Technical Report on Fine-Tuning Parameters for Titanic Survival Models

Name: Ma Yingying Student ID: 2510433

Introduction

This technical report delineates the development and optimization of a deep learning framework designed to predict survival outcomes for the Titanic dataset. The primary objective is to explore a hybrid modeling approach integrating traditional feature engineering with modern transfer learning techniques. By leveraging a frozen pre-trained feature extractor and a custom "Refiner" network, the architecture preserves primary input integrity while utilizing high-level latent representations.

1 Data Preprocessing & Feature Engineering

Automated analysis of unique value counts and distributions was performed to determine variable utility. Features with high cardinality and near-uniform distributions (Feature #1 and Feature #7, see Fig. 1) were identified as noise-heavy and excluded. Furthermore, Feature #9 was removed due to excessive missing values.[1]

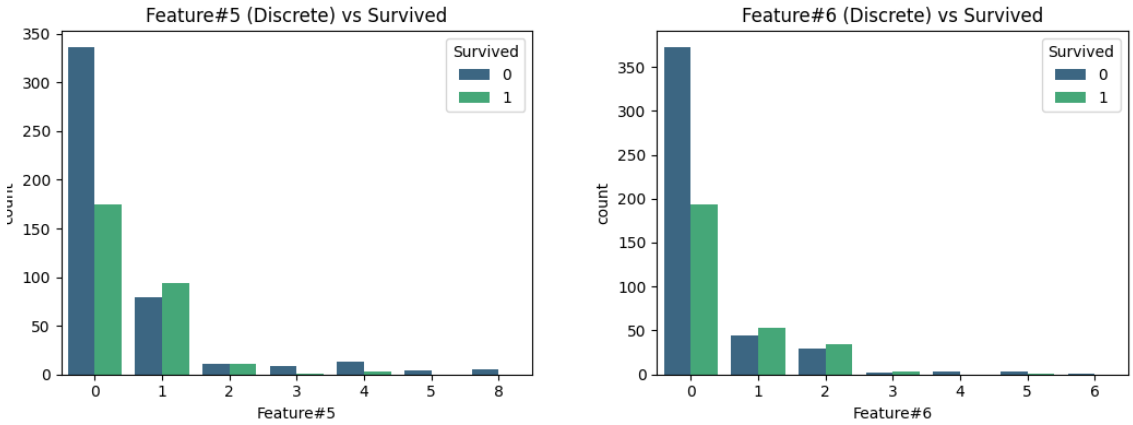

Figure 1: Distribution of individual features and survival relationships [2].

1.1 Feature Engineering and Encoding

Raw string data was processed via regular expressions to extract titles, which were grouped into five categories (Mr, Miss, Mrs, Master, Rare) and mapped to an ordinal scale based on empirical training survival rates [3].

1.2 Handling Missingness

Missing values were addressed via context-aware imputation. Feature #4 (Age) was populated using a Title-based median strategy [4]. This preserved age-social status correlations more effectively than a global mean.

1.3 Synthesis of Feature #56

A composite feature, Feature #56, was synthesized by summing Feature #5 and Feature #6 to represent total family size and reduce data sparsity [5].

2 Feature Selection Method

Feature selection utilized statistical visualization to maximize signal and minimize redundancy. A Pearson correlation matrix identified variables with the strongest linear relationships to the target [6]. Based on this heatmap (Fig. 2), five high-value attributes were selected: Feature #2, Feature #3, Title_Numeric, Feature #8, and Feature #10 [7].

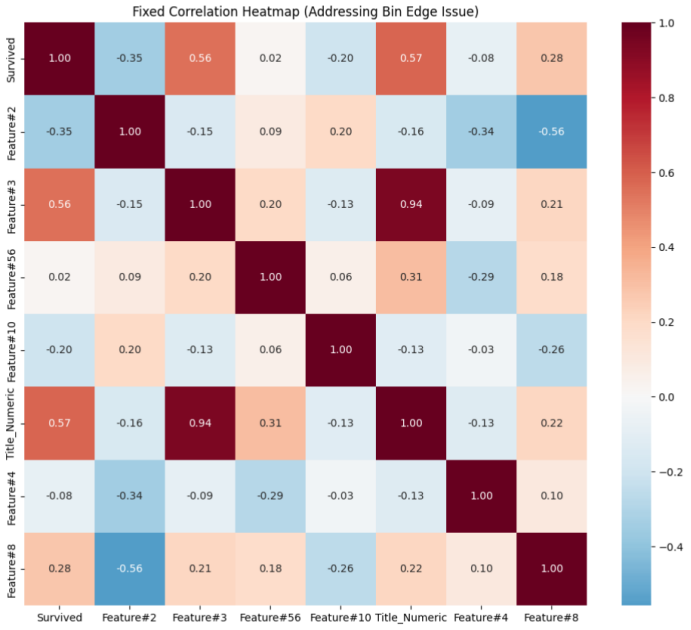

Figure 2: Pearson Correlation Heatmap for dimensionality reduction [6].

3 Model Design

The project employs a Hybrid Residual Neural Network built in PyTorch.

3.1 Transfer Learning and Feature Extraction

A pre-trained model (`pretrained_network.pth`) acts as a backbone [8]. All its parameters were frozen (`requires_grad = False`) to function as a fixed 3-dimensional latent feature extractor [9].

3.2 Residual Architecture and Feature Concatenation

The 3-dimensional frozen output is concatenated with the 5 selected input features, forming an 8-dimensional vector for the refiner network [9]. This ensures access to both abstract and primary data signals.

3.3 The Refiner Network

The "Refiner" is a Multi-Layer Perceptron (MLP) designed for small-sample robustness [9]:

- **Normalization**: Batch Normalization is applied for stability and convergence [9].
- **Activations**: LeakyReLU (slope 0.1) and ReLU are used for non-linear decision boundaries [9].
- **Regularization**: A Dropout layer (rate 0.2) prevents overfitting [9].
- **Output**: A Sigmoid activation facilitates binary classification [9].

4 Results & Training Strategy

Training was conducted for 100 epochs using the Adam optimizer and Binary Cross Entropy (BCE) loss [9].

4.1 Dynamic Learning Rate Optimization

A `ReduceLROnPlateau` scheduler monitored training loss, reducing the learning rate by a factor of 0.5 if loss stalled for 10 epochs to navigate complex minima [9].

4.2 Validation Monitoring and Early Stopping

An Early Stopping strategy terminated training if validation accuracy did not improve for 20 iterations [9]. The system saved only the optimal weights (`best_refiner_weights.pth`), which were reloaded for final test-set inference [10].

References

[1] Python Script Cell [In 8]: Data cleaning and dropping features with excessive missing values.

[2] Python Script Cell [In 10]: Exploratory Data Analysis and feature distribution visualization.

[3] Python Script Cell [In 11]: Regex-based title extraction and ordinal mapping.

[4] Python Script Cell [In 12]: Context-aware imputation for missing age values.

[5] Python Script Cell [In 13]: Feature synthesis Feature#56.

[6] Python Script Cell [In 19]: Pearson correlation matrix and heatmap generation.

[7] Python Script Cell [In 20]: Final feature set selection for model input.

[8] Python Script Cell [In 32]: Pretrained network shell definition and loading.

[9] Python Script Cell [In 36]: Hybrid Residual Model architecture, training loop, and optimizer setup.

[10] Python Script Cell [In 37]: Model reloading, inference, and submission generation.