Genomics of Drug Sensitivity in Cancer (GDSC)

Group 7

曹育瑄 110502507

王恩柔 109607502

李姵儀 110502504

劉彥汝 113527006

GitHub

Genomics of Drug Sensitivity in Cancer (GDSC)

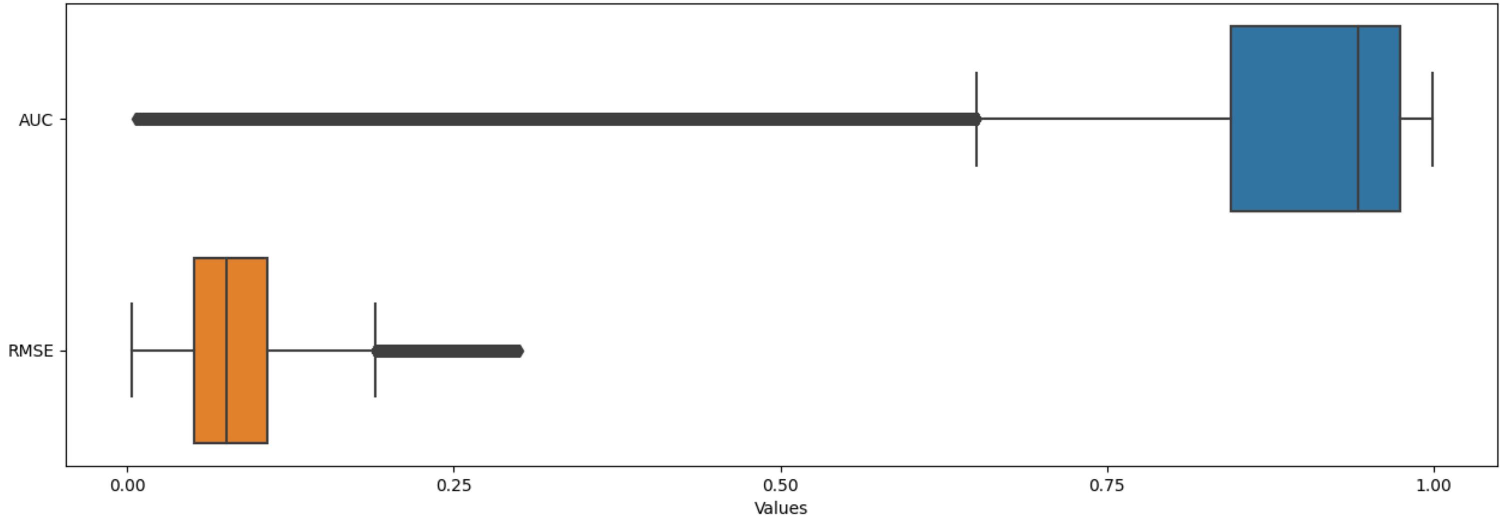
- This dataset is from the GDSC project, which is a collaboration between the UK and the USA.
- The project involves large-scale screening of human cancer cell lines with a wide range of anti-cancer drugs. In the experiments, **cell viability** was checked using the **CellTiter-Glo** test after 72 hours of drug treatment.
- The GDSC dataset provides a comprehensive resource for studying the response of various **cancer cell lines** to a wide range of **anti-cancer drugs**.

- **GDSC2-dataset.csv**
  Contains **drug sensitivity data** for various drugs tested against cancer cell lines
- **Cell_Lines_Details.xlsx**
  Detailed information about the **cancer cell lines**
- **Cell_Lines_Details.xlsx**
  Information about the **drugs** used in the screening
- **GDSC_DATASET.csv**
  It's a merged file combining key information from the above three files

- The target variable in this dataset is **LN_IC50**. This variable represents the concentration of a drug that inhibits cell viability by 50.
- Lower LN_IC50 values indicate higher drug sensitivity

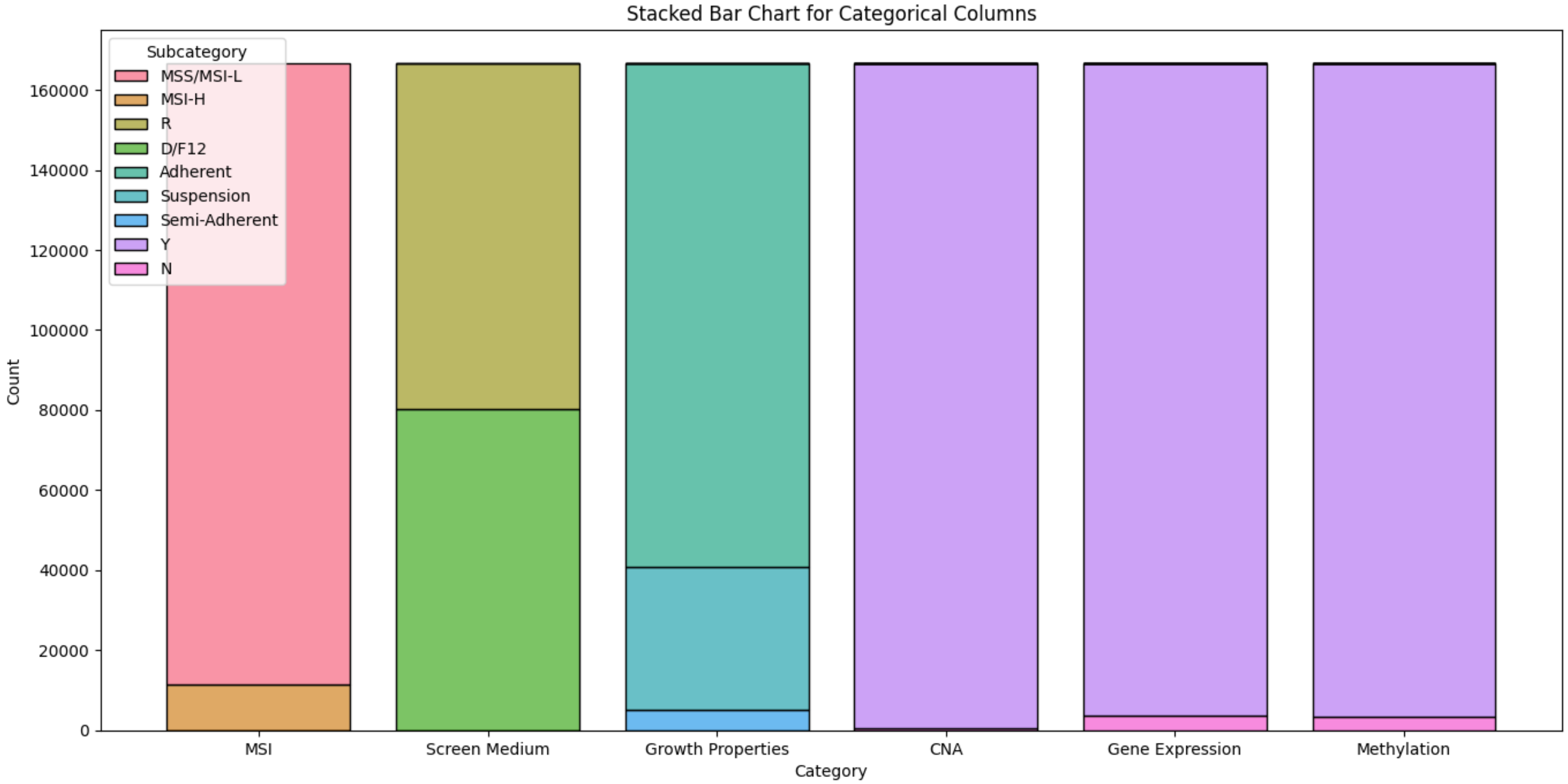
Dataset Exploratory data analysis (EDA)

- **Numerical Columns**
  - **Z_SCORE**: Standardized score of the drug response, allowing comparison across different drugs and cell lines.
  - **AUC**: Area Under the Curve, a measure of drug effectiveness.
  - **RMSE**: Root Mean Square Error, indicating the fit quality of the dose-response curve.



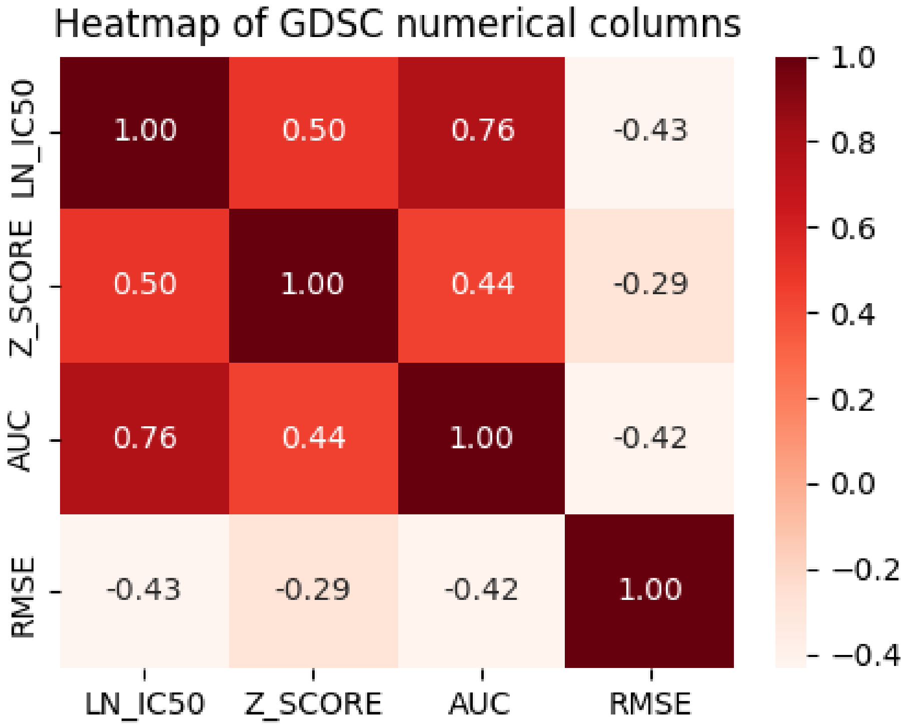
Dataset Exploratory data analysis (EDA)

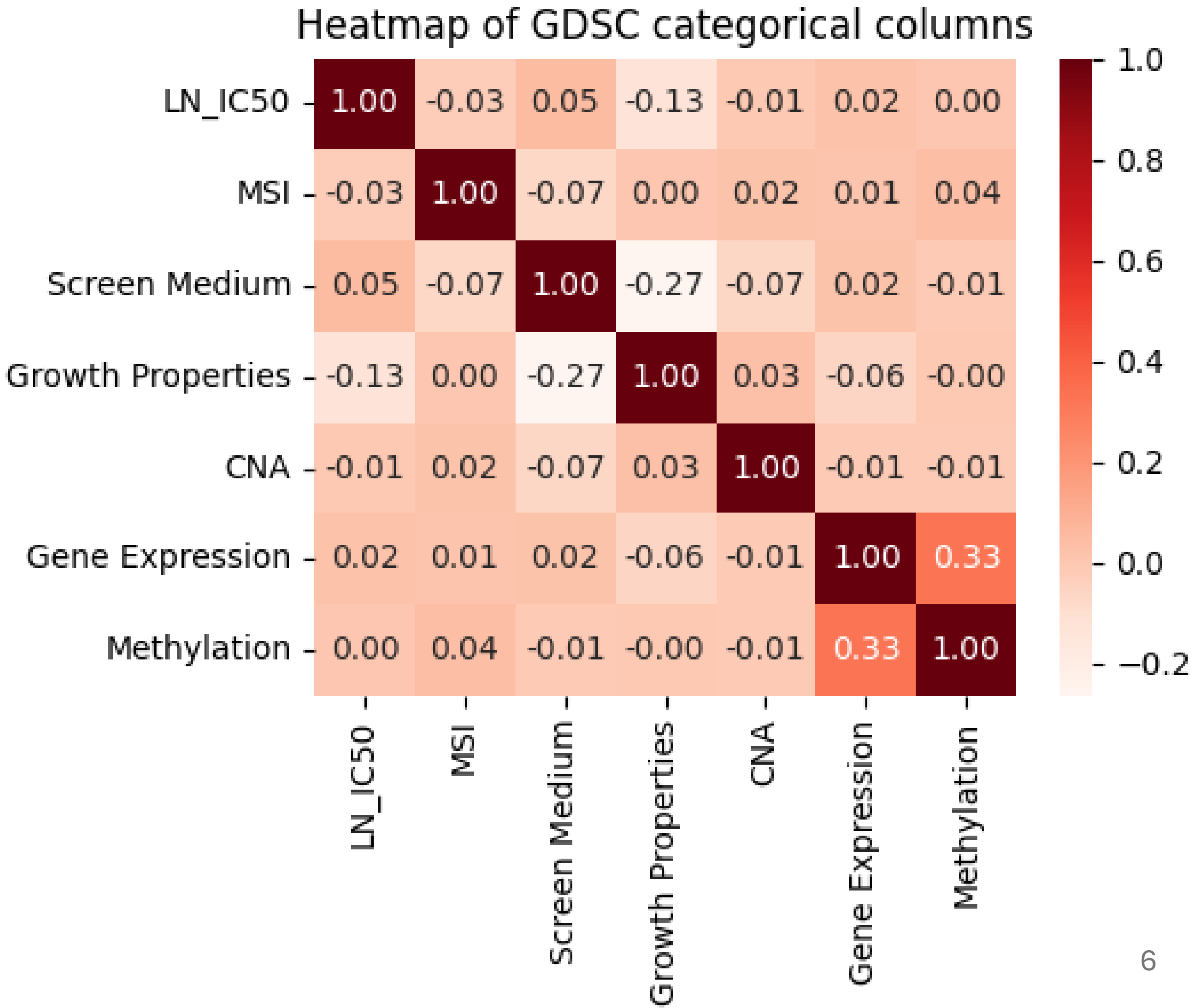
- **Categorical Columns**



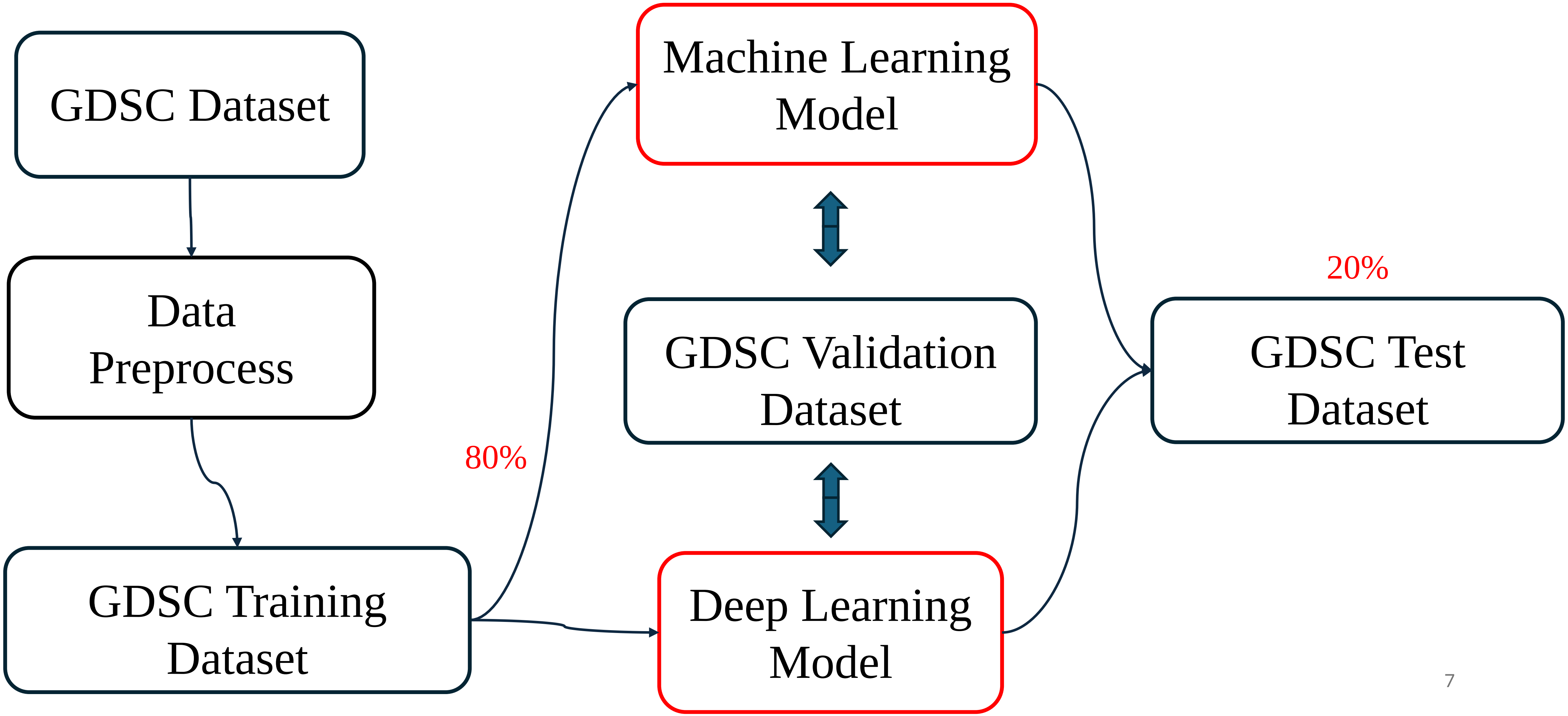
Dataset Exploratory data analysis (EDA)

- **Heatmap**

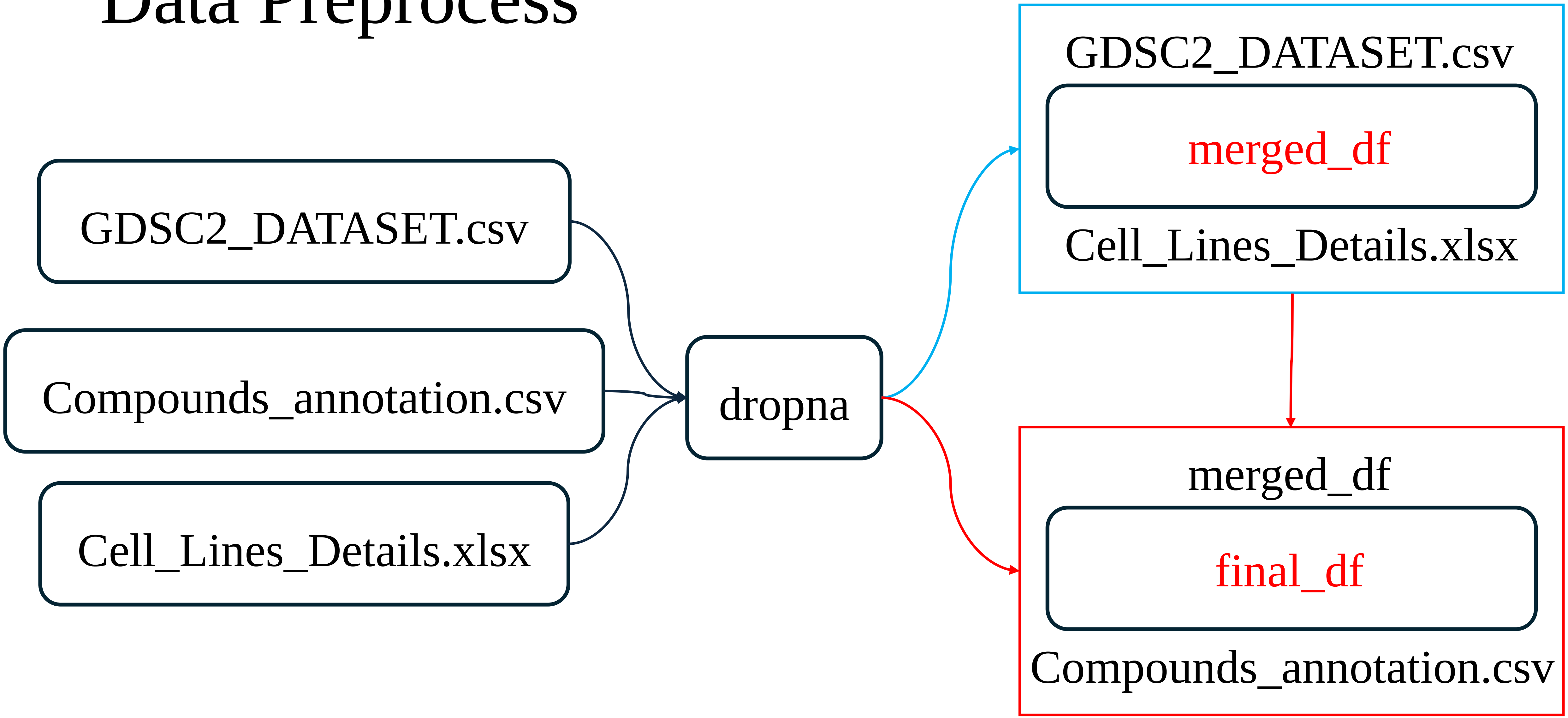




Our training pipeline



Data Preprocess



Machine Learning Model – Methodology Overview

Linear models

- Lasso Regression
- ElasticNet Regression
- Linear Regression
- Ridge Regression

Decision Tree and Ensemble Learning

- Decision Tree
- Random Forest
- HistGradientBoosting Regression
- XGB Regressor
- LightGBM

Gradient Boosting Regression

Instance-Based Learning

- K-Nearest Neighbtors Regression

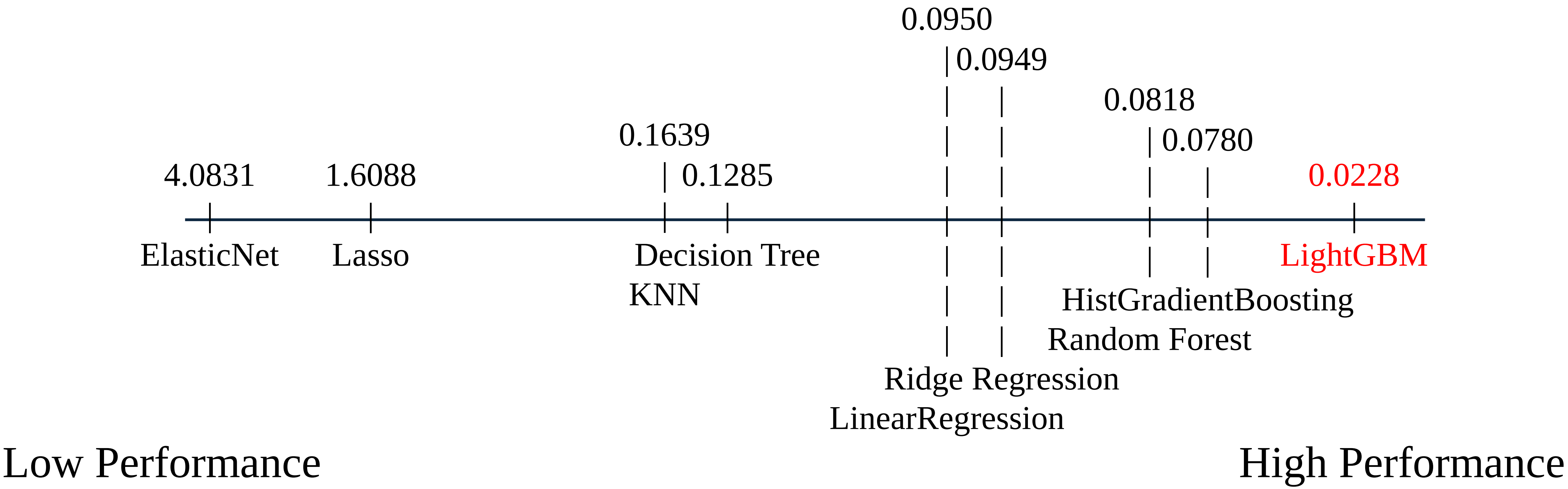
Machine Learning Model - Hyperparameter Tuning and Optimization

- Use GridSearchCV to find the best hyperparameter
- For example :

```
def tune_hyperparameters(self, X, y):
    # Define hyperparameter grid
    param_grid = {
        'n_estimators': [100, 200, 300],
        'max_depth': [None, 10, 20, 30],
        'min_samples_split': [2, 5, 10],
        'min_samples_leaf': [1, 2, 4]
    }

    # Use GridSearchCV to find the best hyperparameters
    gs = GridSearchCV(self.model, param_grid, refit=True, cv=3, scoring='neg_mean_squared_error')
    gs.fit(X, y)

    # Print best parameters and score
    print("Best Parameters:", gs.best_params_)
    print("Best Score:", gs.best_score_)

    # Return the best model
    return gs.best_estimator_, gs.best_params_, gs.best_score_
```

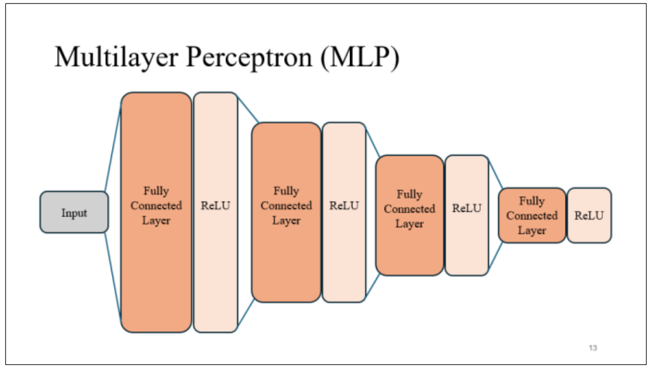
Machine Learning Model - Evaluation

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$

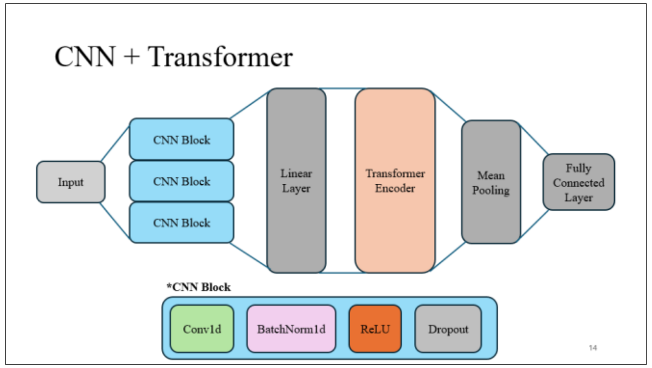


Deep Learning Model

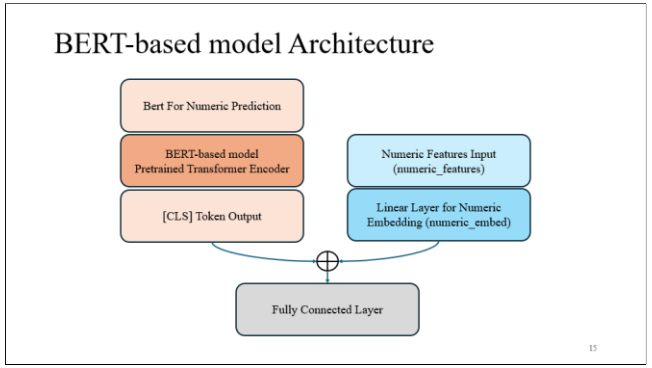
- Multilayer Perceptron (MLP)



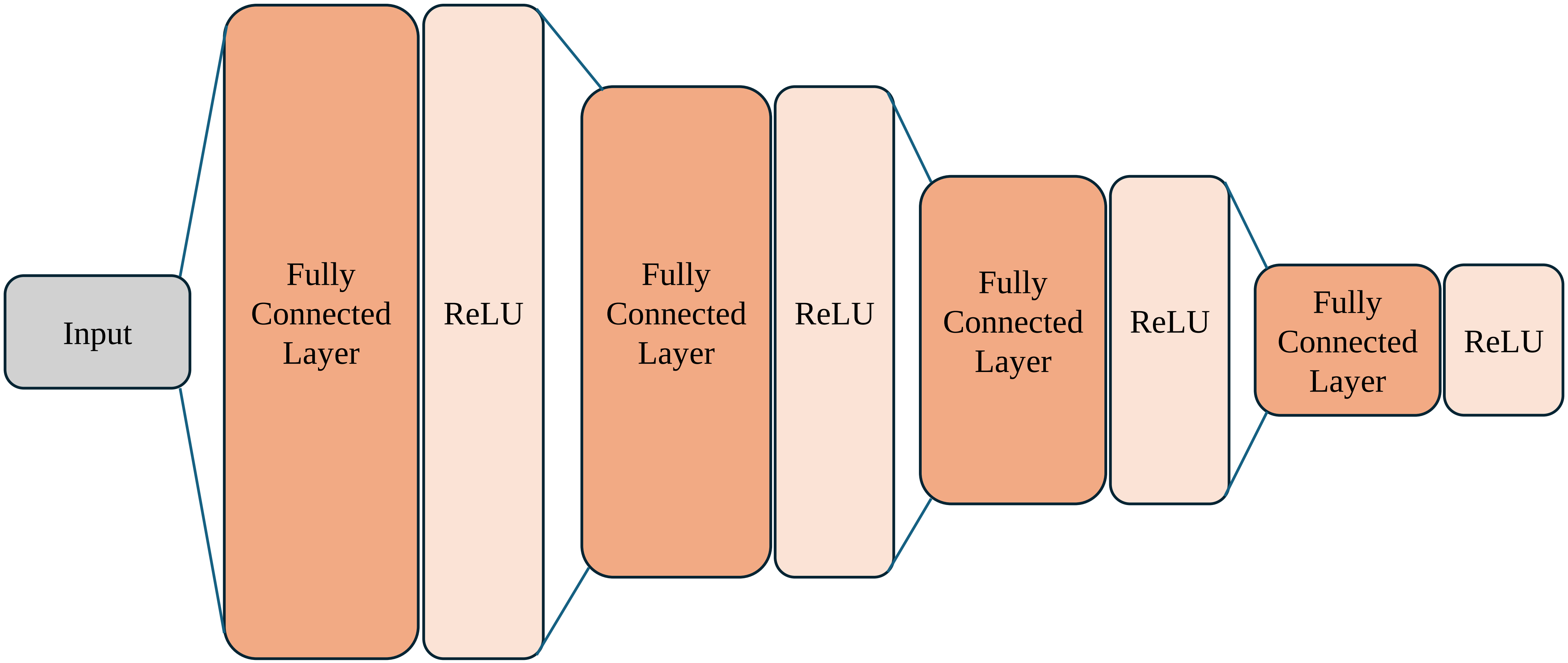
- CNN + Transformer



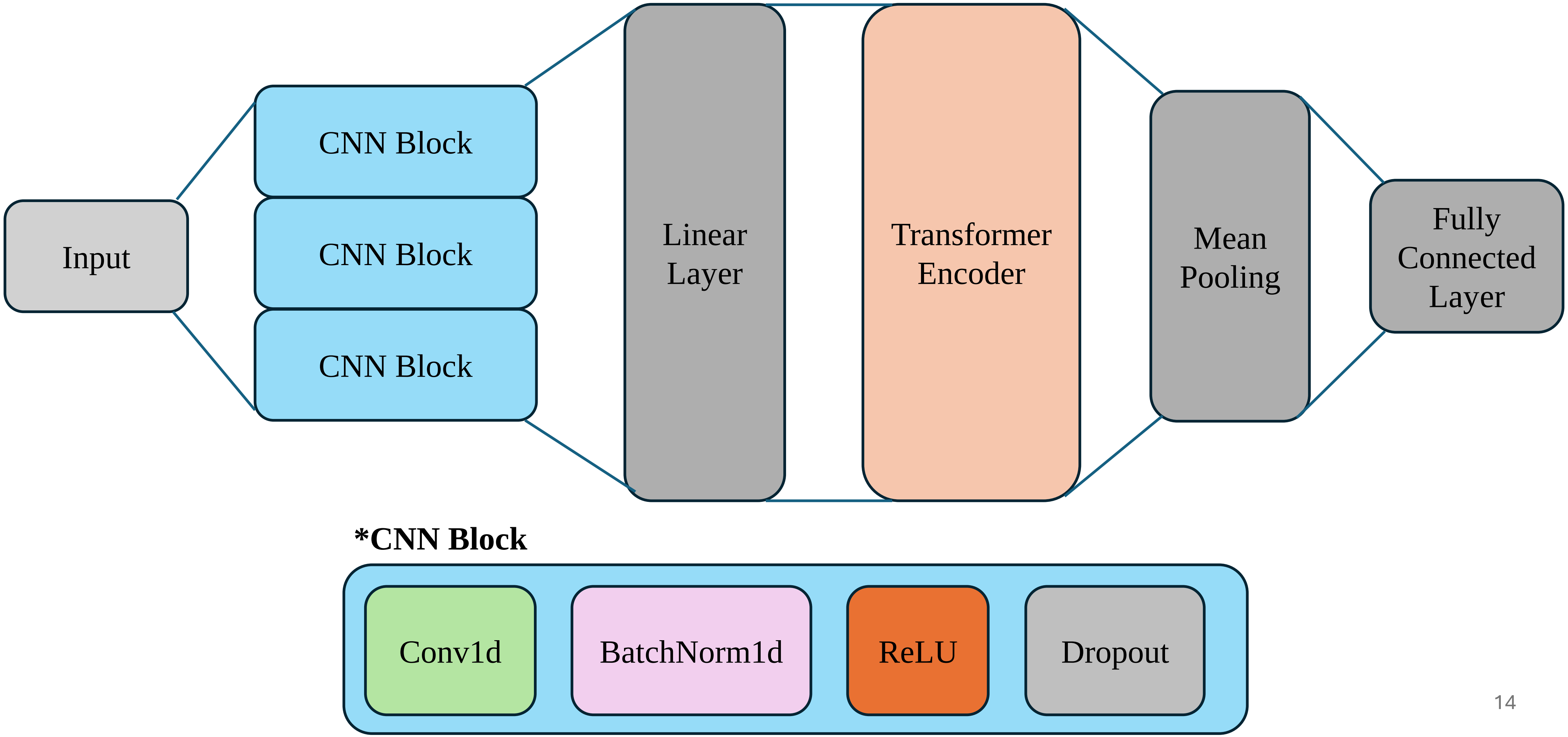
- BERT (bert-base-uncased)
- RoBERTa (roberta-base)
- DeBERTa (microsoft/deberta-v3-base) (microsoft/deberta-v3-large)



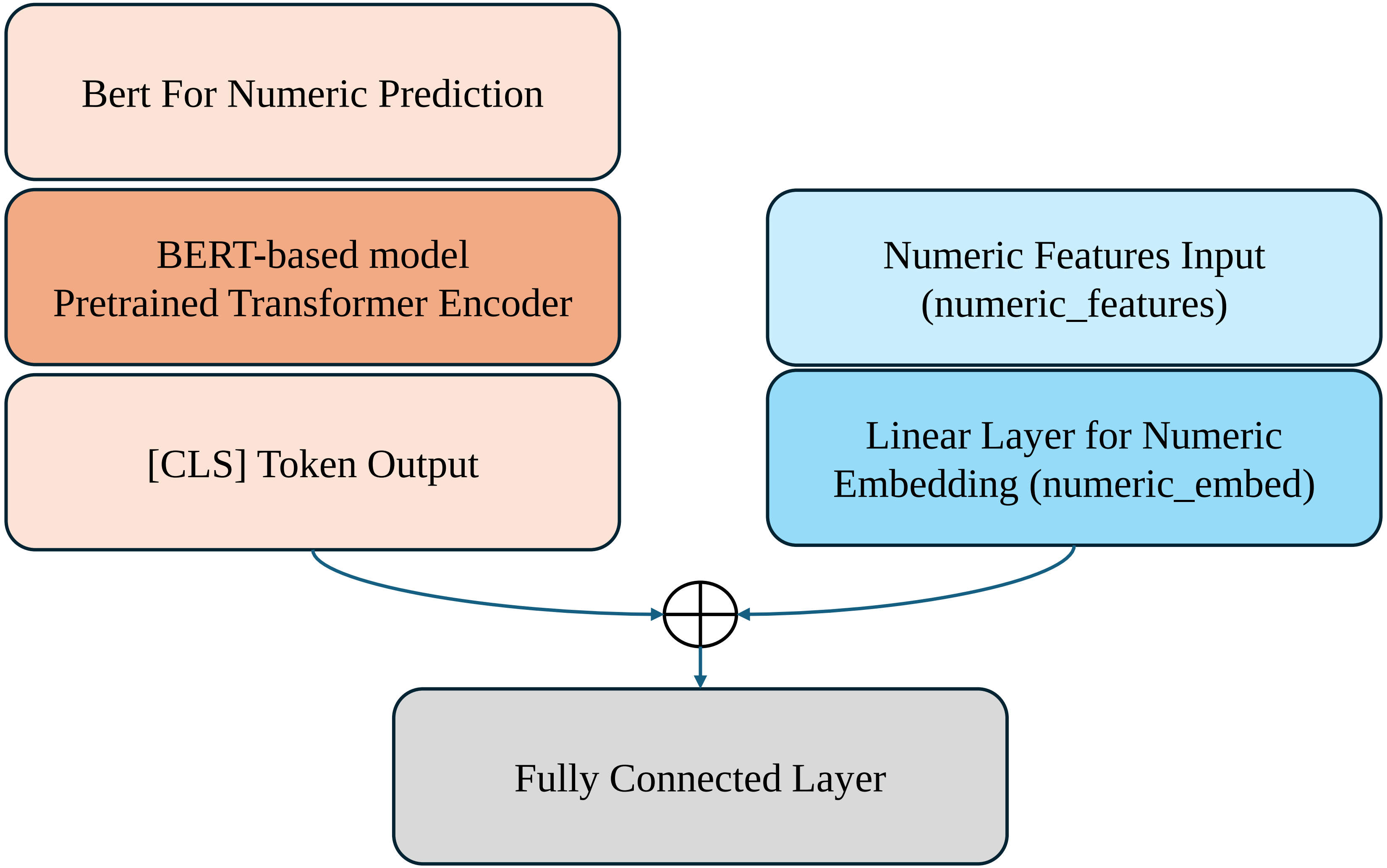
Multilayer Perceptron (MLP)



CNN + Transformer



BERT-based model Architecture



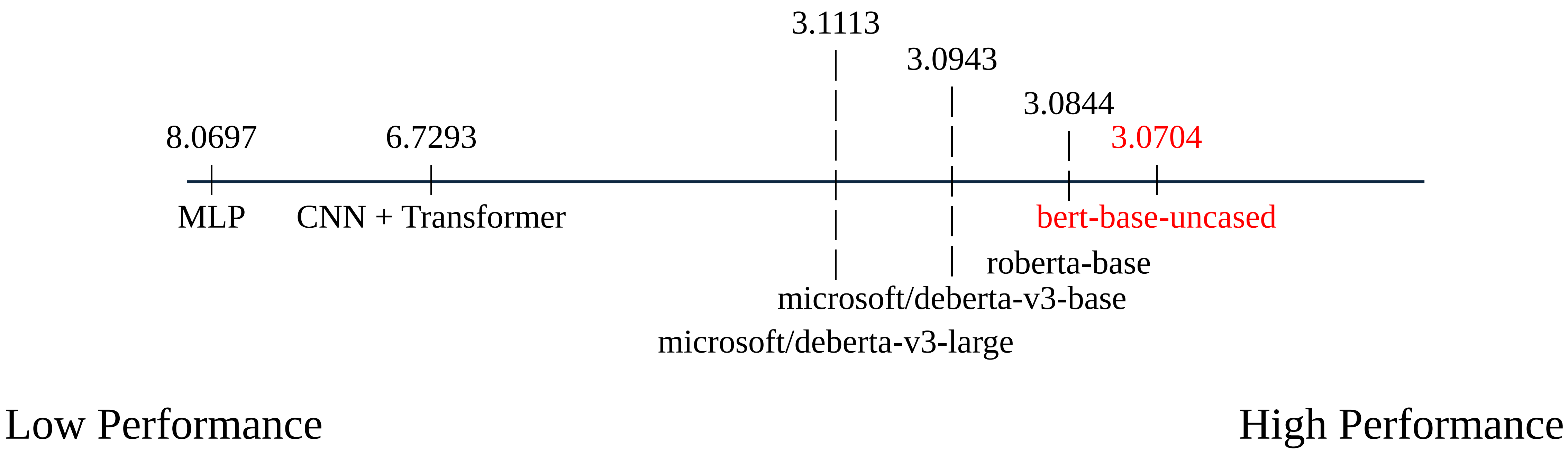
Deep Learning Model

- Use adaptive learning rate to approach the higher performance

```python
def adjust_learning_rate(optimizer, current_loss, previous_loss, factor=0.9, min_lr=1e-6):
    """
    Adjusts the learning rate based on the Mean Squared Error (MSE).
    If the current loss is greater than the previous loss, decrease the learning rate.
    """
    if current_loss > previous_loss:
        for param_group in optimizer.param_groups:
            new_lr = max(param_group['lr'] * factor, min_lr)
            param_group['lr'] = new_lr
            print(f"Learning rate decreased to {new_lr:.6e}")
```

Deep Learning Model - Evaluation

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2$$



Our Contribution

- We implemented a variety of machine learning algorithms, such as LightGBM, RandomForest...
- We developed at least three deep learning models, such as BERT, RoBERTa, DeBERTa, CNN + Transformer.
- We achieved pretty results on this dataset which is based on Mean Squared Error (MSE) metric, demonstrating the effectiveness and practical potential of our approach.

Low Performance High Performance

Conclusion and Future Work

- Machine Learning sometimes can beat Deep Learning.
- Good parameters take time.
- Each task has its own suitable model.
- Conducting feature selection to reduce the time of model training and the data noise.
- Maybe the dataset is too small, which is why BERT performed poorly. We can explore how to perform data augmentation in the future.

THANKS

Q&A