Smart Recipe Finder

Sepideh Forouzi (101599207)

George Brown College

Introduction

- Goal: Build an intelligent recipe-classification and generation system
- Dataset: Kaggle “Food Ingredients and Recipes”, **508 unique ingredients**, **~43,000 recipes**
- Task: Predict cuisine type based on ingredients
- ML Model: TF–IDF vectorizer + Logistic Regression classifier
- Deployment: Convert notebook to script → Docker container → Google Cloud Run
- Outcome: A scalable, real-time web application

Why the Problem Is Significant

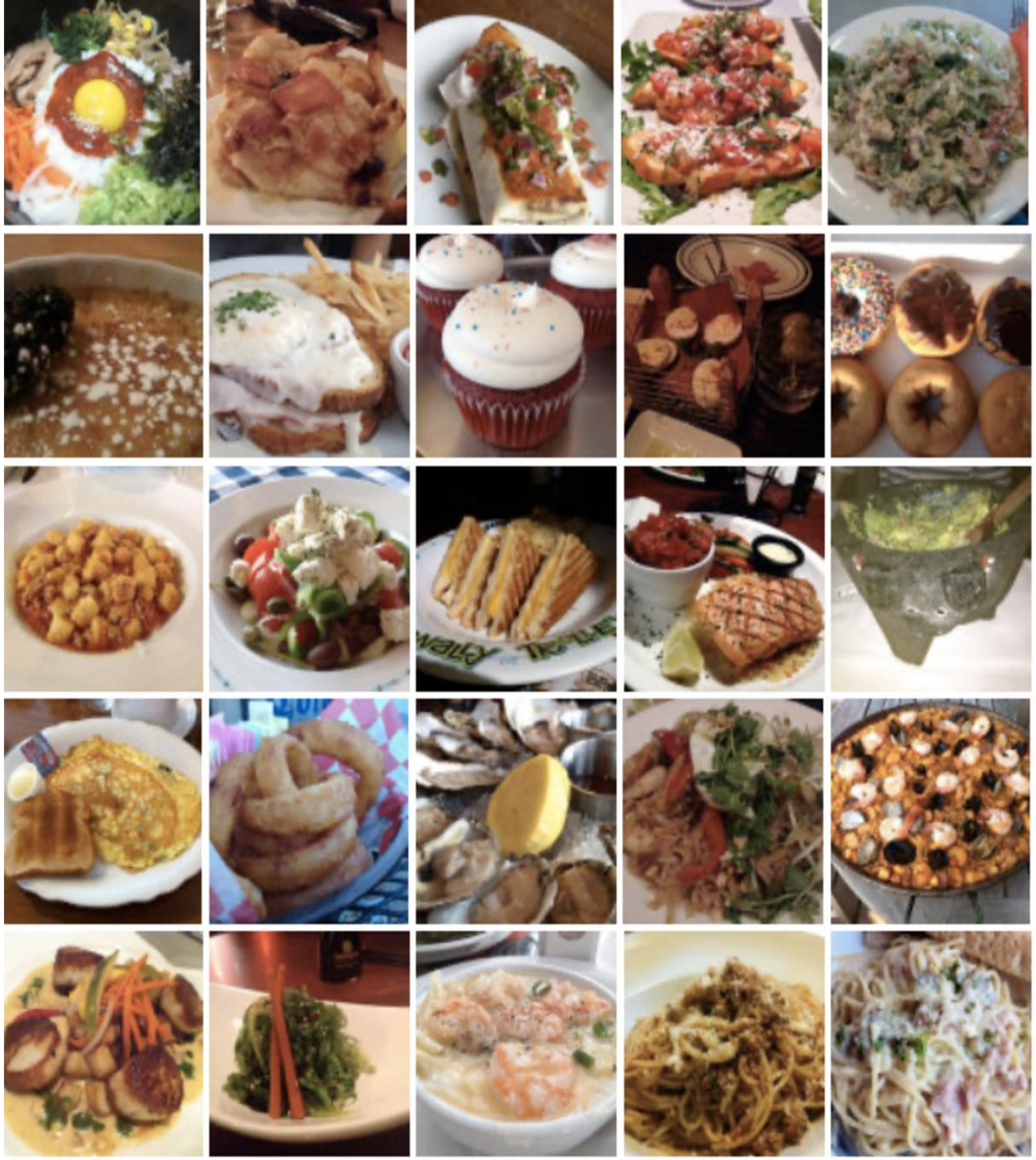
- Helps reduce food waste by suggesting
- recipes based on available ingredients.
- Saves user time and increases convenience.
- Supports healthier cooking choices by showing alternative options.
- Enhances personalization in recipe recommendation systems.

Problem Statement

- Online cooking platforms contain millions of recipes.
- Users often struggle to identify recipes based on the ingredients they already have.
- Searching manually is slow, inefficient, and often irrelevant.
- We need an automated system that understands ingredients and predicts the correct cuisine category.

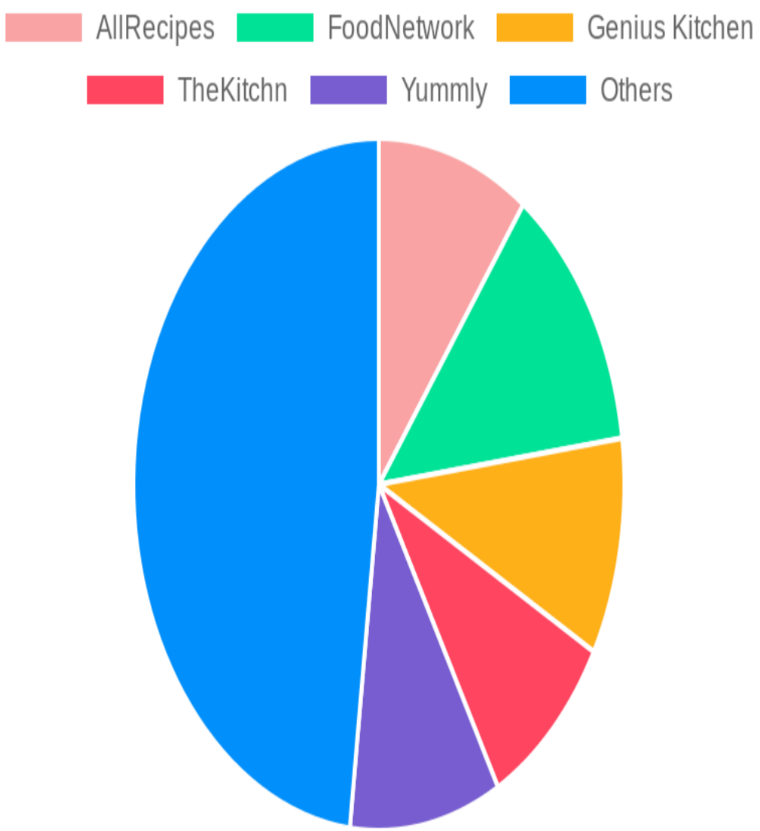
Dataset

Source: Kaggle — *Recipe Ingredients Dataset*
Size:
•~20,000+ recipes
•50+ cuisine labels
•Each sample contains ingredient list + cuisine label
Why this dataset?
•Real-world data
•Clean ingredient lists
•Ideal for TF-IDF + classification

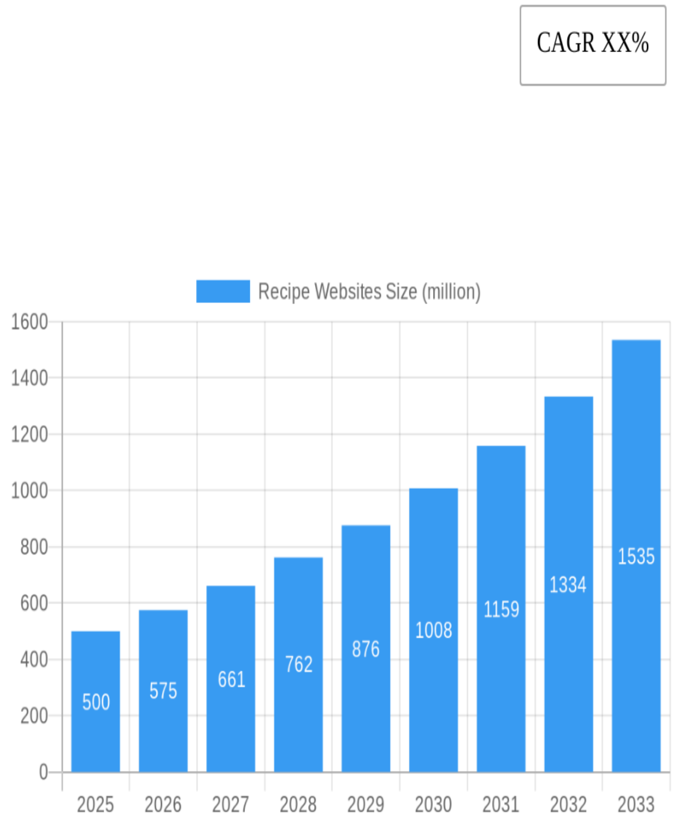


Literature Review / Similar Projects

- Existing systems provide keyword search, not intelligent classification.
- Many apps rely on manual tagging, not machine learning.
- Few works provide ingredient-based cuisine prediction.
- Most previous systems lack modern deployment pipelines (Docker + Cloud).

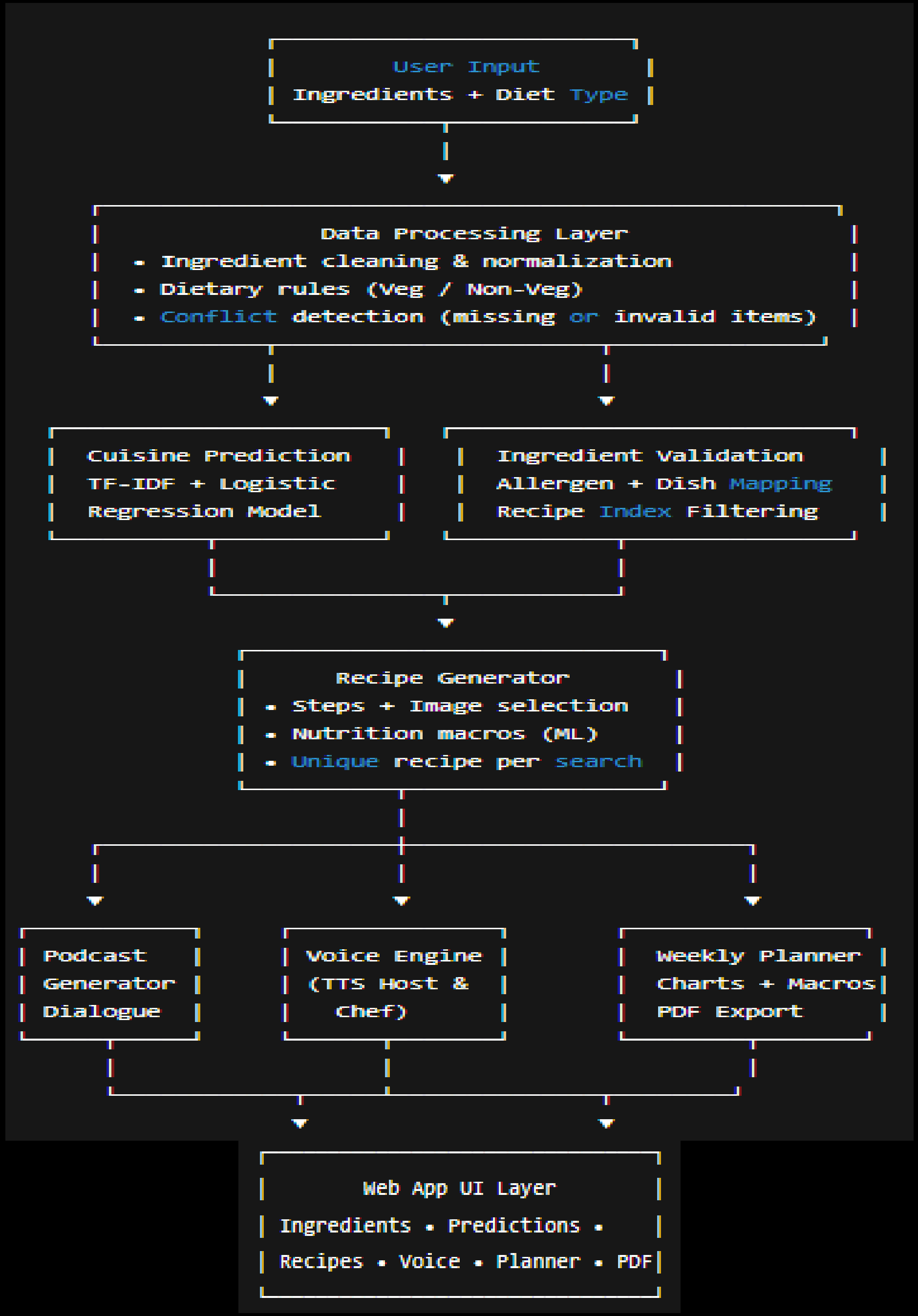


Recipe Websites Leading Players



Model Building

- Cleaned and normalized ingredients
- Applied TF-IDF vectorization
- Trained **Logistic Regression** classifier
- Saved pipeline using joblib
- Tuned vocabulary and parameters for higher accuracy
- **Strengths:**
- Fast
- Lightweight
- Deployment-friendly
- High accuracy for sparse text



The Machine Learning Canvas

PREDICT

Generate cuisine predictions from new ingredient lists using the trained TF–IDF + Logistic Regression model, returning the most likely cuisine and its confidence score.

ML task

Supervised Machine Learning — multi-class text classification model.
The system uses TF–IDF vectorization to transform recipe ingredients into numerical representations, and a Logistic Regression classifier to predict the cuisine category.

Value Propositions

Instant recipe suggestions based on ingredients users already have
Supports vegetarian and non-vegetarian modes
Reduces food waste by maximizing ingredient usage
Lightweight, fast, and explainable model suitable for real-time prediction
User-friendly interface with clean, structured recipe cards

Data Sources

Kaggle Recipe Ingredients Dataset
Source: https://www.kaggle.com/datasets/kaggle/recipe-ingredients-dataset
Size: ~180,000+ recipes
Includes ingredients and cuisine labels
Trained model saved as cuisine_pipeline.joblib

Collecting Data

Load raw recipe data (ingredients + cuisine)
Clean and normalize ingredient text
Tokenize and build TF–IDF vectors
Train Logistic Regression model
Export trained model using joblib
Store recipe metadata (title, steps, ingredients) in JSON structures

Making Predictions

TF–IDF text representation of ingredients
Normalized tokens (lowercased, unicode-normalized, cleaned)
Ingredient coverage matching
Protein-category detection (fish, chicken, beef, seafood, etc.)
Vegetarian substitutions (soy milk, soy yogurt, plant-based equivalents)

Offline Evaluation

Train a Logistic Regression classifier on TF–IDF features
Perform train/test split and cross-validation
Evaluate using accuracy and confusion matrix
Save final model (joblib)
Integrate ML predictions with rule-based logic for filtering and ranking
Expose inference through a Streamlit application

Features

Ingredients are transformed into high-dimensional TF–IDF vectors representing token-level importance across the dataset.
This provides a quantitative and sparse representation suitable for linear classification models.

building Models

Train a multinomial Logistic Regression classifier on the TF–IDF feature vectors.
Hyperparameters tuned through experimentation, followed by full model training and export for deployment.

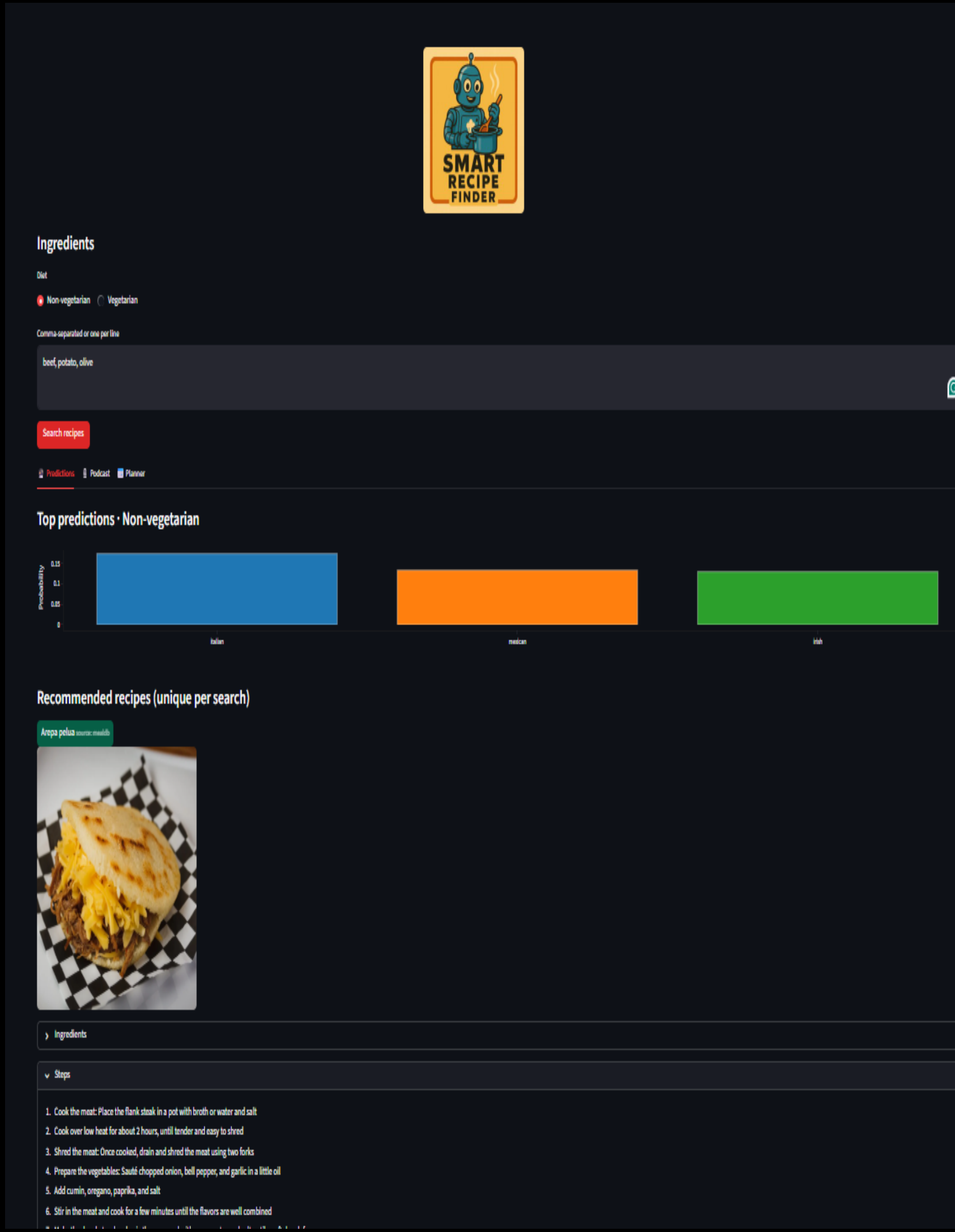
Live Evaluation and Monitoring

Performance validated using a standard hold-out split and 10-fold cross-validation.
Key evaluation metrics include accuracy, macro-F1 score, and confusion matrix analysis to assess class balance and generalization.

Monitor live prediction behavior after deployment on Google Cloud Run, including response latency, request frequency, and prediction distribution across cuisines.
Observe potential drift or misclassification patterns to inform future retraining.

Deployment Pipeline

1. Convert notebook ⟶ Python script
2. Wrap model using FastAPI/Flask-like structure
3. Build Docker image:
   - Installed dependencies
   - Copied model + script
   - Exposed API port
4. Upload Docker image to Google Cloud Artifact Registry
5. Deploy via Cloud Run:
   - Auto-scaling
   - HTTPS endpoint
   - Public access enabled
6. Final deployed app: https://smart-recipe-app-249886303998.northamerica-northeast2.run.app/



Challenges & Issues

•Docker environment conflicts
•Missing dependencies inside container
•Model path issues during deployment
•Cloud Run CPU/memory constraints
•Streamlit ↔ API request handling
•Debugging failed builds
•Ensuring local pipeline = cloud pipeline

How they were solved:
•Rebuilt requirements
•Clean Python environment
•Isolated pipeline in a single script
•Tested container locally before deployment
•Used Google Cloud logs to debug

Conclusion & Recommendations

- Built a complete end-to-end ML system
- Used TF–IDF + Logistic Regression for cuisine classification
- Trained, evaluated, and exported a reproducible ML pipeline
- Containerized the app using Docker
- Deployed the final API + UI on Google Cloud Run
- Gained real experience debugging deployment issues
- Project is fully live, scalable, and ready for future improvements

Thank you
For your
Attention !