Comparison of SVM, RFR, and ANN models' performance on an ingredient substitution regression task

Timofei Polivanov - Sami Rahali

Advancements in the AI and ML fields have enabled innovative applications across different domains, including the culinary industry. One challenge in this field is the identification of ingredient substitutions for people with dietary restrictions or allergies with the same nutritional value and flavor profile. This paper explores the effectiveness of different ML models in predicting optimal substitutions based on two primary factors: nutritional content and flavor similarity.

To achieve this, we constructed two structured datasets: one for retrieving ingredient parameters and another for training machine learning models by mapping original ingredients to their respective substitutes. The dataset creation process involved integrating multiple sources of nutritional, flavor, and substitution data.

Four machine learning models were evaluated: Linear Regression (baseline), Support Vector Machines (SVM), Neural Networks (NN), and Random Forest Regression (RFR). The models were trained and tested using a dataset comprising 90,283 rows and 14 features, where 7 features described the original ingredient, and 7 target variables described the substitute. The models were assessed based on performance metrics such as Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R2 score, and Explained Variance Score (EVS).

Results show that sometimes simpler models can perform better, with the RFR model outperforming all other models, including the NN. Additionally, results proved that optimization through automatic hyperparameter tuning is very effective, as all tuned models performed better than their default counterparts. Future work includes expanding and enhancing the dataset and exploring hybrid models.

Machine learning - Deep learning - Regression

Introduction

Advancements in artificial intelligence have enabled the development of machine learning, a technology proven to enhance efficiency and solve complex problems across various domains. One such application is in the culinary world, where identifying suitable ingredient substitutions for individuals with allergies or dietary restrictions remains a challenge. The study by Shirai et al. [1] emphasizes that the ability to accurately and efficiently recommend alternative ingredients that keep the same nutritional value and original flavor profile is a necessity. In this paper, the machine learning models will predict ingredient substitutions based on two factors: nutritional content and flavor profile. Nutritional information includes macronutrients (proteins, fats, carbohydrates) and micronutrients (vitamins, minerals) [2]. Flavor profiles are compared using cosine similarity, which measures the angle between ingredient vectors to calculate their similarity [3].

Machine learning models can identify ingredient substitutions by utilizing large datasets of nutritional and flavor information to make predictions [4]. Among the numerous machine learning models available, this paper will focus on Neural Networks, Support Vector Machine, Random Forest Regression models, and Linear Regression.

Neural networks (NN) are models based on the function and structure of the human brain. They consist of interconnected layers of neurons that process data through a series of transformations, enabling the model to learn patterns from large datasets. By performing nonlinear transformations, neural networks can learn complex relationships between features [5]. This makes them effective for identifying ingredient substitutions, as they account for nuanced connections of attributes, in particular maintaining the nutritional information and the flavor of the dish.

Support Vector Machines (SVM) are supervised learning models that work by finding the optimal boundary, known as a ‘hyperplane’, that separates data into distinct classes [6]. It is useful in classification tasks where boundaries need to be established between different ingredients. Moreover, SVMs can also handle high-dimensional data, making them compatible with complex data, such as the nutritional information of ingredients based on multiple variables.

Random Forest Regression (RFR) is a model that works by constructing multiple decision trees during training and combining their outputs to make predictions. Each tree in the forest is constructed using a random subset of the data, which helps reduce overfitting and improves generalization. When making a prediction, the model averages the outputs of all individual trees to obtain a final result, making it robust and less prone to errors [7]. This characteristic makes the model well-suited for ingredient substitution tasks where robustness is important.

Linear Regression (LR) is a supervised learning model that predicts by finding the best-fitting line through data points. This line, known as the 'regression line', is determined using the 'least squares method', which minimizes the sum of the squared differences between the predicted and actual values [8]. Due to its simplicity and limitation to modeling only linear relationships, it serves as a baseline for comparing the more advanced models discussed earlier.

This paper will evaluate four models:, Neural Network (NN), Support Vector Machine (SVM), and Random Forest Regression (RFR), and Linear Regression (LR) as a baseline using the following performance metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R^2 score, and

Explained Variance Score (EVS). The evaluation aims to address the following research question:

‘**How do neural network, support vector machine, and random forest regression models compare in the ingredient substitution regression task?**’

Methodology

Dataset Creation

Multiple sources were combined and processed step by step to develop the final datasets used in our project. The goal was to create two final datasets:

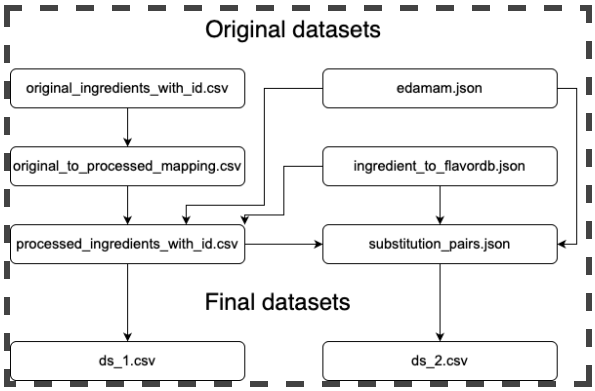
1. A dataset that provides nutritional and flavor-related parameters for a given ingredient based on its name or ID. We'll refer to it as 'ds_1.csv'.
2. A dataset that will be used for training machine learning models, where each sample consists of an original ingredient's features and the corresponding substitution ingredient's target values. Importantly, it is abstracted from the specific ingredients' names and IDs and only maps the parameters of one to another. We'll refer to it as 'ds_2.csv'.

Original datasets

The following datasets served as the foundation for our dataset construction [10]:

- Ingredients data
  - original_ingredients_with_id.csv: Lists original, unprocessed ingredient names with their IDs.
  - processed_ingredients_with_id.csv: Provides a standardized list of ingredient names and corresponding processed IDs.
  - original_to_processed_mapping.csv: Maps original ingredients to their processed versions.
- Nutritional and flavor data
  - edamam.json: Contains nutritional information such as energy content, protein, fat, carbohydrates, and fiber for processed ingredients.
  - ingredient_to_flavordb.json: Maps processed ingredients to their flavor profiles from FlavorDB.
- Substitution data
  - substitution_pairs.json: Defines substitution relationships between ingredients, linking processed ingredient IDs.

Data processing steps



Scheme of sources for the final datasets

The construction of the final datasets involved the following steps.

Step 1: Keep only processed ingredient information

Since substitutions and nutritional data were available only for processed ingredients, and there were a lot of duplicate or similar entries for

original ingredients, we removed columns related to original ingredients.

```
Load dataset "a_ds.csv"
Remove columns: "original_id",
"original"
Save dataset as "aa_ds.csv"
```

Step 2: Remove duplicates

To ensure each ingredient appears only once, we eliminated duplicate entries based on 'processed_id'.

```
Load dataset "aa_ds.csv"
Remove duplicates using
"processed_id" as the unique key
Save dataset as "aaa_ds.csv"
```

Step 3: Integrate nutritional information

We extracted nutritional data from 'edamam.json' and added attributes such as category, label, weight, and five key nutrients.

```
Load dataset "aaa_ds.csv"
Load JSON file "edamam.json"
For each ingredient in the dataset:
    Retrieve corresponding
nutritional information
    Assign values for category,
label, weight, and nutrients
(ENERC_KCAL, PROCNT, FAT, CHOCDF,
FIBTG)
If no match is found, assign default
missing values
Save dataset as "aaaa_ds.csv"
```

Step 4: Integrate flavor information

We added flavor data from 'ingredient_to_flavordb.json', including 'flavordb_id' and 'cosine_similarity', to the dataset.

```
Load dataset "aaaa_ds.csv"
Load JSON file
"ingredient_to_flavordb.json"
For each ingredient in the dataset:
    Retrieve corresponding flavor
profile
    Assign values for flavordb_id and
cosine_similarity
If no match is found, assign default
missing values
Save dataset as "aaaaa_ds.csv"
Save dataset as "ds_1.csv"
```

Step 5: Integrate substitution information

After adding all of the needed information about the original ingredients, we added ingredient substitution data from 'substitution_pairs.json'. This means that each ingredient now was linked to its substitute ingredient(s).

```
Load dataset "aaaaa_ds.csv"
Load JSON file
"substitution_pairs.json"
Extract relevant fields:
"ingredient_processed_id",
"substitution",
"substitution_processed_id"
Merge substitution data into dataset
based on "processed_id"
Expand rows to handle multiple
substitutions per ingredient
Save dataset as "aaaaaa_ds.csv"
```

Step 6: Integrate nutritional information for substitutions

Now, we need to add the same information for the substitute ingredients. Similar to Step 3, we

extracted nutritional attributes for substitute ingredients.

```
Load dataset "aaaaaa_ds.csv"
Load JSON file "edamam.json"
For each substitution in the dataset:
    Retrieve nutritional values
    Assign category, label, weight,
and nutrients (ENERC_KCAL, PROCNT,
FAT, CHOCDF, FIBTG)
If no match is found, assign default
missing values
Save dataset as "aaaaaaa_ds.csv"
```

Step 7: Integrate flavor information for substitutions

We repeated the flavor data integration process for substitute ingredients.

```
Load dataset "aaaaaaa_ds.csv"
Load JSON file
"ingredient_to_flavordb.json"
For each substitution in the dataset:
    Retrieve flavor attributes
    Assign values for flavordb_id and
cosine_similarity
If no match is found, assign default
missing values
Save dataset as "aaaaaaaa_ds.csv"
```

Step 8: Final cleanup

Finally, we removed unnecessary columns, keeping only the 7 input features (for the original ingredient) and 7 target values (for the substitution ingredient).

```
Load dataset "aaaaaaaa_ds.csv"
Drop redundant columns:
    "processed_id", "processed",
"processed_category",
"processed_label", "processed_weight"
    "substitution",
"substitution_id",
"substitution_category",
"substitution_label",
"substitution_weight"
Save final dataset as
"aaaaaaaaa_ds.csv"
Save final dataset as "ds_2.csv"
```

Final Datasets

The dataset creation process resulted in two key datasets:

1. Ingredient feature dataset (ds_1): Allows retrieval of 7 nutritional and flavor-related parameters based on an ingredient's name or ID. This is needed to use the model: first, given an ingredient's name, you retrieve its 7 parameters, pass it through the model, which gives the optimal parameters for the substitution, then you find the ingredient with the closest parameters, and return its name.
2. Model training dataset (ds_2): Maps 7 features of an original ingredient to 7 target values of a substitution ingredient for training ML models. This results in a model that, based on the 7 parameters of a given ingredient, can predict the most optimal parameters for a substitution.

These structured and cleaned datasets enabled the comparison of LR, NN, SVM, and RFR models in predicting suitable ingredient substitutions.

Model Training

Choice of Models

For our ingredient substitution model, we experimented with four different machine learning models:

- Linear Regression (LR): Chosen as the baseline due to its simplicity and interpretability.
- Support Vector Machine (SVM): A robust model known for its ability to handle high-dimensional data and capture complex patterns.
- Neural Network (NN): A deep learning approach that can model non-linear relationships between features and targets.
- Random Forest Regression (RFR): An ensemble learning method that leverages multiple decision trees to improve predictive accuracy and reduce overfitting.

Justification for model selection

We chose these models for different reasons.

- Linear Regression (baseline): LR is computationally efficient and provides a reference for evaluating the complexity and effectiveness of other models.
- Support Vector Machine: SVM is useful when dealing with complex feature interactions but comes at a high computational cost.
- Neural Network: NN can learn intricate patterns in data, making it well-suited for non-linear relationships in ingredient substitution.
- Random Forest Regression: RFR leverages multiple decision trees, making it robust against overfitting and effective in capturing feature interactions.

Training process

The dataset consists of 90,283 rows and 14 columns, where 7 features describe the processed ingredient and 7 target variables describe the substitution ingredient. All models followed a standardized training pipeline:

1. Data preprocessing
    - Standardized features using StandardScaler to normalize values.
    - Split the dataset into 80% training and 20% testing.

2. Default model training
    - Trained each model with default hyperparameters to evaluate their initial performance.

3. Hyperparameter tuning
    - Implemented automated hyperparameter tuning for LR, NN, and RFR.
    - Used GridSearchCV for LR and RFR.
    - Used Keras tuner for NN.
    - SVM tuning was attempted but aborted due to excessive runtime.

Below is a more detailed explanation of the process for each model.

First, we load, scale, and split the dataset into features and targets:

```
Load dataset from file path
Normalize dataset using
StandardScaler
Select feature columns as X
Select target columns as y
Return X, y
```

Then, we split data further into train and test datasets:

```
Split X and y into training and
```

```
testing sets using train_test_split
Return X_train, X_test, y_train,
y_test
```

After this, we train each model with default parameters and then with automatically found best-performing hyperparameters. In the examples, we will provide pseudocode for the latter (except for SVM, due to reasons explained previously).

Baseline model: Linear Regression (LR)

Linear Regression fits a linear model to minimize the mean squared error (MSE) between the predicted and actual values. The optimized version used Ridge and Lasso Regression with hyperparameter tuning via GridSearchCV to improve regularization.

```
Define alpha values for
hyperparameter tuning
Perform GridSearchCV for Ridge
Regression using alpha values
Select best Ridge model
Perform GridSearchCV for Lasso
Regression using alpha values
Select best Lasso model
Train best Ridge model on X_train and
y_train
Train best Lasso model on X_train and
y_train
Predict using Ridge model on X_test
Predict using Lasso model on X_test
```

Support Vector Machine (SVM)

SVM for regression attempts to find a hyperplane that best fits the data. However, it was the only model that underperformed compared to LR. Additionally, the automatic hyperparameter tuning took excessively long to execute, preventing us from obtaining optimized results.

```
Initialize SVR model with RBF kernel,
C=1.0, epsilon=0.1
Convert SVR to handle multi-output
regression using
MultiOutputRegression
Train model on X_train and y_train
Predict using trained model on X_test
```

Neural Network (NN)

A feedforward neural network with two hidden layers was used. It was trained using the Adam optimizer and MSE loss function. For the optimized version, Keras tuner was used to search for the best number of layers, neurons per layer, activation functions, and learning rate.

```
Define model-building function with
tunable hyperparameters:
    - Number of hidden layers (1-3)
    - Number of neurons per layer
(16-128)
    - Activation function (relu,
tanh, selu)
    - Learning rate (0.001, 0.0005,
0.0001)
Use Hyperband tuning to search best
hyperparameters
Retrieve best model configuration
Train best model on X_train and
y_train for 100 epochs
Predict using trained model on X_test
```

Random Forest Regression (RFR)

RFR is a collection of decision trees trained on bootstrapped samples of the dataset. The optimized RFR used GridSearchCV to find the

best hyperparameters, such as the number of trees and the depth of trees.

```
Define hyperparameter search space
for Random Forest:
    - Number of estimators (50, 100,
200, 300)
    - Max depth (None, 10, 20, 30)
    - Min samples split (2, 5, 10)
    - Min samples leaf (1, 2, 4)
    - Max features (auto, sqrt)
    - Bootstrap (True, False)
Perform RandomizedSearchCV to find
best hyperparameters
Select best Random Forest model
Train model on X_train and y_train
Predict using trained model on X_test
Compute MSE and MAE for predictions
Return trained model, MSE, MAE
```

The final step was also the same for all models and involved calculating the metrics and outputting results.

```
Calculate MSE, RMSE, MAE, R2, and EVS
Output evaluation metrics
```

Results

The performance of the models was evaluated using five different metrics: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), R^2 Score, and Explained Variance Score (EVS). The results are summarized in the table below.

Model	MSE	RMSE	MAE	R2 Score	EVS
RFR auto	0.6698	0.8180	0.5246	0.3320	0.3325
RFR	0.6736	0.8207	0.5177	0.3279	0.3286
NN auto	0.7183	0.8476	0.5700	0.2800	0.2805
NN	0.7409	0.8637	0.5857	0.2558	0.2580
LR auto	0.8608	0.9278	0.6527	0.1417	0.1418
LR	0.8608	0.9278	0.6527	0.1417	0.1418
SVM	0.8867	0.9416	0.5308	0.1158	0.1596
SVM auto	N/A	N/A	N/A	N/A	N/A

Explanation of Metrics

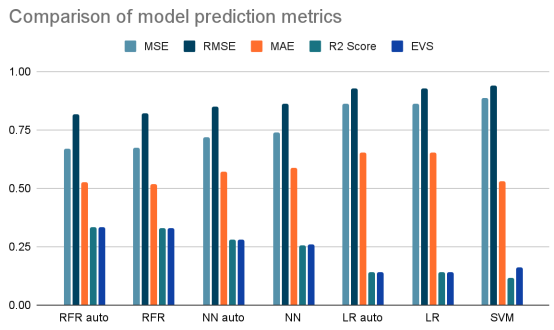
- Mean Squared Error (MSE): Measures the average squared difference between predicted and actual values. Lower values are better.
- Root Mean Squared Error (RMSE): The square root of MSE, providing error values in the same unit as the target variable. Lower values are better.
- Mean Absolute Error (MAE): Measures the average absolute differences between predicted and actual values. Lower values are better.
- R^2 Score: Represents the proportion of variance explained by the model, ranging from negative values to 1. Higher values are better.
- Explained Variance Score (EVS): Similar to R^2, but less sensitive to outliers. Higher values are better.

In general, models with lower MSE, RMSE, and MAE and higher R^2 and EVS perform better and

have a stronger predictive ability. The best-performing model was the auto-tuned Random Forest Regression (RFR auto), while the worst-performing model was the Support Vector Machine (SVM). Hyperparameter tuning was not performed for SVM auto due to excessive run time.

Discussion

Model performance



The results indicate that the RFR with automatic hyperparameter tuning achieved the best performance, having the lowest MSE and MAE, meaning that it made the most accurate predictions compared to other models. This aligns with the fact that ensemble models like RF usually perform well in regression tasks, however, this goes against our expectations of the NN performing the best.

The NN models also performed relatively well. However, they still lagged behind the RFR models, which may indicate that the dataset size or feature representation was not fully optimized for deep learning approaches. LR models had significantly higher MSE and MAE, indicating a poorer fit to the data, which is expected given that linear regression assumes a linear relationship between the feature and the target variables. The SVM model unexpectedly performed the worst, demonstrating the highest MSE and lowest R2 score, even below the baseline, meaning that it couldn't capture the patterns in the data and it isn't suitable for this task.

Considerations

Another notable trend in the results is that all auto-tuned models outperformed their counterparts with a default configuration, particularly in MSE. This shows the importance of optimization through hyperparameter tuning in machine learning, as small adjustments can lead to significant improvements in performance.

Additionally, the relatively high MSE and RMSE across all models suggest that there is room for improvement in feature selection and dataset quality. The dataset currently relies on nutritional and flavor profiles, but integration of additional features, such as the chemical composition of ingredients, could enhance model performance, as it would provide more specifics and details to the substitution pairs. Additionally, different scaling approaches and train/test split percentages can be tested.

Potential improvements and future work

1. Feature expansion: Adding chemical composition data or detailed physical properties (e.g., melting point, solubility) could improve performance.
2. Dataset expansion: Expanding the dataset with additional ingredient substitution examples (i.e., making the dataset larger) could improve performance.
3. Dataset processing: Further testing should be conducted on the impact of alternative scaling methods and train/test split percentages on the training performance.

4. Advanced model architectures: Hybrid models that combine ensemble learning with neural networks (e.g., an RFR-ANN model) could improve performance [9].
5. Explainability methods: SHAP or feature importance analysis could be used to understand the importance of each of the features, and thus, less important features could be omitted, which could potentially improve performance.

Conclusion

The study demonstrated that the Random Forest Regression with automatic tuning is the most effective model for ingredient substitution prediction. However, future improvements in feature engineering and dataset processing could enhance performance even further. Additionally, exploring hybrid models and integrating more concrete domain-specific knowledge could be promising directions for future work.

References

1. S. S. Shirai, O. Seneviratne, M. E. Gordon, C.-H. Chen, and D. L. McGuinness, 'Identifying Ingredient Substitutions Using a Knowledge Graph of Food', *Frontiers in Artificial Intelligence*, vol. 3, 2021. [Online]. Available: https://pmc.ncbi.nlm.nih.gov/articles/PMC7861309. [Accessed: Mar. 28, 2025].
2. H. K. Biesalski and J. Tinz, 'Micronutrients in the life cycle: Requirements and sufficient supply', *NFS Journal*, vol. 11, pp. 1-11, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2352364617300846. [Accessed: Mar. 28, 2025].
3. P. Xia, L. Zhang, and F. Li, 'Learning similarity with cosine similarity ensemble', *Information Sciences*, vol. 307, pp. 39-52, 2015. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/S0020025515001243. [Accessed: Mar. 28, 2025].
4. J. Loesch, I. van Lier, A. de Boer, J. Scholtes, M. Dumontier, and R. Celebi, 'Automated identification of healthier food substitutions through a combination of graph neural networks and nutri-scores', *Journal of Food Composition and Analysis*, vol. 125, no. 2, pp. 1–10, 2024. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0889157523007032. [Accessed: Mar. 28, 2025].
5. N. Kriegeskorte and T. Golan, 'Neural network models and deep learning', *Current Biology*, vol. 29, no. 7, pp. R231–R236, 2019. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0960982219302040. [Accessed: Mar. 28, 2025].
6. T. Fletcher, 'Support Vector Machines Explained', 2008. [Online]. Available: https://www.csd.uwo.ca/~xling/cs860/papers/SVM_Explained.pdf. [Accessed: Mar. 28, 2025].
7. L. Breiman, 'Random Forests', *Machine Learning*, vol. 45, no. 1, pp. 5-32, 2001. [Online]. Available: https://link.springer.com/article/10.1023/A:1010933404324. [Accessed: Mar. 28, 2025].
8. T. M. H. Hope, 'Chapter 4 - Linear Regression', in *Machine Learning: Methods and Applications to Brain Disorders*, A. Mechelli and S. Vieira, Eds., Academic Press, 2020, pp. 67–81. [Online]. Available: https://www.sciencedirect.com/science/article/abs/pii/B9780128157398000043. [Accessed: Mar. 28, 2025].
9. A. Roy, 'Hybrid Random Forest Regression and Artificial Neural Networks for Modelling and Monitoring the State of Health of Li-Ion Battery', *Journal of Electrical Systems*, vol. 20, pp. 2231–2243, 2024. [Online]. Available: https://www.researchgate.net/publication/379866586_Hybrid_Random_Forest_Regression_and_Artificial_Neural_Networks_for_Modelling_and_Monitoring_the_State_of_Health_of_Li-Ion_Battery. [Accessed: Mar. 28, 2025].
10. K. Raj, 'Multimodal Ingredient Substitution', Kaggle, 2024. [Online]. Available: https://www.kaggle.com/datasets/kanakraj/multimodal-ingredient-substitution. [Accessed: Mar. 28, 2025].