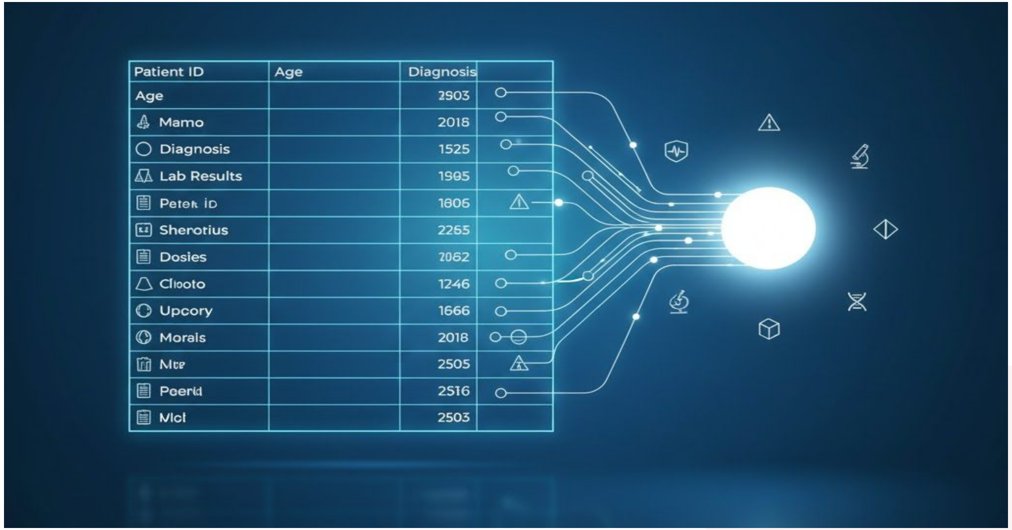
Embeddings for Prediction in Clinical Data



Supervisor:

Prof. Dr. Myra Spiliopoulou

Team Members:

- Sourav Salkoppalu Lingaraju
- Chethan Chinnabhandara Nagraj
- Sajith Kumar Santhosh
- Bhavesh Sharad Yeole

Introduction—Milestone: 2

Building Robust Representations via Simulated Imbalance, Preprocessing, and Tabular Embedding Methods

- **Goal:** Model real-world data variations using a simulator and evaluate its impact (for the proposed methods) on the downstream task.
- **Preprocess** data to ensure clean, consistent, and analysis-ready inputs.
- Generate **embeddings** using **TABPFN** and **DEEPFS** to enhance predictive modeling.
- Aim to create robust representations suitable for **downstream** clinical **prediction** tasks.

Simulator : Dataset Variant Generator

Goal : Use the **simulator** to generator a dataset variant by **downsampling** and inducing **skew** to the dataset.

Main Functionality :

1) **Downsampling → creates wide datasets**

- removes the *same proportion* of rows from each class to maintain distribution shape.

2) **Inducing skew**

- removes rows *only* from the chosen class to create controlled imbalance.

WORKFLOW:

Step 1 : Initialize the `**RowDeletionSimulator**` class with your dataframe

`simulator = RowDeletionSimulator(df, target_col="Label",random_state=42)`

- **random_state** → ensures random selection of row → reproducible experiments)
- **target_column** → specifies the target variable of the dataset.

Step 2-a: Apply **downsampling**:

`downsampled_df = simulator.downsample(target_ratio=0.7)`

- **target_ratio** → specifies the percentage of rows to be removed

Step 2-b: Induce **skew**:

`mild_skew_df = simulator.skew(target_class=1, target_skew=1/4)`

- **target_class** → class selected for targeted reduction
- **target_skew** → desired final proportion (minority:majority) for the selected class

Execution Results

- Original Dataset **Shape**: (477, 198)
  Distribution of Target Variable('Labels') - 0 → 359 and 1 →118

- Result after **downsampling 70.0% (target_ratio)** records
  Dataset **Shape** : (142, 198)
  Distribution of Target Variable('Labels') - 0 → 107 and 1 → 35

- Skew Induction for **target class =1 and target skew** as **60.0%**
  Dataset **Shape** : (406, 198)
  Distribution of Target Variable('Labels') - 0 →359 and 1 → 47

Data Pre-Processing

- Presented by: Bhavesh Sharad Yeole
- Status: Implemented
- Loaded train and test datasets into dataframes.
- Removed duplicates and checked missing percentages.
- Used the simulator to generate HDLSS conditions from the original dataset.
- Automatically detected continuous, integer, binary, and categorical columns.
  So each feature gets the right preprocessing without assumptions.
- Normalisation before feature selection. To ensure numeric values are on a comparable scale so the selector doesn't favor larger-range features.

Algorithm 1: DeepFS

- Presented by: Sajith Kumar Santhosh
- Built a supervised autoencoder to extract a low-dimensional representation of the data and predict the target. This ensures features capture target-relevant patterns; trained with supervised loss plus a small reconstruction penalty.
- Encoded the data into a latent space and normalized the latent dimensions to [0,1]. Normalization makes each latent dimension comparable, so feature importance can be evaluated fairly.
- Computed multivariate ranks using a Sobol sequence to map samples into a uniform, quasi-random space. This captures complex dependencies between samples efficiently using the Hungarian assignment.

K. Li, F. Wang, L. Yang, and R. Liu, "Deep feature screening: Feature selection for ultra high-dimensional data via deep neural networks," Neurocomputing, vol. 538, pp. 126186, 2023.

Algorithm 1: DeepFS

- Calculated robust dependence (RdCorr) between each original feature and the latent space. Rank-based correlation identifies features strongly associated with the target.
- Ranked features by their dependence scores in descending order. This prioritizes the most informative features for automatic selection.
- Selected the most informative subset using the jump ratio method. Detects the largest drop in ranked scores to estimate the optimal number of features to keep.
- What is left to implement? Downstream Classification and Performance Metric Evaluation of the algorithm

```
Label
0    125
1     41
Name: count, dtype: int64
Preprocessing complete. Dataset ready for feature selection.
Choose feature selection algorithm ('deepfs' or 'graces'): deepfs
User selected: DEEPFS
Running DeepFS feature selection...
Selected feature indices (relative to numeric features): [ 13  34 152  49  90 166 184 111 151  44 118  29 124 155 193 164   5 138
   6   8 187 191 136 108]
Corresponding scores: [0.16325516 0.16133109 0.15766754 0.15523534 0.15325436 0.15237229
 0.15229641 0.15118331 0.15117721 0.15090048 0.15080819 0.15062801
 0.14996962 0.14918408 0.14915565 0.14898281 0.14888936 0.14868301
 0.14860588 0.14841931 0.14836493 0.1481192  0.14797176 0.1477222 ]
Feature selection complete. Output saved as data\processed_dia_dataset_deepfs.csv
Selected features: ['Chi4v', 'LabuteASA', 'fr_ether', 'NumAliphaticHeterocycles', 'SlogP_VSA11', 'fr_methoxy', 'fr_pyridine', 'fr_Al_COO', 'fr_ester', 'MolMR', 'fr_Ar_O
H', 'HeavyAtomMolWt', 'fr_HOCCN', 'fr_halogen', 'fr_thiophene', 'fr_lactam', 'Chi1', 'fr_amide', 'Chi1n', 'Chi2n', 'fr_sulfonamd', 'fr_thiazole', 'fr_alkyl_halide', 'VS
A_EState7']
```

Algorithm 2: GRACES

- Presented by: Sajith Kumar Santhosh
- Used a graph-based neural network to model feature interactions and predict the target. This captures complex dependencies between features while considering their relationships through a learned graph structure.
- Initialized features with a bias term and computed univariate F-scores. Provides a starting signal for important features, helping the model prioritize meaningful gradients.
- Iteratively trained a Graph Convolutional Network (GCN) with multiple dropout versions. Dropout introduces stochasticity to simulate ensemble effects, ensuring robust gradient estimation for feature importance.

C. Chen, S. T. Weiss, and Y.-Y. Liu, "Graph convolutional network-based feature selection for high-dimensional and low-sample size data," Bioinformatics, vol. 39, no. 4, pp. btad135, 2023.

Algorithm 2: GRACES

- Calculated input gradients for each feature across dropout models and averaged them. Gradient magnitudes indicate which features have the strongest influence on the prediction, serving as a ranking signal.
- Selected features iteratively by choosing the one with the largest adjusted gradient norm. This builds the feature subset step-by-step until reaching the target number of selected features.
- What is left to implement? Downstream Classification and Performance Metric Evaluation of the algorithm

```
Label
0    125
1     41
Name: count, dtype: int64
Preprocessing complete. Dataset ready for feature selection.
Choose feature selection algorithm ('deepfs' or 'graces'): graces
User selected: GRACES
Running GRACES feature selection...
Enter the number of features to select (integer): 50
F:\Sem 4 ovgu\KMD\KMD Project\.venv\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:110: UserWarning: Features [ 57  86  99 101 103 104 105 106 107
 108 123 129 134 142 143 144 148 160
 161 172 183 192] are constant.
  warnings.warn("Features %s are constant." % constant_features_idx, UserWarning)
F:\Sem 4 ovgu\KMD\KMD Project\.venv\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:111: RuntimeWarning: invalid value encountered in divide
  f = msb / msw
Selected feature indices (relative to numeric features): [41, 26, 42, 114, 77, 27, 75, 140, 35, 43, 37, 11, 40, 13, 50, 175, 36, 38, 51, 125, 9, 44, 67, 145, 19, 10, 10
2, 74, 110, 85, 12, 8, 53, 7, 6, 4, 29, 45, 89, 25, 23, 62, 96, 3, 5, 34, 28, 2, 31, 1]
Feature selection complete. Output saved as data\processed_dia_dataset_graces.csv
Selected features: ['MinEStateIndex', 'FractionCSP3', 'MinPartialCharge', 'fr_ArN', 'RingCount', 'HallKierAlpha', 'PEOE_VSA8', 'fr_aniline', 'MaxAbsEStateIndex', 'MolLo
gP', 'MaxEStateIndex', 'Chi3v', 'MinAbsPartialCharge', 'Chi4v', 'NumAliphaticRings', 'fr_para_hydroxylation', 'MaxAbsPartialCharge', 'MaxPartialCharge', 'NumAromaticCar
bocycles', 'fr_Imine', 'Chi2v', 'MolMR', 'PEOE_VSA13', 'fr_benzene', 'EState_VSA4', 'Chi3n', 'VSA_EState10', 'PEOE_VSA7', 'VSA_EState9', 'SMR_VSA7', 'Chi4n', 'Chi2n', '
NumAromaticRings', 'Chi1v', 'Chi1n', 'Chi0v', 'HeavyAtomMolWt', 'MolWt', 'SlogP_VSA10', 'ExactMolWt', 'EState_VSA8', 'NumValenceElectrons', 'SlogP_VSA6', 'Chi0n', 'Chi1
', 'LabuteASA', 'HeavyAtomCount', 'Chi0', 'Kappa1', 'BertzCT']
Choose downstream model ('svm' or 'random_forest'): random_forest
```

Algorithm 3: TAB-PFN (Tabular Prior - Data Fitted Network)

Week 3-4

A brief overview:

- It is adaptation of the standard transformer encoder.
- Intended for Embedding generation, Supervised Classification and Regression analysis.
- TabPFN is promising especially for HDLSS, as it applies ICL(In-context learning) along with a large synthetic prior to help reduce overfitting and is also **feature-agnostic.**

TAB-PFN Embeddings

- TABPFN embedding extractor extracts **vector representations** from the TAB-PFN after processing the tabular data.
- It is a **192 dimensional vector** generated for the each cell in the tabular data.
- The Transformer uses **two-way attention** across features(**Row-Wise Context**) and samples(**Column-Wise Context**) to **learn** the dataset as context.
- We use this embeddings to perform the supervised downstream task

Embeddings Extraction

- Embedding extraction requires **full task context**
  - **embedder.get_embeddings(X_train, y_train, X_test, data_source='train')**
  - **embedder.get_embeddings(X_train, y_train, X_test, data_source='test')**
- We generate embeddings for both train and test splits.
- X_train, y_train, X_test , treats them as a single sequence to generate embeddings

TAB-PFN Embedding Configuration

- **tabpfn_clf** : The trained TABPFN model used to extract embeddings.
- **n_fold** : Number of task-folds used to compute embeddings. Higher = more stable embeddings
- data_source : Choose `"train"` or `"test"` embeddings after the transformer processes the full task

Project Timeline

Completed Tasks   Planned Tasks

	Week 1	Week 2	Week 3	Week 4	Week 5	Week 6	Week 7	Week 8	Week 9	Week 10	Week 11	Week 12
Literature Review	Chethan, Bhavesh Sourav & Sajith											
Dataset Selection		Sajith										
Data Preprocessing			Chethan Bhavesh									
Simulator(dataset variant generation)			Sajith Sourav									
Embedding Framwork Setup (Model 1)					Sourav Bhavesh							
Embedding Framwork Setup (PTM)(Model 2,3)					Sajith Chethan							
Embedding Generation and Analysis							Bhavesh Sourav					
Downstream Framework Plan							Sajith Chethan					
Downstream Task Execution & Analysis									Chethan Sajith			
Performance Criteria Finalization & Reporting									Bhavesh Sourav			
Pipeline Creation and Test Runs											Chethan Sajith, Sourav, Bhavesh	
Result Analysis and Final Documentation												Chethan Sajith, Sourav, Bhavesh

Thank you!

Questions?