# Transformer Based Modeling for Nutrient Content Prediction Using Ensemble Regression

MSc Research Project
MSc in Data Analytics

Anjali Yenugula
Student ID: x23266431

School of Computing
National College of Ireland

Supervisor: Eric Gymafi

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

| | |
|---|---|
| **Student Name:** | Anjali Yenugula |
| **Student ID:** | x23266431 |
| **Programme:** | MSCDAD_B **Year:** 2025 |
| **Module:** | MSc Research Practicum |
| **Lecturer:** | Eric Gyamfi |
| **Submission Due Date:** | 11-08-2025 |
| **Project Title:** | Transformer Based Modeling for Nutrient Content Prediction Using Ensemble Regression |
| **Word Count:** | 561 **Page Count:** 6 |

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Anjali

**Date:** 11-08-2025

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Transformer Based Modeling for Nutrient Content Prediction Using Ensemble Regression Approaches

Anjali Yenugula
Student ID: x23266431

# 1. Introduction

This configuration manual supports the implementation of a hybrid machine learning systemdeveloped for predicting key nutritional values including energy, protein, and fat from structured food composition data.

The manual intends to help users set up and operate the entire pipeline clearly and step bys tep. It includes:

- Dataset preparation and preprocessing
- Model setting and parameter tuning
- Execution of training and evaluation workflows
- Output formats and interpretation of the results

# 2. System Infrastructure Prerequisites

**Platform Compatibility**

- **Primary Support**: Windows 10 or 11, ideally, with maximum compatibility of libraries and tools.
- **Alternative**: Ubuntu 20.04 LTS or above for the ones who prefer open source/server side configurations.

**Processing Core Framework**

- **Baselevel Capability**: Multi core CPU architecture (e.g., Intel Core i5 / AMD Ryzen 5 or newer).
- **Memory Throughput**: Quad-core and above for speeding up model training and feature-processing

**System Memory Allocation**

- **Minimum Functional Capacity**: 16 GB RAM for running most models and testing
- **Enhanced Operational Performance**: 2 GB or higher for handling large set of data and for parallel testing

**Storage Buffering**

- **Available Disk Space**: 15 to 20 GB free storage to accommodate:

- Raw and preprocessed datasets
- Model artifacts and embeddings
- Visualization outputs and evaluation logs
- Required package libraries and dependencies

# 3. Software Environment Configuration

**Programming Language**

- **Python version : 3.11.7**

**Core Analytical Toolkits**

These foundational libraries support data ingestion, statistical analysis, and visualization:

- **pandas** – for handling structured tabular data and transformations
- **numpy** – for performing the array operations and mathematical functions
- **matplotlib** / **seaborn** – for creating visual insights and feature correlations
- **plotly** *(optional)* – for visualizations suitable for dashboards which are dynamic
- **scikit-learn** – for preprocessing routines, baseline regressors, evaluation metrics, and model validation utilities

**Neural Processing Libraries**

For implementing attention based embedding and ensemble pipelines:

- **PyTorch**– core framework for building, training the TabTransformer architecture

**Training and Optimization Modules**

Modular components that assist with efficient data loading, scaling, and model tuning:

- **DataLoader**, **TensorDataset** – to manage data batching for model training
- **StandardScaler** – for feature normalization before model input

**Development Workspaces**

The system can be implemented and experimented with using any of the following:

- **Jupyter Notebook** – intuitive for testing, visual tracking, and iterative design

**Environment Isolation Tools**

To ensure reproducibility and reduce dependency conflicts:

- **conda** – preferred for managing isolated environments with deep learning stack compatibility

# 4. Dataset Details

Dataset link : https://www.kaggle.com/datasets/thedevastator/the-nutritional-content-of-food-a-comprehensive

- Structured food nutrient dataset (ABBREV.csv) with 50+ attributes per item
- Column renaming, typo correction, and duplicate removal for consistency
- Features with >40% missing values dropped; rest imputed (median/mode)
- Redundant and non-informative columns excluded
- Derived food categories, visualizations, and cleaned data used for modeling and embeddings

# 5. Model Configuration

- Support Vector Regressor (SVR),
- K-Nearest Neighbors Regressor (KNN)
- Random Forest Regressor

```
classical_models = {
    'SVR': MultiOutputRegressor(SVR(kernel='rbf')),
    'KNN': MultiOutputRegressor(KNeighborsRegressor(n_neighbors=1)),
    'Random Forest': MultiOutputRegressor(RandomForestRegressor(n_estimators=5, max_depth=4, random_state=42))

}

baseline_metrics_df = pd.DataFrame()


for model_name, model in classical_models.items():
    print(f" Training and evaluating {model_name}...")
    model.fit(X_train_le, y_train)
    y_pred = model.predict(X_test_le)

    for i, target in enumerate(y_train.columns):
        y_true = y_test[target].values
        y_pred_target = y_pred[:, i]

        mse = mean_squared_error(y_true, y_pred_target)
        rmse = np.sqrt(mse)
        mae = mean_absolute_error(y_true, y_pred_target)
        r2 = r2_score(y_true, y_pred_target)


        new_row = {
            'Model': model_name,
            'Target': target,
            'RMSE': rmse,
            'MSE': mse,
            'MAE': mae,
            'R2': r2
        }
        baseline_metrics_df = pd.concat([baseline_metrics_df, pd.DataFrame([new_row])], ignore_index=True)
```

**TabTransformer + LightGBM (Hybrid Model)**

```python
def evaluate_per_target(model_dict, model_name, X_test, y_test, target_cols):
    print("=" * 60)
    print(f" Evaluation Results for Model: {model_name}")
    print("=" * 60)

    results = []

    for target in target_cols:
        preds = model_dict[target].predict(X_test)
        y_true = y_test[target].values

        mse = mean_squared_error(y_true, preds)
        rmse = np.sqrt(mse)
        mae = mean_absolute_error(y_true, preds)
        r2 = r2_score(y_true, preds)


        results.append({
            'Model': model_name,
            'Target': target,
            'RMSE': rmse,
            'MSE': mse,
            'MAE': mae,
            'R2': r2
        })


        table = [
            ['RMSE', f"{rmse:.4f}"],
            ['MSE', f"{mse:.4f}"],
            ['MAE', f"{mae:.4f}"],
            ['R2 Score', f"{r2:.4f}"]
        ]
        print(f"\n Metrics for Target: {target}")
        print(tabulate(table, headers=['Metric', 'Value'], tablefmt='fancy_grid'))
```

### TabTransformer + Random Forest (Hybrid Model)

```python
import lightgbm as lgb
from sklearn.ensemble import RandomForestRegressor

models_lgbm = {}
models_rf = {}

for target in target_cols:
    print(f"\n Training models for target: {target}")

    # LightGBM
    train_data_lgb = lgb.Dataset(X_train_combined, label=y_train[target])
    model_lgb = lgb.train({'objective': 'regression', 'metric': 'rmse'}, train_data_lgb)
    models_lgbm[target] = model_lgb
    print(f" LightGBM trained for {target}")

    # Random Forest
    model_rf = RandomForestRegressor(n_estimators=100, max_depth=10, random_state=42)
    model_rf.fit(X_train_combined, y_train[target])
    models_rf[target] = model_rf
    print(f" Random Forest trained for {target}")
```

# 6. Evaluation and Results

```
metrics_df_final
```

| | Model | Target | RMSE | MSE | MAE | R2 |
|---|---|---|---|---|---|---|
| 0 | SVR | energ_kcal | 175.133897 | 30671.882040 | 133.820454 | -0.027792 |
| 1 | SVR | protein_g | 10.520046 | 110.671367 | 8.065439 | -0.053153 |
| 2 | SVR | lipid_tot_g | 17.193001 | 295.599271 | 9.222288 | -0.121599 |
| 3 | KNN | energ_kcal | 99.938004 | 9987.604664 | 51.302048 | 0.665323 |
| 4 | KNN | protein_g | 6.064179 | 36.774268 | 2.551013 | 0.650054 |
| 5 | KNN | lipid_tot_g | 10.416489 | 108.503249 | 4.496018 | 0.588303 |
| 6 | Random Forest | energ_kcal | 39.233587 | 1539.274353 | 21.668598 | 0.948420 |
| 7 | Random Forest | protein_g | 3.422979 | 11.716786 | 2.012628 | 0.888503 |
| 8 | Random Forest | lipid_tot_g | 5.049943 | 25.501928 | 2.534189 | 0.903237 |
| 9 | TabTransformer_LightGBM | energ_kcal | 17.429645 | 303.792532 | 7.766922 | 0.989820 |
| 10 | TabTransformer_LightGBM | protein_g | 1.030600 | 1.062136 | 0.508992 | 0.989893 |
| 11 | TabTransformer_LightGBM | lipid_tot_g | 2.192605 | 4.807519 | 0.797293 | 0.981759 |
| 12 | TabTransformer_RandomForest | energ_kcal | 21.572699 | 465.381338 | 9.213726 | 0.984405 |
| 13 | TabTransformer_RandomForest | protein_g | 1.461816 | 2.136906 | 0.584519 | 0.979665 |
| 14 | TabTransformer_RandomForest | lipid_tot_g | 2.868862 | 8.230369 | 0.937682 | 0.968771 |

# References

## Core Libraries and Frameworks


- **Python:** https://docs.python.org/3/
- **NumPy**: https://numpy.org/doc/
- **Pandas**: https://pandas.pydata.org/docs/
- **Scikit-learn**: https://scikit-learn.org/stable/documentation.html


## Deep Learning Frameworks


- **PyTorch**: https://pytorch.org/docs/stable/index.html
- **TensorFlow (Keras API)**: https://www.tensorflow.org/api_docs/python/tf/keras


## Ensemble Learning & Gradient Boosting


- **LightGBM**: https://lightgbm.readthedocs.io/en/latest/
- **XGBoost**: https://xgboost.readthedocs.io/en/stable/


## Visualization Tools


- **Matplotlib**: https://matplotlib.org/stable/users/index.html
- **Seaborn**: https://seaborn.pydata.org/

- **Plotly for Python**: https://plotly.com/python/

**Development & Experimentation**

- **Jupyter Notebook**: https://jupyter.org/documentation