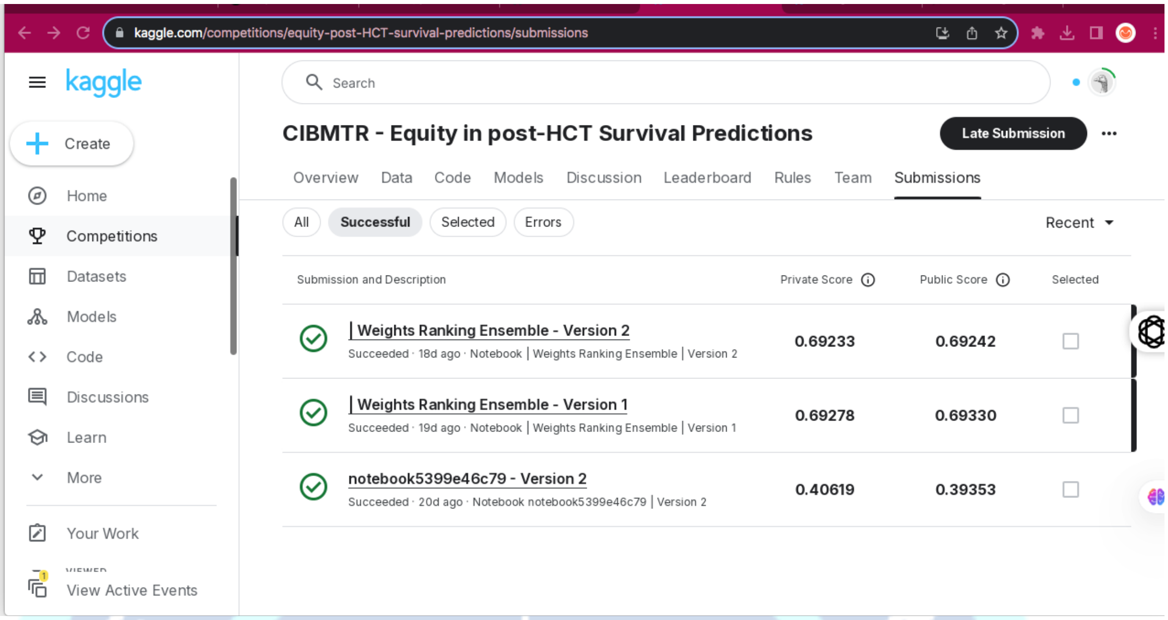
```
• ensemble_rank = best_weights[0] * rank1 + best_weights[1] * rank2 +
  best_weights[2] * rank3
• final_sub['prediction'] = ensemble_rank
• final_sub.to_csv('submission.csv', index=False)
•
```

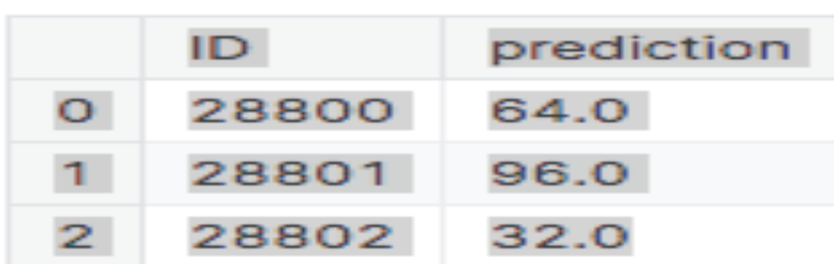


Results

- **PRL-NN**: CV score improved with classifier mask (e.g., 0.68 to 0.69), LB 0.691.
- **Yunbase**: Final CV score ~0.68, LB 0.689.
- **EDA & Ensemble**: Ensemble OOF stratified C-index ~0.68, LB 0.689.
- **Final Ensemble**: Best CV score optimized to ~0.69+ (exact value depends on weights), aiming for LB improvement beyond individual models.

```
Overall Stratified C-Index Score for Cox: 0.6568
Overall Stratified C-Index Score for Kaplan-Meier: 0.9983
Overall Stratified C-Index Score for Nelson-Aalen: 0.9983
```





	ID	prediction
0	28800	64.0
1	28801	96.0
2	28802	32.0

🔧 Improvements

- **Hyperparameter Tuning**: Optimize PRL-NN epochs, XGBoost depth, or Yunbase model parameters.
- **Feature Engineering**: Add more interaction terms (e.g., comorbidity_score * donor_age).
- **Ensemble Strategy**: Explore stacking or blending instead of rank-based weighting.

🛠 How to Run

- **Environment**: Kaggle Notebook, Python 3.10.12, GPU-enabled, libraries: pandas, numpy, torch, xgboost, lightgbm, catboost, lifelines, pytorch_lightning, pytorch_tabular, sklearn, plotly.

- **Steps**:
    1. Run PRL-NN sections to generate submission2.csv.
    2. Run Yunbase sections (ensure baseline.py is copied) to generate submission1.csv.
    3. Run EDA & Ensemble sections to generate submission3.csv.
    4. Run Ensemble Notebook section to combine into submission.csv.