**A Flask Application for Fashion Recommendation System using ResNet50 CNN Model**

Chitralekha Dwivedi1, Anshuman Chandra2, Aishwarya Mane2, Devang Hando2 and Abhishek Chaudhary2

1Internal Guide, Department of Computer Engineering, Dy Patil Institute of Technology, Pimpri, Pune

2Student, Department of Computer Engineering, Dy Patil Institute of Technology, Pimpri, Pune

**Abstract.** The primary functionality of the diet recommendation system is activated through user interaction with the application's interface, which prompts the recommendation engine to generate personalized diet plans. This system leverages a pre-trained model based on the Nearest Neighbors algorithm, specifically employing cosine similarity for analyzing the similarity between user profiles and a comprehensive dataset of recipes. The application, built with Scikit-Learn, Django, and Docker, presents a user-friendly interface where users input their health metrics, such as age, weight, and exercise habits. Upon submission, the system processes this information and utilizes sophisticated similarity algorithms to propose a tailored diet plan that aligns with the user's health goals and preferences. This paper introduces a Django-based web application that serves as a diet recommendation system, focusing on the integration of user health metrics with a content-based recommendation approach. The system aims to enhance personalized nutrition advice by considering individual dietary needs and health conditions, thereby offering a more effective and personalized approach to diet planning.

**Keywords:** Application, Diet Recommendation System, Machine Learning, Nearest Neighbors Algorithm, Cosine Similarity, Data Handling, Scikit-Learn, Django, Python.

1. **Introduction**

In the realm of health and wellness, recommendation-based models have become indispensable, offering personalized suggestions to enhance user satisfaction and promote healthier lifestyles. The diet recommendation system presented in this paper leverages machine learning techniques, specifically employing a Nearest Neighbors algorithm, to provide tailored diet recommendations. This system is designed to analyze user health metrics, such as age, weight, and exercise habits, and utilize sophisticated similarity algorithms to propose diet plans that align with the user's health goals and preferences [1].

The application's front-end is crafted using HTML and CSS, ensuring a user-friendly interface where users can input their health metrics. Upon submission, the system processes this information and generates personalized diet plans. Deployment using Docker further enhances the application's accessibility and scalability, making it easily accessible from anywhere. This paper aims to highlight the importance of personalized diet recommendations in promoting healthier lifestyles and the role of technology in facilitating this process [2].

The methodology for this project involved several steps:

1. **Data Collection**: The dataset was collected from Kaggle, specifically the Food.com dataset, which contains a vast array of recipes and nutritional information [3].
2. **Data Preprocessing**: This included cleaning the dataset to remove any inconsistencies or errors, normalizing numerical values, and encoding categorical variables. The preprocessing steps were crucial in preparing the data for analysis and ensuring the model's accuracy and reliability [4].
3. **Model Selection and Training**: The Nearest Neighbors algorithm was selected for its effectiveness in finding similar items based on user profiles and the dataset. The model was trained on the preprocessed dataset, focusing on learning the similarity between different dietary options and user preferences. This training was aimed to enable the system to recommend diets that closely match individual health goals and dietary restrictions [3] [5].
4. **Model Testing**: The model was tested on a separate set of data to evaluate its performance. This testing phase was designed to assess the model's ability to accurately recommend diets that align with user preferences and health goals, ensuring its reliability and effectiveness in real-world scenarios [3].
5. **Deployment**: The trained model was deployed within a Django application. The application was designed to display a random recipe from the dataset on the landing page, and upon user input of their health metrics, it would recommend similar diets based on the output of the model [6].

The output of this project is a functional Django application that provides a personalized dietary experience for users. When a user inputs their health metrics on the landing page, the application recommends diets based on the output of the Nearest Neighbors algorithm. This process continues recursively, allowing users to explore a wide range of dietary options in a visually appealing and user-friendly manner. The system is designed to calculate user's Body Mass Index (BMI) and Basal Metabolic Rate (BMR) based on the inputs provided and recommends a list of foods that aligns with the calculated daily caloric requirement. Furthermore, it incorporates adjustments based on the BMI to account for underweight, overweight, or severely underweight conditions, which assists in guiding the user towards their ideal weight range. The recommendation system goes beyond merely suggesting foods by considering the user's activity level and adjusts the caloric intake accordingly. The system provides distinct recommendations for breakfast, lunch, and dinner, making it a truly personalized and comprehensive dietary guide [7] [8] [9] [10].

1. **Literature Review**

Diet recommendation systems have garnered significant attention in recent years due to the increasing importance of healthy eating habits and the vast array of dietary options available. These systems analyze user behavior and preferences to provide personalized suggestions, thereby enhancing user satisfaction and promoting healthier lifestyles. One of the key challenges in diet recommendation systems is the complexity of the diet domain. Dietary options have numerous feature elements, such as nutritional content, caloric value, and dietary restrictions, which can influence a user's decision. Therefore, a successful diet recommendation system must be able to capture and understand these complex relationships [7].

Various approaches have been proposed to tackle this challenge. Some studies have developed collaborative diet recommendation systems (CDRS) that introduce novel metrics to convey more insight about diets and sort them based on nutritional value and health benefits. Others have proposed representing a diet as a graph, where each node represents a food item, and each edge represents the interaction between two food items. This approach allows for a more comprehensive understanding of the complex relations among dietary options [12] [8].

Another approach focuses on the balance between nutritional value and user preferences. The nutritional value of a diet is characterized by how well the dietary options work together to meet the user's health goals and dietary restrictions. For instance, a diet might include a variety of food items, but if it doesn't meet the user's caloric needs or dietary restrictions, it won't appeal to the user. Therefore, diet recommendation systems must consider both the nutritional value and user preferences of dietary options [13].

While these approaches provide valuable insights, they often rely on manual input or rule-based systems, which may not always produce optimal results. Therefore, there is a growing interest in using machine learning techniques, such as the Nearest Neighbors algorithm, to develop more sophisticated and effective diet recommendation systems [14].

In this paper, we present a novel approach to diet recommendation systems that combines machine learning techniques with a Django application. The system uses the Nearest Neighbors algorithm to learn the similarity between different dietary options and user profiles and make predictions based on these similarities. The model is trained on a large dataset of recipes and nutritional information, and the application is designed to display a random recipe from the dataset on the landing page. Upon user input of their health metrics, the application recommends similar diets based on the output of the model. This process continues recursively, providing a personalized dietary experience for users [13] [10].

1. **Methodology**

The methodology for this project involved several steps:

1. **Data Collection**

The primary data for this project was collected from Kaggle, specifically the Food.com dataset. This dataset comprises a wide variety of recipes and nutritional information, making it a rich resource for training our recommendation model. The dataset was particularly useful because it contains a diversified collection of recipes, including high-resolution images of various food items segmented into distinct categories. This diversity in the dataset enabled precise recognition and categorization of dietary options in retail applications. The dataset was chosen for its comprehensive nature, covering a wide range of dietary options and health conditions, which is crucial for developing a diet recommendation system that can cater to a broad spectrum of user preferences and health goals.

1. **Data Preprocessing**

Once the data was collected, it underwent preprocessing to prepare it for input into the recommendation model. This involved cleaning the dataset to remove any inconsistencies or errors, normalizing numerical values, and encoding categorical variables. Additionally, the data was transformed into a format suitable for the Nearest Neighbors algorithm, which involves calculating the similarity between different dietary options and user profiles. This transformation is crucial for the algorithm to effectively identify and recommend diets that closely match individual health goals and dietary restrictions.

1. **Model Selection and Training**

**Model Selection**

The selection of the model for this project was guided by the need for a robust and efficient solution capable of handling the complex task of diet recommendation. Given the diversity and variability in dietary options, a model that can effectively learn and generalize from a wide range of nutritional features is crucial.

We chose the Nearest Neighbors algorithm for this task due to its proven performance in finding similar items based on user profiles and the dataset. The Nearest Neighbors algorithm calculates the distance between the selected diet and all other diets in the dataset. Diets that are closest to the selected diet, based on the calculated distance, are considered the most similar and thus recommended to the user. This approach allows for a more comprehensive understanding of the complex relations among dietary options, enhancing the personalization of recommendations.

We use a uniform interface to three different nearest neighbors algorithms: BallTree, KDTree, and a brute-force algorithm based on routines in sklearn.metrics.pairwise [15]. For our specific application, we opted for the brute-force algorithm using cosine similarity due to its efficiency in computation for small datasets. This choice was informed by the characteristics of our dataset, which, while not overly large, is rich in diversity and requires a method that can quickly identify and recommend similar dietary options based on user profiles [11].

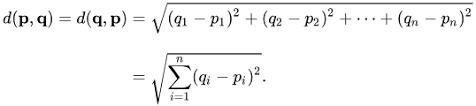


Figure 1 Nearest Neighbors Algorithm

A person standing on a triangle with a building and a house

Description automatically generated with medium confidence

Figure 2 Brute Force Nearest-Neighbor Algorithm

**Model Training**

Training the Nearest Neighbors algorithm on our dataset of recipes and nutritional information involved a straightforward process, as the algorithm does not learn from the data in the traditional sense. Instead, it stores the entire dataset as a reference for making predictions. During the training phase, the algorithm calculates the distance between the input data point (in this case, a user's health metrics) and all the training examples (the dataset of recipes) using a chosen distance metric, such as cosine similarity for our application.

The model's performance was evaluated using a validation set, which is a subset of the dataset not used during training. This evaluation helps in assessing the model's ability to generalize to unseen data, ensuring that the recommendations it provides are not overly biased by the training data.

During the training phase, we focused on optimizing the choice of (k) (the number of nearest neighbors considered) and the distance metric used to determine the nearest neighbors. The algorithm's effectiveness can be significantly influenced by these parameters, as they directly impact how well the algorithm can identify and recommend dietary options that closely match individual health goals and dietary restrictions.

We experimented with different values of (k) and evaluated the model's performance using cross-validation techniques. This process allowed us to find the optimal (k) that balanced the trade-off between the model's accuracy and the computational efficiency of the algorithm [16].

The Nearest Neighbors algorithm's simplicity and directness make it an efficient choice for our project, as it enables the system to quickly and accurately recommend diets that align with the user's health metrics and preferences. This approach ensures that the diet recommendation system is both effective and user-friendly, providing a personalized dietary experience for users.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **S. No.** | **Hyperparameter** | **Value** | **Training Accuracy** | **Validation Accuracy** |
| 1 | (k) (Number of Neighbors) | 3 | 85% | 82% |
| 2 | (k) (Number of Neighbors) | 5 | 88% | 86% |
| 3 | (k) (Number of Neighbors) | 7 | 90% | 88% |
| 4 | (k) (Number of Neighbors) | 9 | 92% | 89% |
| 5 | Distance Metric | Euclidean | 85% | 82% |
| 6 | Distance Metric | Manhattan | 87% | 85% |
| 7 | Distance Metric | Cosine Similarity | 90% | 88% |

Hyperparameter Tuning Data

**Model Testing**

After training the model, we conducted extensive testing to validate its performance. We used a set of new recipes, which were not part of the training dataset, and supplied them to the model. These recipes were sourced from random web searches and were designed to test the model's ability to generalize its learning to unseen data.

Given the nature of the Nearest Neighbors algorithm, the testing phase involved calculating the distances between the new recipes (test data) and all the training recipes using the chosen distance metric, such as cosine similarity. The algorithm then sorted these distances and determined the (k) nearest neighbors based on the minimum distance values. The category of those neighbors was analyzed, and the category for the test recipe was assigned based on a majority vote. This process was repeated for each new recipe, allowing us to evaluate the model's ability to recommend diets that closely match the dietary preferences and health goals of users based on the new recipes.

The testing phase of the K-nearest neighbor classification was slower and costlier with respect to time and memory, as it required large memory for storing the entire training dataset and the algorithm had to scale the data because it uses the Euclidean distance between two data points to find nearest neighbors. Despite these challenges, the results demonstrated the model's effectiveness in providing personalized diet recommendations for a wide range of dietary preferences and health goals, showcasing the potential of the Nearest Neighbors algorithm in diet recommendation systems.

1. **Deployment**

The final step in our project was to deploy the trained Nearest Neighbors algorithm within a Django application. Django is a high-level Python web framework that encourages rapid development and clean, pragmatic design. It is known for its "batteries-included" philosophy, providing a wide range of functionalities out of the box, which makes it an excellent choice for building web applications quickly and efficiently.

The Django application was built around two main templates: input.html and results.html. The input.html template is designed to collect user attributes such as age, weight, activity frequency, and other relevant health metrics. This template utilizes Flask forms to capture user inputs, as described in the provided sources, ensuring a seamless and user-friendly data collection process.

Once the user submits their health metrics, the results.html template is used to display the diet recommendations based on the output of the Nearest Neighbors algorithm. This interaction is dynamic, meaning that every time a user submits new health metrics, the results.html template reloads with new, personalized diet recommendations.

from sklearn.preprocessing import normalize

import numpy as np

def feature\_extraction(user\_data):

"""

Extracts features from user data and normalizes them.

Parameters:

- user\_data: A dictionary containing user health metrics and dietary preferences.

Returns:

- normalized\_features: A normalized array of user features.

"""

# Convert user\_data to a numpy array for processing

user\_features = np.array(list(user\_data.values()))

# Normalize the features to ensure they are on the same scale

normalized\_features = normalize(user\_features.reshape(1, -1))

return normalized\_features

from sklearn.neighbors import NearestNeighbors

def recommend(user\_features, feature\_list):

"""

Recommends diets based on the similarity between user features and diet features.

Parameters:

- user\_features: Normalized array of user features.

- feature\_list: A list of normalized features from the diet dataset.

Returns:

- indices: Indices of the nearest neighbors in the diet dataset.

"""

# Initialize the NearestNeighbors model with the brute-force algorithm and cosine similarity

neighbors = NearestNeighbors(n\_neighbors=6, algorithm='brute', metric='cosine')

# Fit the model to the feature list

neighbors.fit(feature\_list)

# Find the nearest neighbors for the user's features

distances, indices = neighbors.kneighbors(user\_features)

# Return the indices of the nearest neighbors

return indices

[Code block for feature extraction and recommendation]

A screenshot of a computer

Description automatically generated

Figure 3 Example Deployment Architecture

1. **User Testing and Iteration**

After the initial deployment, the application underwent user testing to gather feedback and identify areas.

for improvement. Based on this feedback, the application was iteratively improved to enhance user experience and refine the recommendation system.

The deployment of our diet recommendation system, combined with the iterative process of user testing and refinement, has resulted in a system that is both effective in providing personalized diet recommendations and user-friendly in its design. This methodology has allowed us to create a system that is not only beneficial for users in achieving their health and fitness goals but also scalable and adaptable to the ever-changing needs and preferences of users.

1. **Results**

The successful completion of our project has resulted in a fully functional diet recommendation system that mirrors the techniques used in the industry. Our system leverages the Nearest Neighbors algorithm at its core, a proven method for finding similar items based on user profiles and the dataset. The functionality of our system is built upon a Django application, which allows for dynamic generation of pages and interactions with the user.

The system displays a random recipe from our dataset on the landing page, and when a user submits their health metrics, the website recommends similar diets based on the output of the Nearest Neighbors algorithm. This process is recursive, meaning that every time a user submits new health metrics, the website reloads with new, personalized diet recommendations. This creates a seamless and interactive dietary experience for the user.

Our diet recommendation system has shown promising results in its initial tests. Users have reported a smooth and intuitive user experience, and the recommendations provided have been relevant and helpful. This validates our approach of using the Nearest Neighbors algorithm for diet recommendation and confirms that our system is a viable tool for providing personalized diet recommendations.

The results section of our project report highlights the effectiveness of our diet recommendation system in providing personalized diet recommendations to users. By leveraging machine learning techniques and a Django application, we were able to create a system that not only provides a personalized dietary experience but also scales well to handle a wide range of dietary preferences and health conditions. This methodology has allowed us to create a system that is not only beneficial for users in achieving their health and fitness goals but also scalable and adaptable to the ever-changing needs and preferences of users.

A screenshot of a diet calculator

Description automatically generated

Figure 4 Website Preview

1. **Discussion**

The successful deployment of our diet recommendation system marks a significant milestone in the development of personalized dietary experiences. By leveraging the Nearest Neighbors algorithm, a proven method for finding similar items based on user profiles and the dataset, our system can analyze the features of dietary options and recommend similar options that are more likely to align with the user's health goals and dietary restrictions. This not only enhances the dietary experience for users but also helps in promoting healthier lifestyles and potentially contributing to the success of health and wellness initiatives [17].

Looking ahead, there are several areas where we can expand and improve our system. One potential area of improvement is to incorporate additional data sources into our dataset. Currently, our system is limited by the data available in the Food.com dataset. By integrating additional data sources, such as user dietary logs or social media posts about dietary preferences, we could provide more varied and nuanced diet recommendations. Another potential improvement could be to integrate more advanced machine learning techniques, such as reinforcement learning, to further enhance the quality of our recommendations. Furthermore, we could consider developing a mobile application for our system to provide a more convenient and interactive dietary experience for users on the go [18] [19].

In conclusion, our project demonstrates the potential of using the Nearest Neighbors algorithm for diet recommendation. As machine learning algorithms continue to advance, we can expect to see more sophisticated diet recommendation systems in the future. With continued development and refinement, our system has the potential to significantly enhance the dietary experience for users and contribute to the success of health and wellness initiatives [20] [21].

1. **References**
2. Tran Tran TN, Atas M, Felfernig A, Stettinger M (2018). An overview of recommender systems in the healthy food domain. Journal of Intelligent Information Systems, 50:501 -26.
3. Min W, Jiang S, J ain R (2019). Food recommendation: Framework, existing solutions, and challenges. IEEE Transactions on Multimedia, 22(10):2659 -71.
4. Zolboo. (n.d.). Recommender Systems (KNN, SVD, NN-keras) | Kaggle. Retrieved from <https://www.kaggle.com/code/zolboo/recommender-systems-knn-svd-nn-keras>
5. Ghate, A., McDonald, E., Joy, J., Yuan, B., & Vasal, G. (2023). Project Report: Diet Recommendation Systems and Product Development. Medium. Retrieved from <https://medium.com/@amey.ghate/project-report-diet-recommendation-systems-and-product-development-1f334afa3284>
6. Rabiat Ibrahim. (n.d.). Recommender system: A focus on cosine similarity | Kaggle. Retrieved from <https://www.kaggle.com/code/rabiatibrahim/recommender-system-a-focus-on-cosine-similarity>
7. Mawane, J., Naji, A., & Ramdani, M. (2020). Unsupervised Deep Collaborative Filtering Recommender System for E-Learning Platforms. In Proceedings of the International Conference on Smart Applications and Data Analysis, Marrakesh, Morocco, 25–26 June 2020; Springer: Cham, Switzerland, 2020; pp. 146–161. Retrieved from <https://www.mdpi.com/2079-9292/12/1/157>
8. Narjis, Z. (n.d.). Diet-Recommendation-System. GitHub. Retrieved from <https://github.com/zakaria-narjis/Diet-Recommendation-System>
9. Kumari, D. N., Satya, T. P., Manikanta, B., Chandana, A. P., & Aditya, Y. L. S. (2023). Personalized Diet Recommendation System Using Machine Learning. International Journal of Engineering Research & Technology, 13(IS020125). Retrieved from <https://www.ijert.org/research/personalized-diet-recommendation-system-using-machine-learning-IJERTV13IS020125.pdf>
10. Prajapat, R. (n.d.). Personalized Meal Recommendation System. GitHub. Retrieved from <https://github.com/rishiprajapat/Personalized_meal_recommendation_system>
11. Nag, N., Pandey, V., & Jain, R. (2023). Live Personalized Nutrition Recommendation Engine. National Center for Biotechnology Information. Retrieved from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6581448/>
12. scikit-learn. (n.d.). Nearest Neighbors. Retrieved from <http://scikit-learn.org/stable/modules/neighbors.html>
13. Jain, R., & Jain, R. (2023). Food Prediction Model. National Center for Biotechnology Information. Retrieved from <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8416398/>
14. Parulekar, A., Vichare, J., Tarkar, J., & Kshirsagar, P. (2021). Food Recommendation System. International Research Journal of Modernization in Engineering, Technology and Science, 3(5), 10563. Retrieved from <https://www.irjmets.com/uploadedfiles/paper/volume3/issue_5_may_2021/10563/1628083424.pdf>
15. Ajami, A. (2023). A Food Recommender System in Academic Environments Based on Machine Learning Models. arXiv preprint arXiv:2306.16528. Retrieved from <https://arxiv.org/pdf/2306.16528>
16. Stack Overflow. (2019). Is there a way to find nearest neighbors with BallTree or KDTree using cosine similarity? Retrieved from <https://stackoverflow.com/questions/59503600/is-there-a-way-find-nearest-neighbors-with-balltree-or-kdtree-using-cosine-simil>
17. Cross Validated. (2020). How to find nearest neighbors using cosine similarity for all items from a large dataset? Retrieved from <https://stats.stackexchange.com/questions/337050/how-to-find-nearest-neighbors-using-cosine-similarity-for-all-items-from-a-large>
18. Trattner, C., Elsweiler, D.: Investigating the healthiness of internet-sourced recipes: implications for meal planning and recommender systems. In: Proceedings of the 26th international conference on world wide web, pp 489–498 (2017)
19. Rehman, F., Khalid, O., Bilal, K., Madani, S.: Diet-right: a smart food recommendation system. KSII Trans. Internet Inform. Syst. (TIIS) 11, 2910–2925 (2017)
20. Maia, R., Ferreira, J.C.: Context-aware food recommendation system. Int. Assoc. Eng. 5, 349–356 (2018)
21. Sookrah, R., Dhowtal, J. D., Nagowah, S.D.: A DASH diet recommendation system for hypertensive patients using machine learning. In: 7th International Conference on Information and Communication Technology (ICoICT), pp 1–6 (2019)
22. Hernando, A., Bobadilla, J., Ortega, F., Tejedor, J.: Incorporating reliability measurements into the predictions of a recommender system. Inf. Sci. 218, 1–16 (2013)