AI-Powered Recipe Difficulty Classification

Natalya Smith

August 13, 2024

Abstract

In today's digital culinary landscape, an overwhelming abundance of online recipes leaves home chefs pondering the challenge level of each dish. To address this dilemma, an innovative solution is proposed — an AI-powered recipe difficulty classifier. This machine learning (ML) model harnesses the ingredients list and cooking instructions to categorise recipes as 'easy,' 'hard', 'medium', or 'very hard'. This system streamlines the selection process for cooks, empowering them to choose recipes that align with their skills and time constraints. The project explores various ML models, ranging from traditional algorithms to neural networks (NN). The final custom NN model, purpose-built for the task, achieved impressive results with an accuracy of 0.7503, precision of 0.7573, and F1 score of 0.7530. Unlike established architectures, this model employs an embedding layer, GlobalAveragePooling1D layer, and dense layers, emphasising a nuanced evaluation of recipe complexity by considering metrics derived from ingredients and preparation steps. Beyond its culinary applications, this project paves the way for adaptable systems that can accommodate diverse cuisines and dietary preferences, enhancing the culinary experience for users worldwide. The source code and project details are available on the GitHub repository, offering a valuable tool for culinary enthusiasts and professionals alike.

1 Introduction

In the vast digital culinary landscape, home chefs often wonder what they can make with ingredients on hand and how challenging a recipe might be. It is vital for chefs to quickly gauge recipe difficulty to match their skills and available time. To assist, we propose an artificial intelligence(AI)-powered system to classify recipe difficulty levels. The model takes into account the ingredients list and cooking directions to determine whether a recipe falls into categories such as 'easy', 'hard', 'medium', and 'very hard'. This system aims to streamline the recipe selection process by offering cooks a quick insight into the complexity of a recipe, aiding them in making informed decisions based on their skill level and time constraints[1].

Various ML models were employed to identify the most proficient one for subsequent predictions. Additionally, Local Interpretable Model-agnostic Explanations (LIME) technique was utilised to understand the predictions made by the model. The exploration began with several models, initiating with a baseline model and progressing through a series of more complex algorithms. The final custom NN model, despite its simplicity, demonstrated exceptional performance with an accuracy of 0.7503, a precision of 0.7573, and an F1 score of 0.7530.

Several aspects underscore this project's novelty and contribution to the field of culinary AI. In particular, the trained model provides a nuanced evaluation of recipe complexity, utilising explicit metrics derived from the ingredients and preparation steps, rather than merely counting

[1]For a possible scenario on the application of such a model, see Appendix A.

steps or ingredients. This allows for a more detailed and accurate assessment of a recipe's difficulty, taking into account various factors that contribute to the complexity of preparing the dish. Ensuring a balanced representation of various difficulty levels in the training and validation dataset, the model's predictions are unbiased, and it is equally exposed to all difficulty classes during training. Implementing a robust cross-validation methodology ensures that the model's performance metrics are representative of its general predictive power, reinforcing the reliability of the proposed model. The source code and the project's brief can be found on the project's GitHub repository here. Contributions are welcomed with the hope of expanding the repository to support a wide range of use cases.

This report proceeds as follows. Section 2 discusses related literature and provides an overview of the dataset used in the project. Model development, training and evaluation is discussed in Section 3. Section 4 summarises the performance across different ML models and section 5 presents model's explanations based on LIME. Section 6 concludes and includes a critical reflection, discusses project's limitations and suggests future work avenues.

2 Related Literature and Dataset Overview

Recent advancements in NN architectures [1] have paved the way for innovative research in the domain of culinary content. For example, [2] undertook the classification of food images into 101 distinct categories, utilising a dataset comprising approximately 100,000 images. The availability of the Recipe1M+ dataset [3],[4] has been a catalyst for numerous studies, opening doors to exciting possibilities. Notably, [5] merged the Recipe1M+ dataset with an extensive collection of 13 million food images, aiming to create joint embeddings of recipes and images for improved coherence between text and visuals. [3] and further [4] employed the Recipe1M+ dataset to generate simplified recipes while omitting ingredient quantities and units, evaluating their models with perplexity scores and textual-image coherence metrics.

These efforts have collectively demonstrated the power of deep NNs and large-scale culinary datasets. A new task of generating full recipes with quantities and units was proposed in [6], these authors published a carefully prepared RecipeNLG daatset, specifically tailored for tasks in Natural Language Generation to ease the process of generating and evaluating recipes. TheRecipeNLG builds upon this rich history of culinary AI research as in [3], [4], [5], among others, offering a new resource for enhancing recipe generation and text-to-image understanding[2]. It offers a wide range of recipes featuring different cooking methods and types of dishes. Every recipe provides detailed information about ingredients, their quantities, preparation steps, and cooking instructions. A wealth of culinary data is contained within this dataset, originating from various websites, with each website contributing a significant volume of recipes.

However, to maintain the manageability of the project while ensuring its feasibility within the available resources and timeframe, the decision was made to work with a carefully selected subset of the dataset. This, as a result, provided a more manageable amount of data without compromising the quality of the analysis. The www.foodnetwork.com subset was chosen due to its substantial size, making it a representative sample of the larger dataset. This specific website's data forms a valuable foundation for the project, allowing for the exploration of culinary trends, patterns, and insights while avoiding the complexity associated with handling the entire dataset. The dataset used contains 49,443 records.

[2]see https://recipenlg.cs.put.poznan.pl/dataset

3 Model Development, Training and Evaluation

Python was utilised for all stages of model design, training, and assessment. The dataset underwent pre-processing, and recipes were programmatically assigned difficulty levels based on criteria such as the number of ingredients and the length of preparation steps. We introduced the metric *Step Complexity*, determined by the average word count of all preparation steps. Another metric, *Technique Diversity*, was assessed by counting various cooking techniques in the preparation, and then standardizing by the total techniques referenced. Additionally, *Ingredient Diversity* was determined by counting unique ingredients.

After computing these metrics, they were normalised and weighted to derive a *Total Complexity* score for each recipe. Based on the quantiles of this score, recipes were categorised into "Easy", "Medium", "Hard", or "Very Hard". Notably, the distribution of labels was evenly split: 'Easy', 'Medium', 'Hard', and 'Very Hard' each represented about 25.00%. Such a distribution is ideal for classification tasks, ensuring the model can adequately learn from each category. With balanced classes, the base accuracy is 0.25, implying that a naive or random prediction would be correct 25% of the time.

3.1 Model Training and Evaluation

We trained a text classification model using a *Random Forest Classifier* paired with *Term Frequency-Inverse Document Frequency (TF-IDF) Vectorization*, leveraging the previously calculated complexity metrics as features. Through k-fold cross-validation and a separate test set evaluation, the model yielded a cross-validation accuracy of around 0.9928 and a test accuracy of about 0.9933 (refer to source code). The high accuracy implies the assigned difficulty level labels are both reliable and accurate.

The model training utilised a pipeline, sequentially applying data transformations and the classifier. This pipeline employed *TfidfVectorizer* to numerically transform text data into TF-IDF features suitable for ML, followed by a *Classifier*—with various classifiers tested. We extensively evaluated the model using multiple metrics via Scikit-learn to gauge its efficacy and potential improvement areas. Furthermore, a 5-fold cross-validation assessed the model's robustness and performance by partitioning the dataset into five subsets. The averaged cross-validation scores shed light on the model's consistent predictive accuracy.

4 Performance across ML models

To develop a robust classification model, various ML and deep learning models were trained and evaluated. We next discuss the results of these models[3], offering a comparative discussion on their performance. The following models were evaluated based on their accuracy, precision, recall, and F1 score: Baseline (Dummy Classifier), Logistic Regression (LR), Gradient Boosting Classifier (RBC), Random Forest Classifier (RFC), Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN), Recurrent Neural Network (RNN), Support Vector Machine (SVM), Naive Bayes (NB), Custom Neural Network (NN), see Table 1.

[3]In the presentation, initial results were showcased, serving as a foundational step in this project. Upon closer scrutiny of the model's behaviour, signs of overfitting were observed, prompting a deeper dive into hyperparameter tuning and model refinement. To address these issues, strategic modifications were made to the model, incorporating L2 regularisation, adjusting the dropout rate, and fine-tuning the learning rate schedule. The resultant model exhibited more stable training dynamics, as evidenced by a more harmonious convergence pattern between training and validation loss, and an improved balance between training and validation accuracy. Consequently, in the report, the refined and final results are presented, underscoring the efficacy of the iterative model development process and the robustness of the final model.

Table 1: Summary of Model Performances

Model	Accuracy	Precision	Recall	F1 Score
Baseline	0.2500	-	-	-
Logistic Regression	0.6936	0.6961	0.6936	0.6947
Gradient Boosting	0.6844	0.6846	0.6844	0.6844
Random Forest	0.6360	0.6304	0.6360	0.6324
MLP	0.7183	0.7167	0.7183	0.7170
CNN	0.6900	0.6927	0.6900	0.6908
RNN	0.6821	0.6805	0.6821	0.6807
SVM	0.6668	0.6681	0.6668	0.6674
Naive Bayes	0.4494	0.4680	0.4494	0.4408
Initial Custom NN	0.7523	0.7540	0.7523	0.7529
Final Custom NN	0.7503	0.7573	0.7503	0.7530

A baseline model utilising a Dummy Classifier achieved an accuracy of 0.2500, establishing a rudimentary performance metric. Referencing Table 1, among the traditional ML models, LR attained the highest F1 score of 0.6947, and an accuracy of 0.6936, outperforming GBC and RFC in terms of both accuracy and F1 score. SVM yielded an accuracy of 0.6668 and an F1 score of 0.6674, notably surpassing the NB model which achieved an accuracy of 0.4494 and an F1 score of 0.4408.

The MLP shows superior performance with an accuracy of 0.7183 and an F1 score of 0.7170, edging out the CNN and RNN models. The custom NN model had the highest accuracy and F1 score of 0.7523 and 0.7529, respectively, among all models, indicating that tailored architectures might yield enhanced performances. However, after analysing this model further, we noticed that the model was overfitting. This had led to a thorough review and adjustment of hyperparameters and model structure.

Further refinement of the model had resulted in a model with steadier training patterns. This had been reflected in the closer alignment of training and validation metrics. The final NN model shows an impressive accuracy of 0.7523 and an F1 score of 0.7529, underscoring the potential of tailored NN designs as shown in Table 1.

In terms of its architecture:

- Embedding Layer transforms the input sequence of word indices into dense vectors of fixed size, `output_dim=32`. This is the first layer after the input and is primarily used for dimensionality reduction and to capture semantic meanings of words.
- LSTM[4] Layer has 64 units.
- GlobalAveragePooling1D Layer averages the sequence dimension and outputs a fixed-size vector, which allows the model to handle input of variable length.
- Dense Layer is a fully connected layer with 128 neurons and ReLU activation function. This layer is regularised using L2 regularisation to prevent overfitting.
- Dropout Layer, with a rate of 0.6, randomly sets a fraction of input units to 0 at each update during training, which helps to prevent overfitting.
- Output Layer is a dense layer with 4 neurons (equal to the number of classes in the dataset, i.e., 4) uses a softmax activation to provide a probability distribution over these 4 classes.

[4]LSTMs are a type of RNN that can capture long-term dependencies and sequential patterns in the data, especially applicable in our case.

In terms of its complexity, the model is moderately complex. It is not a very deep network, but the incorporation of LSTM makes it adept at capturing sequential patterns. The use of regularisation and dropout indicates efforts to combat overfitting, ensuring the model generalises well on unseen data.

The fact that the training loss is decreasing, as shown in Figure 1, indicates that the model is learning and fitting to the training data. The increase in validation loss at about 1 epoch, followed by a decrease at 2 epochs, suggests some volatility or instability in the model's performance on the validation set. This could be due to several reasons including, e.g., random initialisation of weights. Also, learning rate might be a bit high causing oscillations. The batch size and the data distribution in each batch might also have an impact, and so would the noise in the data. The increase in training accuracy suggests the model is becoming better at predicting the correct labels for the training data. The decrease in validation accuracy at 1 epoch followed by an increase at 2 epochs mirrors the behaviour observed in the validation loss. The initial decrease might be an indication that the model started to overfit on the training data but then managed to generalise better in the subsequent epoch.

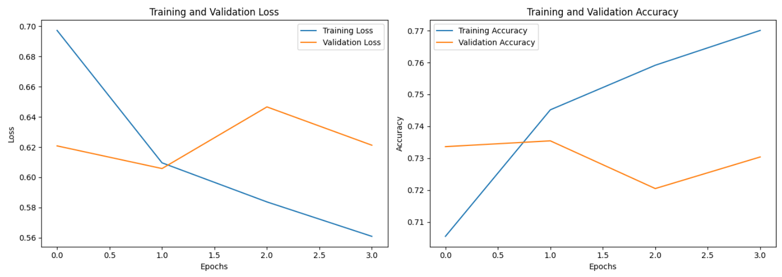


Figure 1: Training and Validation Loss vs. Accuracy

Therefore, such behaviour suggests the model is indeed in the early stages of training. Typically, as training progresses, one hopes to see a steady decrease in both training and validation loss, and a steady increase in both training and validation accuracy. If the validation loss starts to increase consistently while the training loss continues to decrease, it might be an indication of overfitting[5]. Overall, this model is designed as a balance between simplicity and complexity.

5 Interpretation with LIME

Our final task was to interpret the predictions of the best-performed text classification model (we used the final custom NN model as best performing).[6] The results from LIME interpretaions are shown in Figure 2. It allows to understand the model's predictions on a per-instance basis, which is crucial, especially in applications where the interpretability of predictions is essential[7]. One of its advantages is that by understanding which features are contributing the most to a particular prediction, we can debug and improve our models[7].

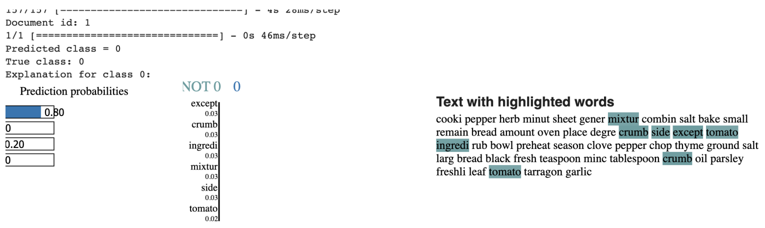
LIME enhances interpretability in text classification tasks by pinpointing words vital to the model's decision-making on a case-by-case basis. This approach involves altering the input data,

[5]This means that the model is getting very good at the training data but is losing its generalisation capability on new, unseen data.

[6]This is a straightforward feed-forward NN designed for text classification, involving embedding and average pooling to handle sequential data.

[7]For instance, if a model is placing undue emphasis on a feature that is known to be irrelevant or noisy, this can be identified and addressed.

crafting new texts by excluding specific words, and examining the variations in model predictions. LIME then forms a new, easy-to-interpret dataset by deploying an interpretable model, often linear regression, to shed light on the predictions in an accessible format[7].



```
157/157 [==============================] - 4s 28ms/step
Document id: 1
1/1 [==============================] - 0s 46ms/step
Predicted class = 0
True class: 0
Explanation for class 0:
```

Text with highlighted words

cooki pepper herb minut sheet gener mixtur combin salt bake small remain bread amount oven place degre crumb side except tomato ingredi rub bowl preheat season clove pepper chop thyme ground salt larg bread black fresh teaspoon minc tablespoon crumb oil parsley freshli leaf tomato tarragon garlic

Figure 2: Results Interpretation with LIME

In this case, based on the extracted text from the LIME results, the prediction probabilities were as follows:

$$\begin{aligned} P(\text{Class 0, i.e., "Easy"}) &= 0.80 \\ P(\text{Class 1, i.e., "Medium"}) &= 0.00 \\ P(\text{Class 2, i.e., "Hard"}) &= 0.20 \\ P(\text{Class 3, i.e., "Very Hard"}) &= 0.00 \end{aligned}$$

The predicted class matches the true class for this instance, indicating the model made a correct prediction. The model is quite confident in its prediction for class 0 ('Easy'), assigning a probability of 80% to it. The positive influence on prediction had words such as 'mixtury': 0.03 percent, 'ingredi': 0.03, 'side': 0.03, 'tomato': 0.02, and 'crumb': 0.03. The values next to each word (as in, e.g., 0.03 for 'mixtury') represent the weights or importance of each word in influencing the prediction

Since, for example, 'mixtur' is a step frequently found in easy recipes, their presence might push the model towards classifying a recipe as 'Easy'[8]. In essence, the model used the highlighted words to decide that the recipe is likely easy to make. Indeed, this insight are valuable because we are trying to understand why certain recipes are classified in a particular way[9]. That is, the LIME explanation provided insights into the words that were pivotal in determining the difficulty level of a recipe, offering a glimpse into the model's decision-making process[10].

6 Conclusion

This project showcased AI's transformative potential in the culinary field, transitioning from raw data handling to predictive modelling. The ML models serve beyond mere classifiers; they refine user experiences on culinary platforms by tailoring content to individual tastes and skill levels. We explored a range of models from traditional ML to NNs, offering insights on their

[8]Conversely, if certain words were more associated with 'Hard' recipes but had a low or negative weight, it would indicate they did not play a significant role in this specific prediction.

[9]Also valuable when we want to provide feedback to users on what factors (i.e., steps or ingredients) contributed to a recipe's difficulty level

[10]However, it is important to remember that the interpretable model (e.g., a linear model) used by LIME might not have captured fully the complex relationships[7] in the model, especially because it is a deep NN.

classification capabilities[11]. Despite its unique and simple structure, our custom NN achieved an accuracy of 0.7503, precision of 0.7573, and F1 score of 0.7530. Instead of following typical architectures like CNNs or RNNs, it was tailored for our specific application, and optimised to both comprehend and cater to the distinct nature of the data and problem statement, substantiating its classification capability.

This project was a comprehensive learning journey, covering both technical and philosophical dimensions of AI in real-world contexts. It emphasised a user-centric focus and the need for rigorous model evaluation to prevent overfitting. Using LIME, we gained insight into the key recipe elements influencing the model's decisions, enhancing our understanding and trust in the NN. This not only reassured us of the model's robustness but also bridged the gap between the complexity of deep learning and the transparency needed for user trust and debugging.

Even though the project had its successes, it is crucial to recognise its limitations. For example, the intrinsic simplicity of the custom model, while beneficial for computational efficiency, might sacrifice the capability to capture more complex patterns in the data. The interpretability provided by LIME, while invaluable, is an approximation and may not fully encapsulate the intricate decision boundaries formed by the model in the high-dimensional space.

Saying that, this project lays the groundwork for more advanced, user-adaptive culinary systems. It is scalable, accommodating diverse cuisines and diets, and can enhance the cooking experience on various platforms globally. The ML model can be expanded to cover a wide range of recipes, serving both enthusiasts and professionals.

7 Appendices

7.1 Appendix A: Scenario

In your new role as a data scientist at a prominent culinary tech company (or culinary website), your ongoing project will center on the development of an AI-driven recipe difficulty classifier. The core objective is to empower users to effortlessly discover recipes that precisely match their culinary skills and preferences. Leveraging a rich and expansive culinary dataset, encompassing a wide array of culinary information, including ingredients, preparation steps, and relevant details, your task is to engineer a state-of-the-art ML model.

References

[1] Liang, M. and Hu, X. (2015). Recurrent Convolutional Neural Network for Object Recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR).

[2] Bossard, L., Guillaumin, M. and Gool, L. V. (2014). Food-101–Mining Discriminative cComponents with Random Forests. Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, Proceedings, Part VI 13. Springer International Publishing.

[3] Salvador, A., Hynes, N., Aytar, Y., Marin, J., Ofli, F., Weber, I. and Torralba, A. (2017). Recipe1M+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food Images. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 11081-11090.

[11]The task of predicting difficulty level based on recipe text might inherently be challenging. Some recipes might be borderline between difficulty levels.

[4] Salvador, A., Drozdzal, M., Giró-i-Nieto, X. and Romero, A. (2019). Inverse Cooking: Recipe Generation from Food Images. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 10453-10462.

[5] Marin, J., Biswas, A., Ofli, F., Hynes, N., Salvador, A., Aytar, Y., Weber, I. and Torralba, A. (2021). Recipe1m+: A Dataset for Learning Cross-Modal Embeddings for Cooking Recipes and Food images. IEEE transactions on Pattern Analysis and Machine Intelligence, 43(1), pp. 187-203.

[6] Bień, M., Gilski, M., Maciejewska, M., Taisner, W., Wisniewski, D. and Lawrynowicz, A. (2020). RecipeNLG: A Cooking Recipes Dataset for Semi-Structured Text Generation. In Proceedings of the 13th International Conference on Natural Language Generation, pp. 22-28.

[7] Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). 'Why should I trust you?' Explaining the Predictions of Any Classifier. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 1135-1144.