REPORT

FOOD RECIPE SUGGESTION FROM DETECTED INGREDIENTS

Group 4

Chotiros Srisiam #101411914

Pat Boonprasertsri #101410612

Pek Chansatit #101439953

AASD 4016: Full Stack Data Science Systems Applied A.I. Solutions Development

Contents

Project Overview

In today's fast-paced world, many people struggle with meal planning while shopping, leading to frustration and inefficiency. To address this challenge, our project aims to revolutionize the retail experience by introducing a food recipe suggestion feature powered by AI technology. By leveraging machine learning algorithms and image recognition technology, we seek to offer personalized recipe recommendations to customers based on their preferences and available ingredients. This innovative solution not only simplifies the meal planning process but also enhances the overall shopping experience, making it more enjoyable and efficient for customers. Through this initiative, we aspire to drive customer engagement, satisfaction, and loyalty while positioning our organization as a leader in leveraging technology to meet the evolving needs of consumers in the retail industry.

Problem Statement

Many people struggle with meal planning while shopping, leading to frustration and inefficiency. Traditional shopping experiences often lack personalized recommendations and guidance for customers, particularly in the realm of meal planning and recipe suggestions. This gap in service leaves customers feeling disconnected and underserved, exacerbating the challenges of meal planning while shopping.

Significance

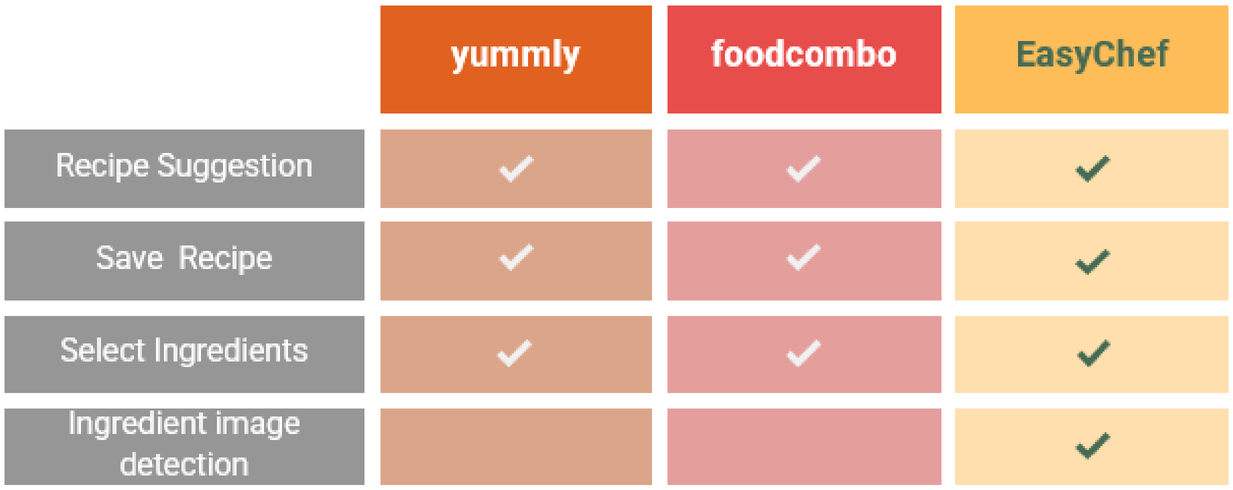
To address this challenge, we aim to offer a solution that simplifies meal planning and enhances the overall shopping experience. By leveraging AI technology to provide personalized recipe suggestions based on customer preferences and available ingredients, we seek to transform the way customers shop for groceries. This innovative approach not only streamlines the meal planning process but also enriches the shopping experience, ultimately leading to greater enjoyment and satisfaction for customers.

Literature Review

In exploring existing research and initiatives in the domain of meal planning and recipe suggestion, two prominent reference sources stand out: Yummly and Foodcombo.

1. **Yummly:** Yummly, a popular digital platform, has been at the forefront of providing personalized recipe recommendations to users based on their preferences and dietary restrictions. Their innovative approach to leveraging machine learning algorithms and user data has garnered widespread acclaim in the food-tech industry.
2. **Foodcombo:** Similarly, Foodcombo offers a comprehensive database of ingredient combinations and recipes, allowing users to discover new and exciting culinary creations.

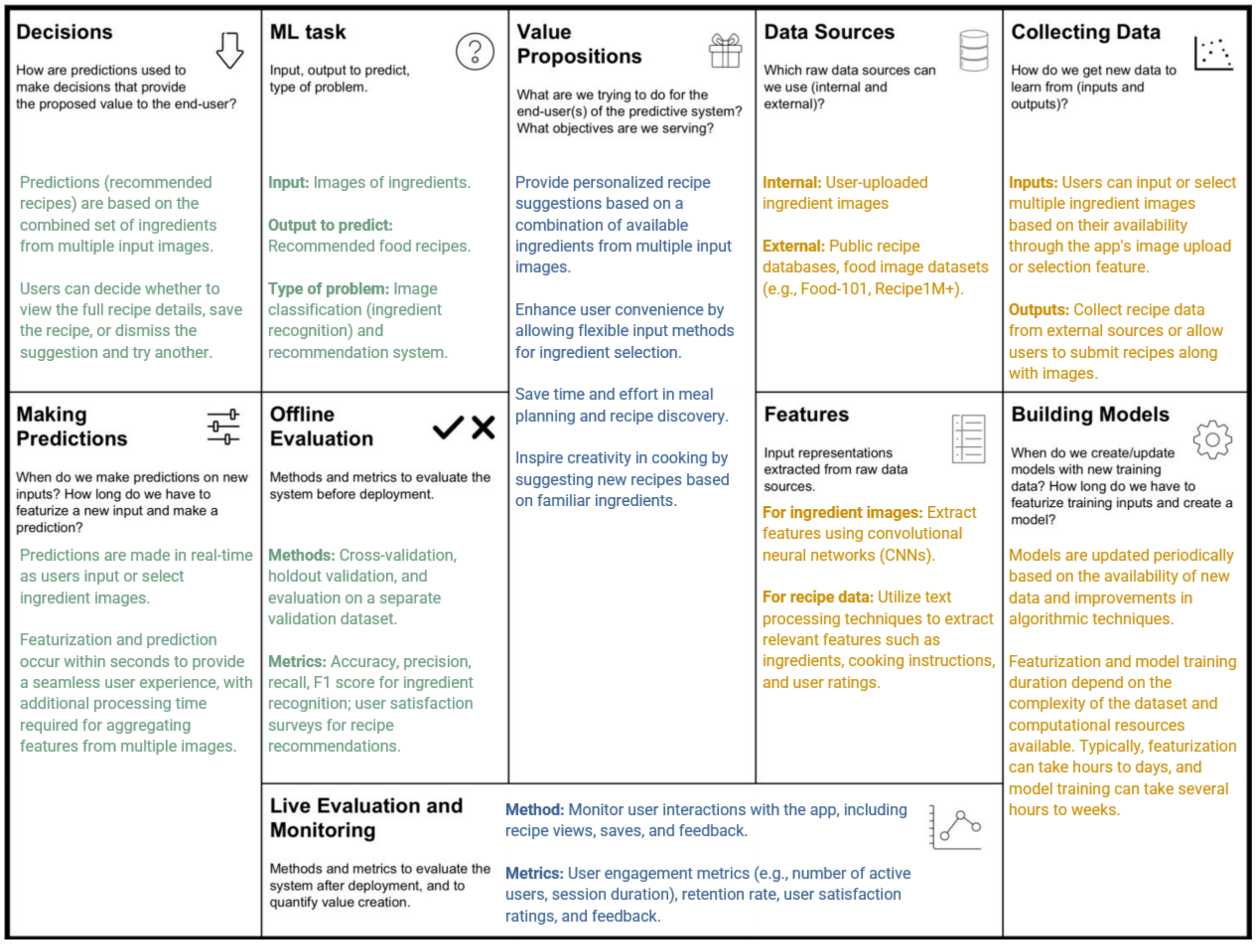
By analyzing and synthesizing insights from these reference sources, we aim to gain a deeper understanding of best practices and emerging trends in meal planning and recipe suggestion, which will inform the development of our own solution with an ingredient detection feature.



	yummly	foodcombo	EasyChef
Recipe Suggestion	✓	✓	✓
Save Recipe	✓	✓	✓
Select Ingredients	✓	✓	✓
Ingredient image detection			✓

Figure 1: Function Comparison Table

Methodology – ML Canvas



Decisions

How are predictions used to make decisions that provide the proposed value to the end-user?

Predictions (recommended recipes) are based on the combined set of ingredients from multiple input images.

Users can decide whether to view the full recipe details, save the recipe, or dismiss the suggestion and try another.

ML task

Input, output to predict, type of problem.

Input: Images of ingredients.

Output to predict: Recommended food recipes.

Type of problem: Image classification (ingredient recognition) and recommendation system.

Value Propositions

What are we trying to do for the end-user(s) of the predictive system? What objectives are we serving?

Provide personalized recipe suggestions based on a combination of available ingredients from multiple input images.

Enhance user convenience by allowing flexible input methods for ingredient selection.

Save time and effort in meal planning and recipe discovery.

Inspire creativity in cooking by suggesting new recipes based on familiar ingredients.

Data Sources

Which raw data sources can we use (internal and external)?

Internal: User-uploaded ingredient images

External: Public recipe databases, food image datasets (e.g., Food-101, Recipe1M+).

Collecting Data

How do we get new data to learn from (inputs and outputs)?

Inputs: Users can input or select multiple ingredient images based on their availability through the app's image upload or selection feature.

Outputs: Collect recipe data from external sources or allow users to submit recipes along with images.

Making Predictions

When do we make predictions on new inputs? How long do we have to featurize a new input and make a prediction?

Predictions are made in real-time as users input or select ingredient images.

Featurization and prediction occur within seconds to provide a seamless user experience, with additional processing time required for aggregating features from multiple images.

Offline Evaluation

Methods and metrics to evaluate the system before deployment.

Methods: Cross-validation, holdout validation, and evaluation on a separate validation dataset.

Metrics: Accuracy, precision, recall, F1 score for ingredient recognition; user satisfaction surveys for recipe recommendations.

Features

Input representations extracted from raw data sources.

For ingredient images: Extract features using convolutional neural networks (CNNs).

For recipe data: Utilize text processing techniques to extract relevant features such as ingredients, cooking instructions, and user ratings.

Building Models

When do we create/update models with new training data? How long do we have to featurize training inputs and create a model?

Models are updated periodically based on the availability of new data and improvements in algorithmic techniques.

Featurization and model training duration depend on the complexity of the dataset and computational resources available. Typically, featurization can take hours to days, and model training can take several hours to weeks.

Live Evaluation and Monitoring

Methods and metrics to evaluate the system after deployment, and to quantify value creation.

Method: Monitor user interactions with the app, including recipe views, saves, and feedback.

Metrics: User engagement metrics (e.g., number of active users, session duration), retention rate, user satisfaction ratings, and feedback.

Figure 2: Machine Learning Canvas

Model Benchmarking

For our project, which focuses on ingredient detection in food images, we conducted a comprehensive benchmarking exercise involving three popular deep learning architectures: YOLOv5, MobileNetV2, and ResNet50. Each of these architectures offers unique strengths and capabilities, making them suitable candidates for our task.

The primary goal of this benchmarking process was to identify the model that demonstrates superior performance in terms of accuracy, speed, and efficiency. By systematically evaluating these models against a standardized set of metrics and benchmarks, we aimed to make an informed decision regarding the most suitable model for our application.

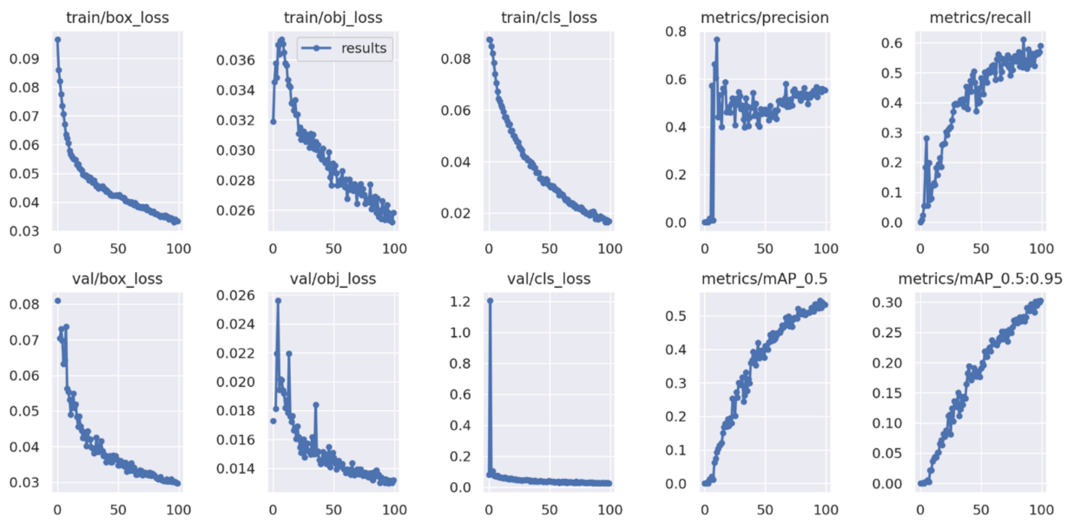


Figure 3: YOLOv5 Result

- **YOLOv5:** YOLOv5n model is learning to detect ingredients with improving accuracy throughout the training process. There seems to be a good balance between box, object, and classification loss, while the mAP_0.5 metric remains promising.

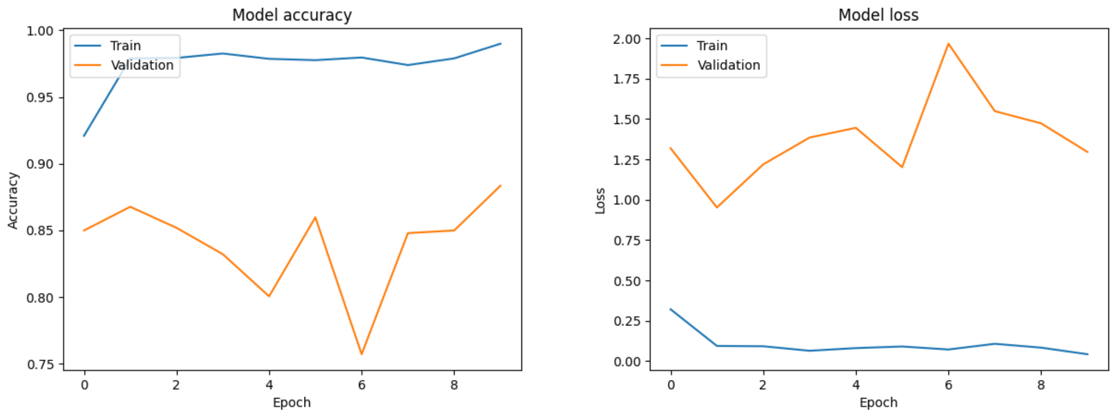


Figure 4: MobilenetV2 Result

- **MobilenetV2:** The MobileNetV2 model's accuracy is high throughout the training epochs, and it seems to be improving as the training progresses. The loss curve is also steadily decreasing, which indicates that the model is learning to minimize errors.

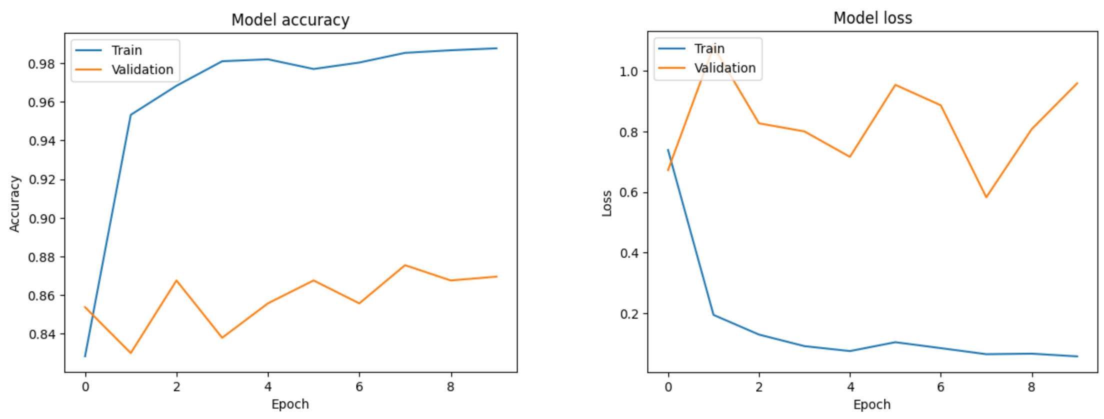
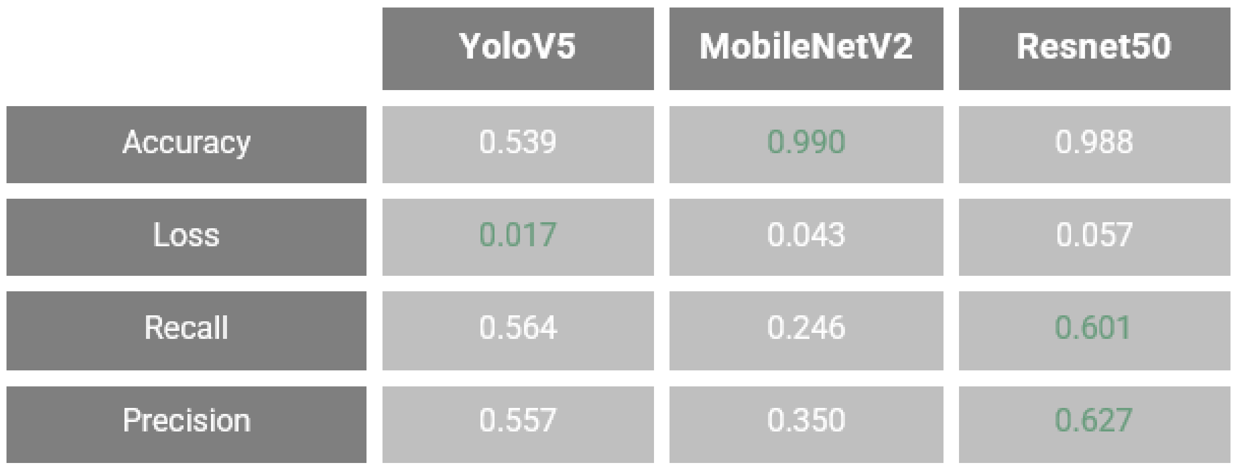


Figure 5: ResNet50 Result

- **ResNet50**: The accuracy on the training set reaches up to 98% while the validation accuracy stays around 96%. This suggests that the model is performing well on the data it trained on and generalizes reasonably well to unseen data. The loss curve also shows a signifi cant decrease throughout the epochs, indicating the model is effectively learning to reduce errors. However, it's important to note that a slight overfitting might be happening since the training accuracy is a bit higher than the validation accuracy. This could potentially lead to the model performing poorly on completely new data. Overall, while the results are promising, further investigation might be needed to determine if ResNet50 is the optimal model for this task.



	YoloV5	MobileNetV2	Resnet50
Accuracy	0.539	0.990	0.988
Loss	0.017	0.043	0.057
Recall	0.564	0.246	0.601
Precision	0.557	0.350	0.627

Figure 6: Model Benchmarking Table

After thorough experimentation and analysis, we concluded that YOLOv5 emerged as the top-performing model for ingredient detection in our project.

- **Balance between Speed and Accuracy:** YOLOv5 is known for being a good balance between speed and accuracy. This is important for ingredient detection because you want the model to be able to identify ingredients quickly and accurately.
- **Ease of Use:** YOLOv5 is a relatively easy-to-use model, which can be helpful for projects where you don't have a lot of experience with deep learning.
- **Versatility:** YOLOv5 can be used to detect a wide variety of objects, which could be helpful if you want to expand the scope of your ingredient detection project in the future.

Model Deployment

1. **Transition from Notebook to Production:** Our first step in model deployment involves transitioning from Jupyter notebook code to standalone scripts. This ensures that our code is modular, well-organized, and suitable for production environments.
2. **Script to Web Service:** Next, we will deploy our model as a web service using Flask for website framework and FastAPI for API endpoints detection. This enables users to interact with our model through a web interface or programmatically via API calls.
3. **Containerization with Docker:** To enhance reproducibility and scalability, we will containerize our application using Docker. Containerization allows us to encapsulate our web application and its dependencies into lightweight, portable containers, making it easier to deploy and manage across different environments.
4. **Cloud Hosting:** For deployment, we will leverage cloud hosting solutions, such as DigitalOcean, to deploy our containerized web application. Cloud platforms offer scalability, reliability, and flexibility, allowing us to easily scale our application based on demand and ensure high availability.
5. **Consideration for Web Application Deployment:** Additionally, we will consider various web application deployment strategies, such as serverless computing, microservices architecture, and Kubernetes orchestration, to optimize performance and resource utilization based on our application requirements.

Challenges & Issues Encountered

1. **Dataset label (Recipe format, limited labeled data):** One of the primary challenges we encountered during the development process was the format of the dataset labels. The recipe format varied across various sources, making it challenging to standardize the data for training our model. Additionally, we faced limitations in the availability of labeled data, which impacted the performance and generalization of our model.
2. **Image format (From Mobile Phone):** Another significant challenge stemmed from the diverse image formats captured from mobile phones. These images varied in resolution, quality, and aspect ratio, posing challenges for preprocessing and feature extraction. Ensuring consistency and accuracy in image processing techniques proved to be a critical aspect of our model development.
3. **Limit resource (e.g., CPU, GPU, memory, and disk space):** The project encountered constraints in computational resources, including CPU, GPU, memory, and disk space.

Training machine learning models, especially deep learning models, requires substantial computational power and memory, which posed challenges due to limited resources. This limitation affected the scale and complexity of our model training process, necessitating optimizations and trade-offs to achieve satisfactory results within resource constraints.

4. **Running Docker**: After setting up the Docker environment and configuring the required libraries, we encountered issues with importing functions from another Python script.