

**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.

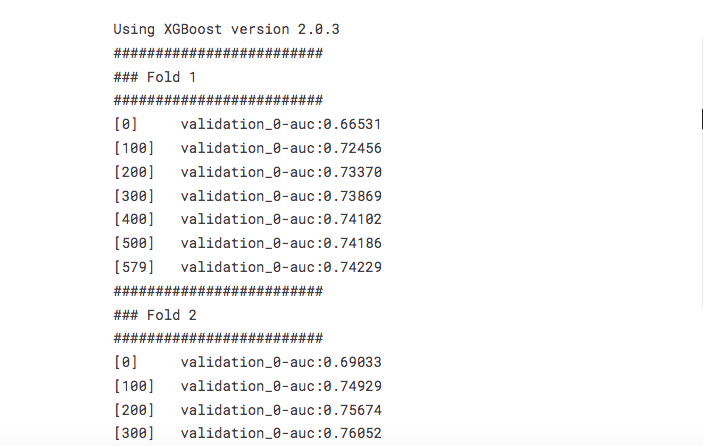
### 🚀 Project Workflow

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

****

**Data Loading and Preprocessing**:

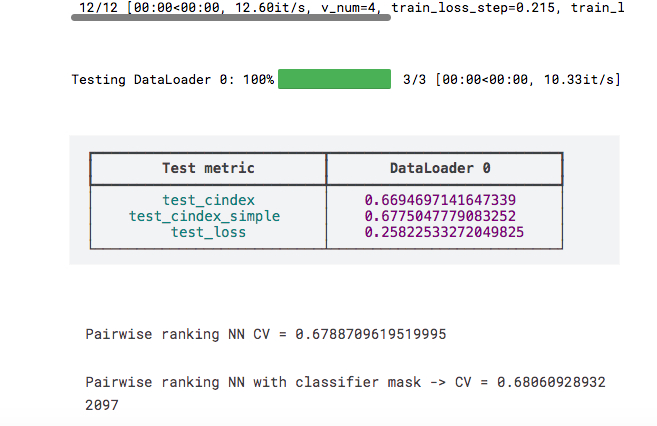
* train = pd.read\_csv("/kaggle/input/equity-post-HCT-survival-predictions/train.csv")
* test = pd.read\_csv("/kaggle/input/equity-post-HCT-survival-predictions/test.csv")
* train = preprocess\_data(train) # Fill NA, encode categoricals
* test = preprocess\_data(test)
* train = features\_engineering(train) # 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)



#### 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)

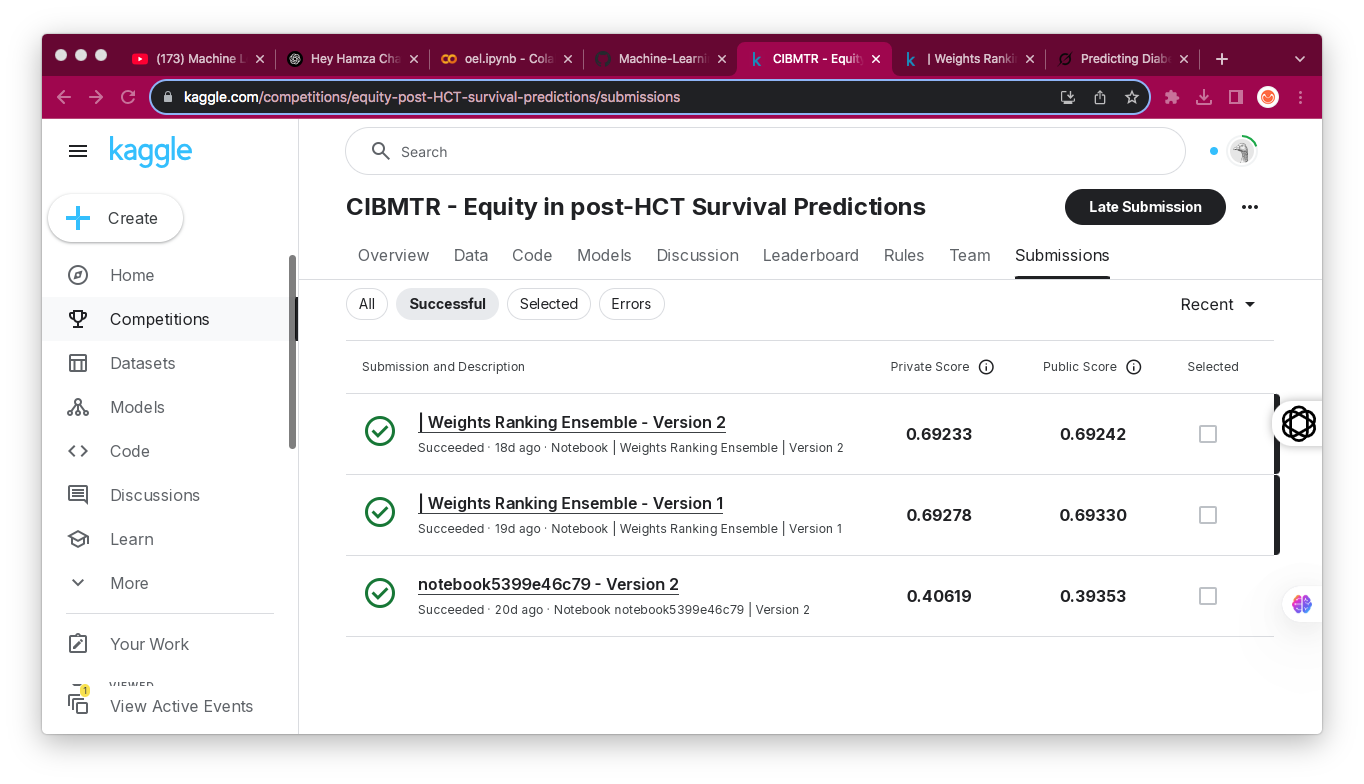
### 📈 Results

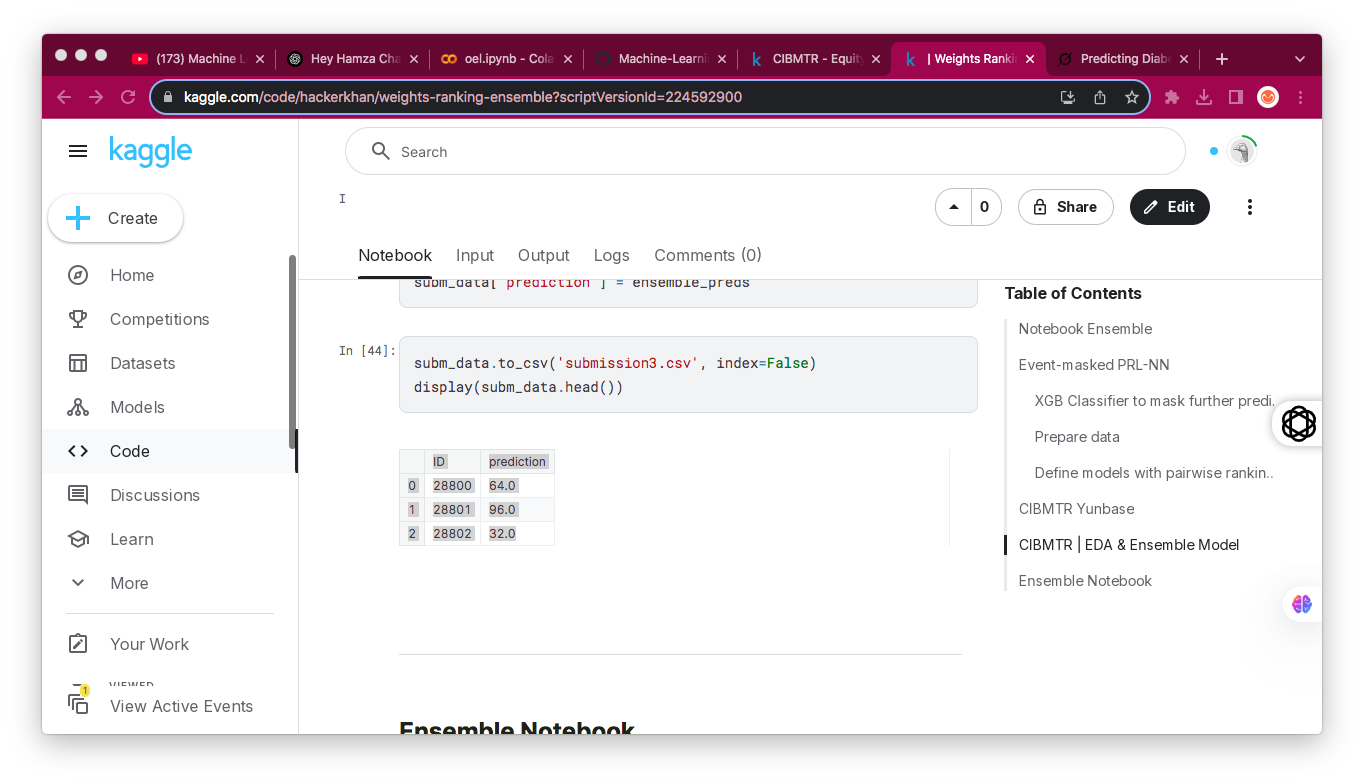
* **PRL-NN**: CV score improved with classifier mask (e.g., 0.68 to 0.69), LB 0.691.
* **Yunbase**: Final CV score ~0.68, LB 0.689.
* **EDA & Ensemble**: Ensemble OOF stratified C-index ~0.68, LB 0.689.
* **Final Ensemble**: Best CV score optimized to ~0.69+ (exact value depends on weights), aiming for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





### 🔧 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.