Datathon | KUET Bitfest 2025

Team Name: **Sirius**

Presented By:

Ashik Shahriar, EEE, BUET (Team Leader)

Abdullah Muhammed Amimul Ehsan, CSE, BUET

Anik Saha, CSE, BUET

Jaber Ahmed Deeder, CSE, BUET

Problem Definition

The problem asks for designing a model to predict the matched score between a candidate's profile and a job's requirements.

Candidate:

- skills
- degree names
- major field of studies
- positions
- job experience dates
- related skills in jobs

Job:

- responsibilities
- job position name
- educational requirement
- experience requirement
- skills required

Our Approach

We explored the dataset to figure columns that are necessary for model training

We filled empty columns with default values

Generated features of the important columns

Trained the model

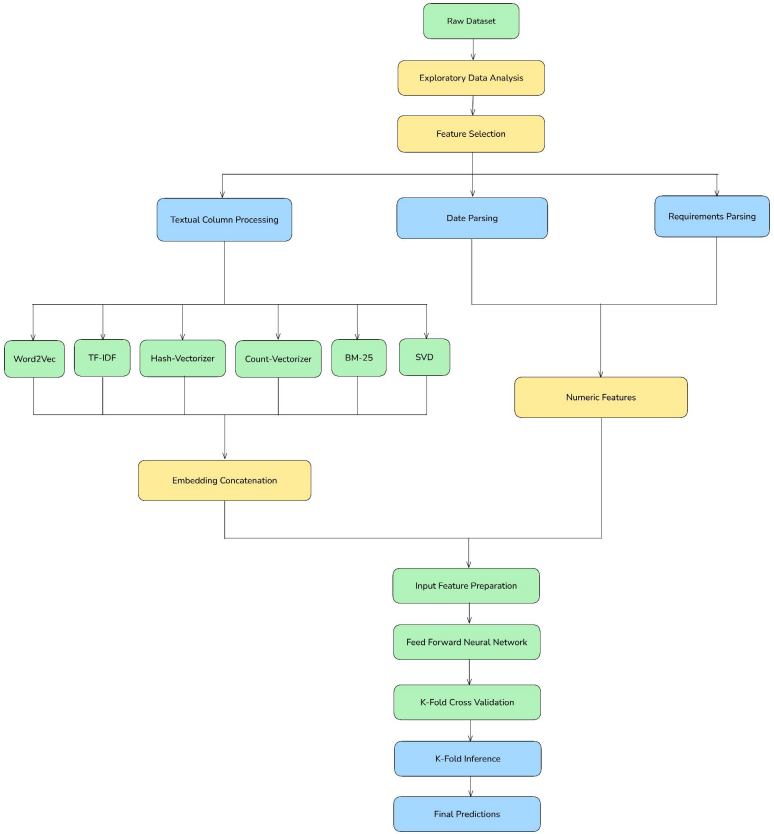
Our Solution Overview

We address the major challenges in the following ways:

- Textual features are embedded using models fit on only the training dataset
- Numeric features are extracted by parsing and concatenated to the embeddings
- A Feedforward Neural Network is trained with K-fold cross validation
- Inference is carried across over the test dataset with K-fold ensemble inference

We run our solutions using the GPU-T4x2 in Kaggle platform.

Our Solution Overview



Data Analysis

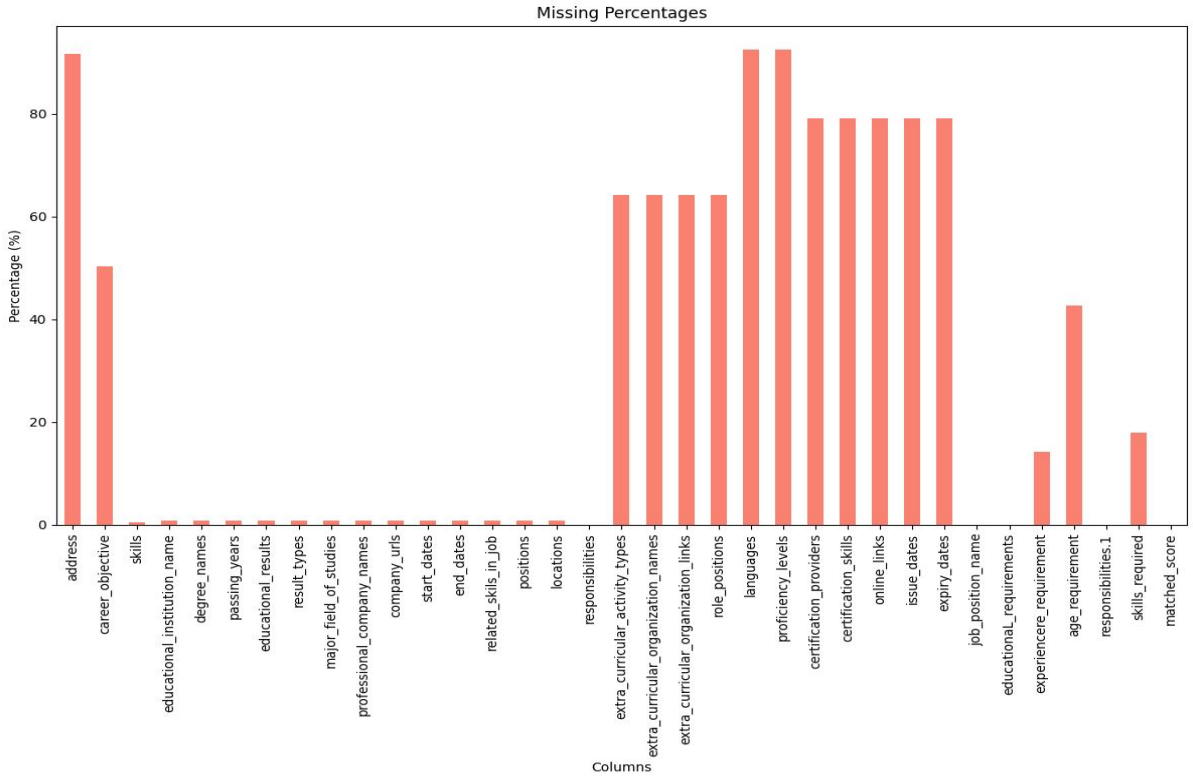
- Missing Values
- Unimportant Columns
- Diverse Textual Data
- Date and Numeric Columns in Textual Form

Missing Values

Inspecting the train and test data, we find that several columns have significant amount of missing values. These include,

- address
- languages
- proficiency_levels
- certification_providers
- extra_curricular_activity_types
- role_positions
- proficiency_levels, and many more

Missing Values



Unimportant Columns

We also notice that some fields clearly do not contribute at all to the prediction problem, For instance,

- address
- company_urls
- online_links

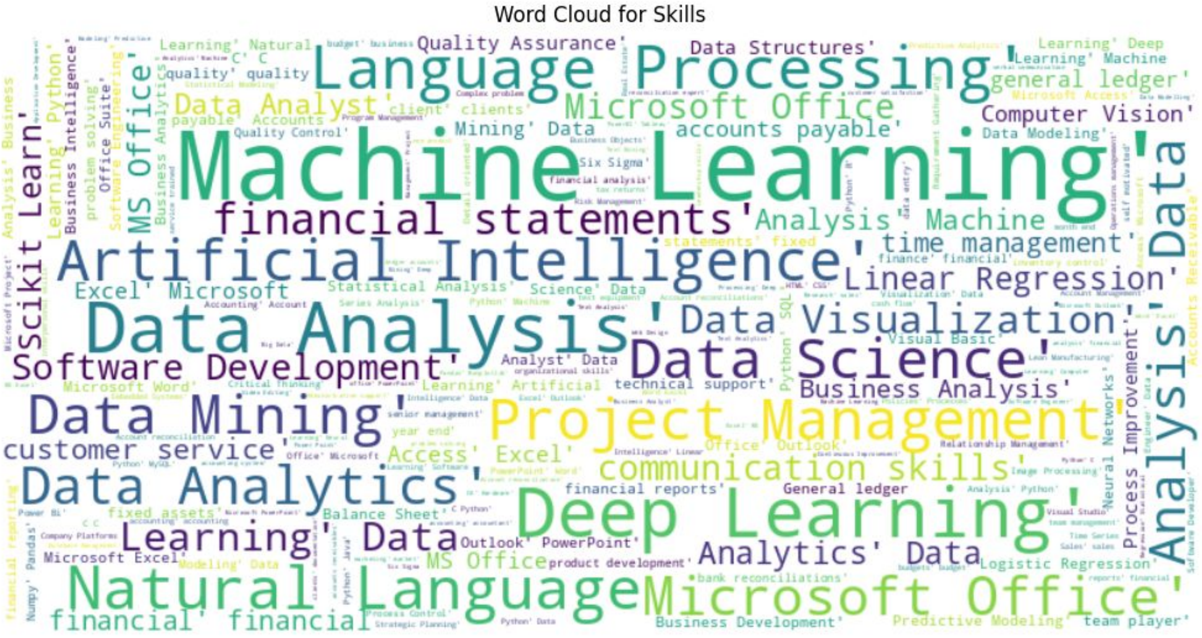
Diverse Textual Data

We notice that a lot of the columns contain textual data which have a rather high number of unique values. For instance,

- skills
- responsibilities
- related_skills_in_job
- degree_names

One exception, however, is ‘educationaL_requirements’ which contains only 20 unique values.

Word Cloud for ‘skills’



Date and Numeric Data

We note that, columns like ‘start_dates’, ‘end_dates’ actually present date values, but are given in text format, like ‘18/2022’ or ‘Current’ etc

In addition, some of the columns like ‘experiencere_requirement’, ‘age_requirement’ etc actually present minimum and maximum allowed values, but are given in text format, like ‘2 to 5 years’

Data Manipulation

- Feature Selection
- Textual Data Processing
- Date and Numeric Data Extraction
- Textual Data Embedding

Feature Selection

Based on our observations, we decide to **ignore** some features completely. These are,

- address
- extra_curricular_activity_types
- extra_curricular_organization_names
- extra_curricular_organization_links
- role_positions
- languages
- proficiency_levels
- certification_providers
- certification_skills
- online_links
- issue_dates
- expiry_dates

Textual Data Processing

The textual columns present list formatted strings in some cases. We process them to obtain comma-separated plain-text formatted strings.

For instance, ‘[‘Python’,’Java’]’ is converted to “Python, Java”

Date and Numeric Data Extraction

From the columns ‘start_dates’, ‘end_date’, we extract exact dates in (month, year) format using a Python based parser function.

From the columns, ‘experiencere_requirement’, ‘age_requirement’, we extract minimum and maximum values using Python-based parser.

Textual Data Embedding

We embed the textual columns data using a grouped hybrid embedder. A concatenation of the vectors obtained from each of these embeddings is used.

- Word2Vec
- TF-IDF
- Hashing Vectorizer
- Count Vectorizer
- Truncated SVD (Singular Value Decomposition)
- BM25
- Mean Pooling
- Max Pooling

Textual Data Embedding

As the embedding scheme, we first tried doing column-wise embeddings, by fitting embedders on a vocabulary built by merging the texts in individual columns.

Later, however, we switched to Grouped Hybrid Embedding, with the observation that some columns like ‘skills’, ‘responsibilities’, ‘related_skills_in_job’ share similar vocabulary.

So we define groups based on this and fit embedders based on them.

Text Embedding Comparison

Embedding Strategy	MSE (Public)
Word2Vec	0.00762
TF-IDF	0.00794
SVD	0.00798
Hash	0.00834
BM25	0.00789
Hybrid Embedding	0.00753
Grouped Hybrid Embedding	**0.00724**

Training Methodology

- Input Feature Preparation
- Model Exploration and Selection
- Hyperparameter Tuning
- Model Architecture
- Training

Input Feature Preparation

We concatenated all the textual embeddings obtained from the following columns,

- educationaL_requirements
- job_position_name
- responsibilities
- skills_required
- degree_names
- major_field_of_studies
- positions
- related_skils_in_job
- skills
- career_objective
- professional_company_names
- experiencere_requirement

Input Feature Preparation

- Textual embeddings are first generated for the aforementioned columns
- Numeric features are extracted from the columns ‘experiencere_requirement’ and ‘age_requirement’
- Then we concatenate all the obtained features
- For vector sizes, different values were explored ranging from 50 to 224

Model Exploration and Selection

We tried the following models,

ML Models:

- Random Forest Regressor
- XGBoost regressor
- Catboost Regressor
- LightGBM regressor

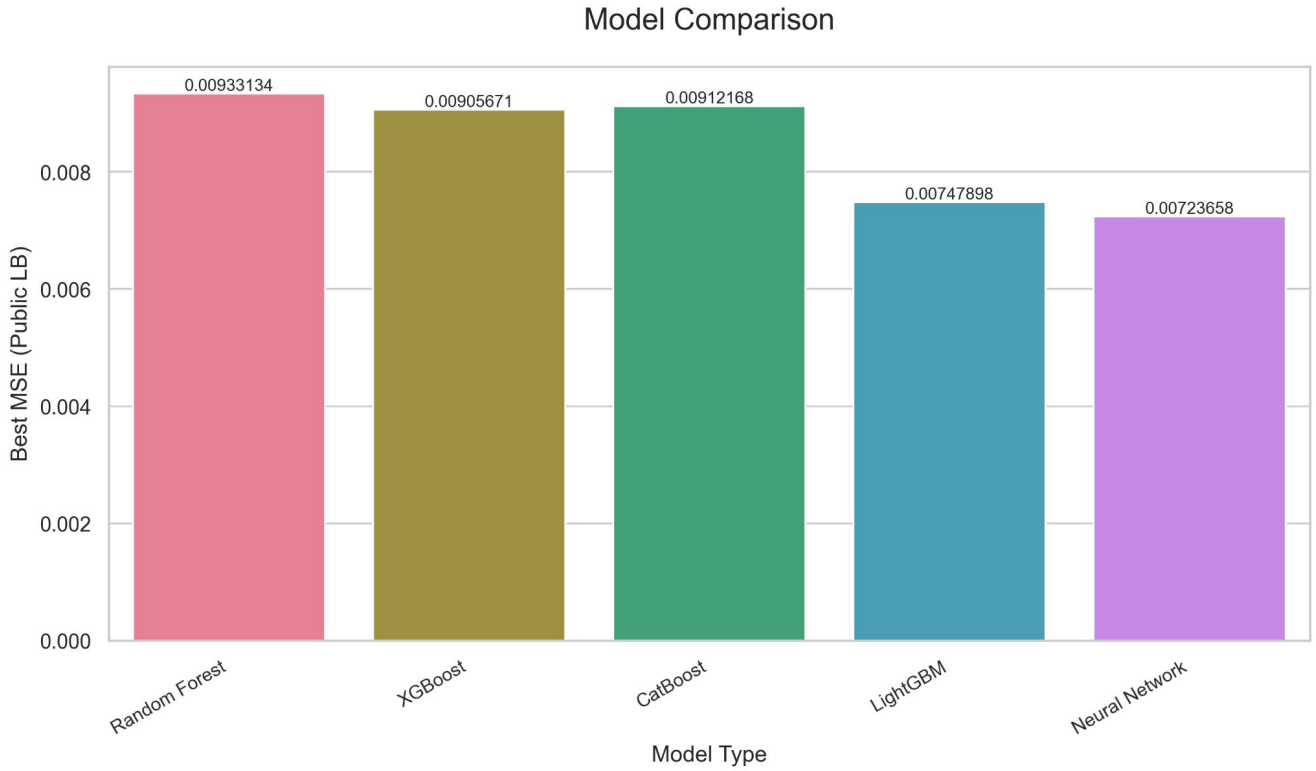
DL Models:

- Feed Forward Deep Neural Network

Model Exploration and Selection

- Among the ML Models, LightGBM performed best
- Finally, among all the models, Feed Forward Neural Network. Therefore, we go on with Feed Forward Neural Network

Model Exploration and Selection



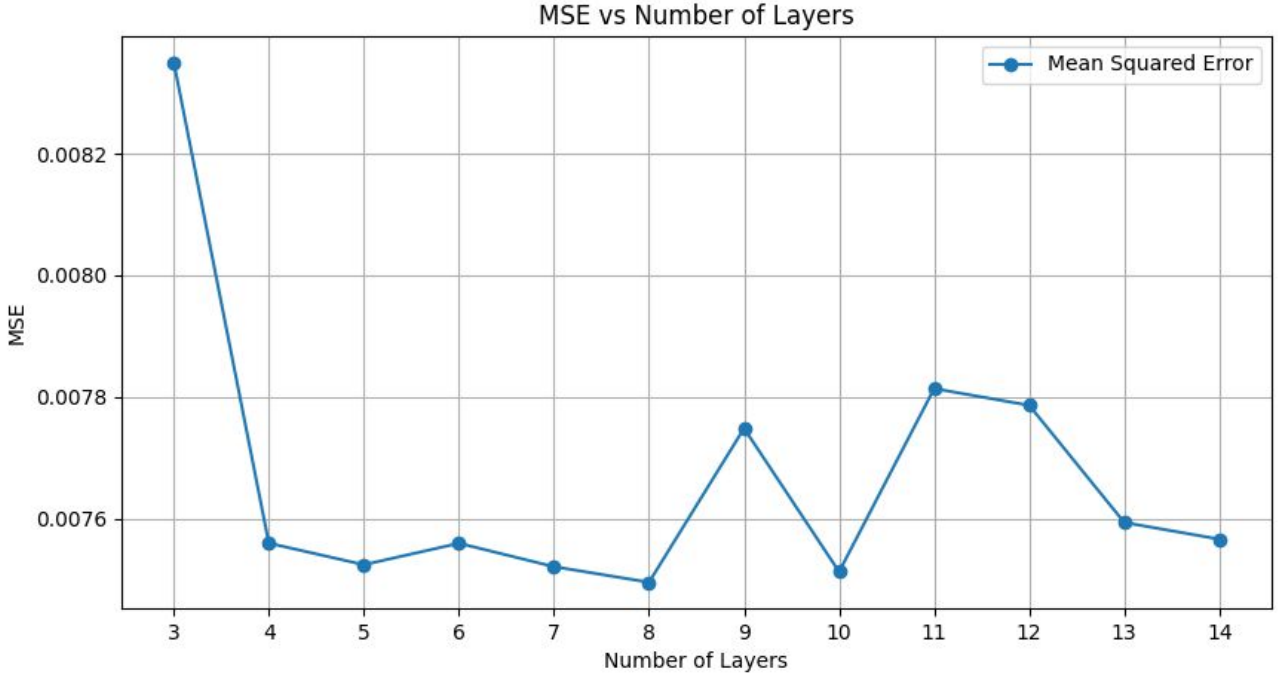
Hyperparameter Tuning

We search for different hyperparameters for the Feed Forward Neural Network. Alterations of the following properties are explored,

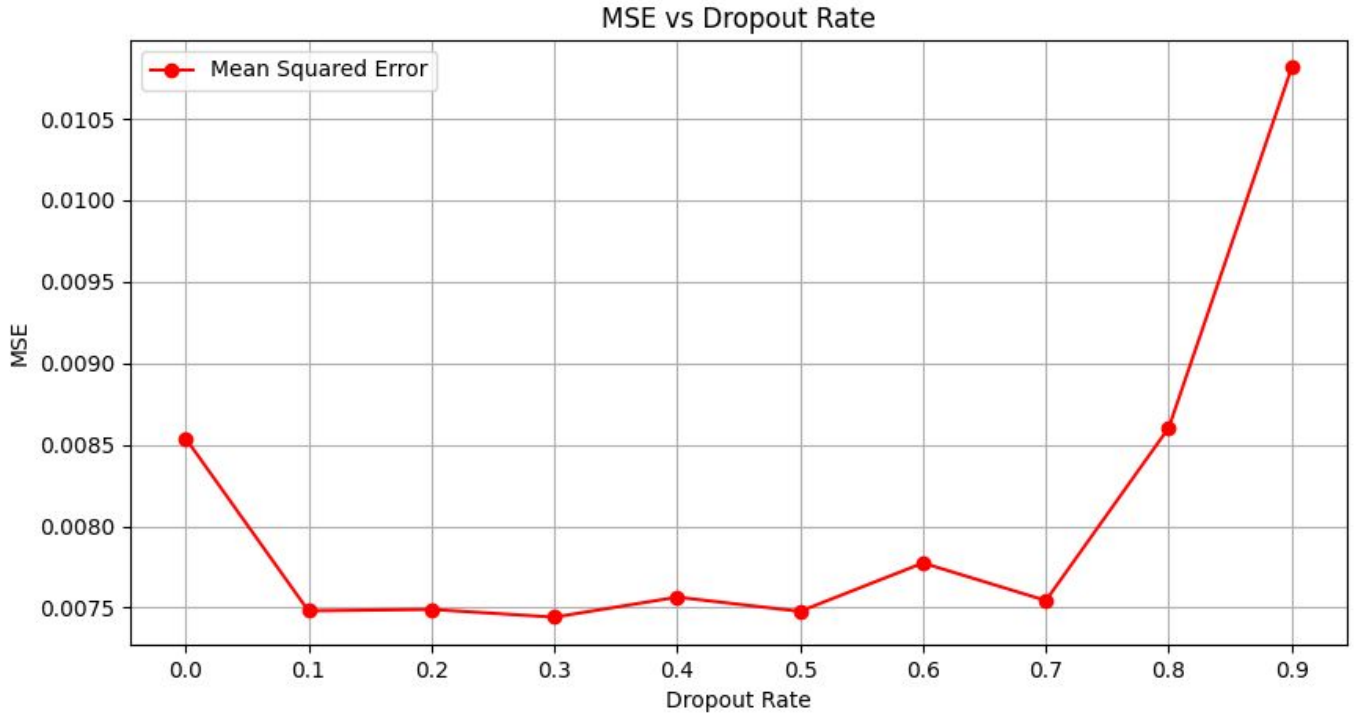
- number of layers
- dropout rate
- number of epochs
- learning rate

We next present the variations in Mean Squared Error we encountered while varying the ‘number_of_layers’ and ‘dropout_rate’ and keeping other parameters fixed.

Hyperparameter Tuning



Hyperparameter Tuning



Training Methodology - Model Architecture

In our final Feed Forward Neural Network, the notable properties are,

- 7 layers
- Batch Normalization
- Dropout rate = 0.5
- Residual Connections
- Activation Function: GELU
- Learning rate = 0.001

Training Methodology - The Main Loop

We use pytorch framework to write the training loop. Notable properties of the training are:

- **Early Stopping:** We keep saving the model every 100 iterations and keep track of the validation losses. Then we load the state where the loss dropped to the lowest. This helps us battle overfitting.
- **K-Fold Cross Validation:** We divide the train data into folds with and train on them using a remaining part as validation set. For the value of K, we try values from 5 to 10.

Inference on Test Set

- We infer on test set with the same K-Fold Ensembled Inference scheme. That means, for each of the trained models on folds, we predict for the whole test set
- Then for each of the test data points, we mean over all those inferred values
- Finally we append all those predictions on the test set
- Note that, this folded inference seemed to improve the performance over training a single model over the whole train set

Ablation Studies

Component	MSE With	MSE Without	Change
Batch Normalization	0.00723658	0.00733190	1.32%
Residual Connections	0.00723658	0.01291629	78.48%
Dropout	0.00723658	0.00916782	26.69%
K-Fold Ensembled Inference	0.00723658	0.00803023	10.97%
Early Stopping	0.00723658	0.00747334	3.27%

Our Final Scores

Public Score	Private Score
0.00723658	0.00734709

Thank You