Allergen Detection Using Deep Learning

Sean Ruan
University of California, Davis
ECS 171
Davis, US
jhruan@ucdavis.edu

James Kimball
University of California, Davis
ECS 171
Davis, US
jpkimball@ucdavis.edu

Ahmed Irtija
University of California, Davis
ECS 171
Davis, US
airtija@ucdavis.edu

Suparn Sathya
University of California, Davis
ECS 171
Davis, US
spsathya@ucdavis.edu

Junyang Ding
University of California, Davis
ECS 171
Davis, US
jyading@ucdavis.edu

***Abstract*—This document is a model comparison between logistic regression and neural networks. The model ”9375.h5” has the highest accuracy of 93.75%. The logistic regression model has an accuracy of 92.5%.**

***Index Terms*—Methodology, Results**

INTRODUCTION

Food allergens play a significant role in the overall health of the community as well as how food is filtered before it reaches the hands of the consumer. Allergens can pose a variety of risks to people whether it be breakout of hives or much more severe reactions such as anaphylaxis. This model will allow food companies, restaurants, and numerous other vendors to provide greater transparency on the allergens present in their products. The use of machine learning in this field is continuing to grow to the need for text analysis and image classification while also attempting to eliminate human error. Prior research has shown that the results from machine learning models in this field are very promising. The models are able to generate features and do a commendable job when it comes to performing classification. We have seen the use of supervised and unsupervised learning models in this field being specifically applied to textual, labeled data. Since the use of deep learning is becoming more common to determine allergen status the usage of neural networks has increased in frequency. Deep Learning models can generate better results in some cases due to the machine being able to find some set of features that help with classification within the dataset. Allergen status is prone to error due to the variety of ways that foods are prepared, but also because of how models may be implemented. Sometimes models use images and predict what allergens will be present based on an image, and other times it takes in recipes or ingredients and determines from the list what substances are allergens. There could be issues with how the dataset is processed in these scenarios, and in the image classification, the model would need to have a strong training period so that it can accurately perform classification so that it can determine what the correct allergens are.

- Improved symptoms for people with allergies
  - People who have allergies to certain foods can be assisted with the help of a reliable allergen detection model. The model will be able to determine what allergens are present in the food and provide transparency for the customer. It will allow them to know what allergens are present thereby preventing them from unknowingly consuming an allergen.
- Enhanced Food Labeling Compliance
  - The model will be able to detect allergens and because of this provide another method to determine the overall composition of food being sold as well as the allergens present in them. This will allow companies to better follow regulations and compliance policies.
- Cost Reduction for Food Manufacturers
  - The model will be able to detect allergens and because of this provide another method to determine the overall composition of food being sold as well as the allergens present in them. This will allow companies to save money because it will allow them to catch cross-contamination much earlier and not have to spend as much money on recalls or wasting food. Also since the problem is solved before it reaches the consumer it will also help the company image.
- Real-time Monitoring
  - Eventually models can be adapted to take in real-time data and provide predictions and detection of allergens. Since the models are very quick and can also predict potential issues it allows for much faster response times thereby reducing the amount of potential cross-contamination while enhancing safety measures.

There are many uses for an Allergen detection model and it also provides a lot of robustness because of the fact that you can add various other methods such as image classification even in real-time. Eventually, models such as this can be used to scan and analyze food products in real-time and extract their nutritional and allergen information to provide greater convenience in understanding the presence of various vitamins and products in their food.

Literature Review

In the models that were covered in the research papers, we realized that they all used deep learning in order to capture some sort of pattern in the data. Based on the articles we found on Allergen detection we decided to also make a neural network, but we wanted to set up a comparison so that we could compare a deep learning model to a supervised learning model.

The first paper which was written by Wang et al. used deep learning in order to determine the allergenicity of food proteins. They used a deep learning model—transformer with a self-attention mechanism, and ensemble learning models (representative as Light Gradient Boosting Machine (LightGBM), and XGBoost to solve the problem. They then compared the deep learning model to other common existing models, and their result was: "of 5-fold cross-validation showed that the area under the receiver operating characteristic curve (AUC) of the deep model was the highest (0.9578), which was better than the ensemble learning and baseline algorithms." The second paper focused on Kumar et al. work on using ensembles of Extra Tree, Deep Belief Network (DBN), and CatBoost models in order to identify protein allergens. Their model achieved higher accuracy when they combined the results of the different models and used majority voting. The accuracy that they ended up with was 89.16%. The final paper is slightly different from our project because they connected image classification with their allergen detection program. Rohini et al. decided to use a 2-Tab Deep Learning based Application to provide the nutrient and allergen content in fruits and vegetables and, to display allergen information in packaged food using OCR. They used a novel Deep Learning Framework, they can capture a picture of the Fruit or Vegetable so that they can then provide the nutritional facts and allergen information. They fine-tuned their algorithm and obtained an accuracy of 96.37%.

We realized from reading over these papers how common Deep Learning algorithms were in this context. Based on what we read it seems that the researchers wanted to see if the machine could generate features that would help determine if certain allergens were present in future classification. The researchers felt that deep learning models could extract high-dimensional features which is a prediction method that holds promise. They were also impressed by the positive results associated with the use of machine learning, specifically deep learning models in this field.

Data Exploration and Preprocessing

The data set we used was the Allergen Status of Food Products which was in the form of a CSV. The table contains 400 rows and 9 columns: Food Product, Main Ingredient, Sweetener, Fat/Oil, Seasoning, Allergens, Price($), Customer rating (Out of 5), and Prediction. The first 7 columns are all strings that are associated with that column. For example, the Seasoning column will contain different types of seasonings such as Salt, Granola, and Mozzarella, and the Food Product column will contain content such as Chicken Parmesan or Greek Gyro Wrap. The Customer Rating column is just a score that was awarded to that product, and the final column which is the Prediction Column tells if there is an allergen present or not. The data is split into a training set(80%) and a testing set(20%). The purpose of our model is to determine whether an allergen is present in the food product, and what allergens are present in the food product. There were 23 allergens within the data set. All the allergens are listed in Figure 1. From Figure 1, there are 256 food products in the data set containing allergens, and there are 143 food products that do not contain allergens.

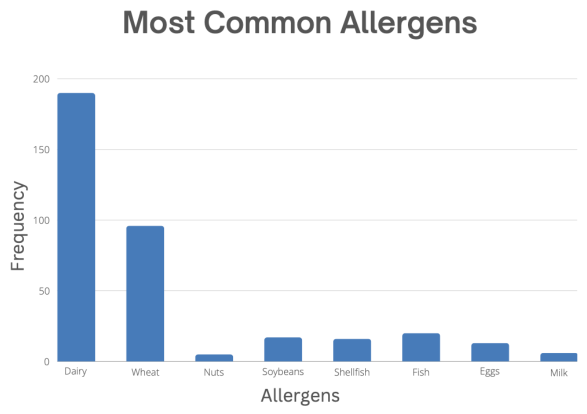


Fig. 1. Most Common Allergens in the data set

Figure 2 shows the frequency of allergens reported in the data set. Based on the data we determined that the most common allergens were Wheat and Dairy which had 190 and 96 instances within the data set respectively. The rest of the allergens in the data set have a frequency between 5-20. It is possible that some of the less frequent allergens such as Coconut which only shows up once in our data set may alter our results. Based on Figure 1 and Figure 2, we conclude that this data set is unbalanced. The number of contains is nearly eighty percent more than does not contain. The frequency of dairy and the frequency of wheat is nearly ten times higher than even other allergens from the data set.

Proposed Methodology

The problem we needed to solve was related to binary classification, which dealt with textual data stored within a table. Based on the information we gathered from previous research papers, we realized that a neural network would be the best way to solve the problem because it would be able to generate features from the data which we could then rely on for classification. We split the project into two parts: The first team worked on the model and the second team worked on the UI and the model integration. Since our data was string input contained within a table, we needed to take this data and store it within a Data frame so it could be easier for us to use in our model. We also made sure that before we ran any of our models we tested to make sure all the cells in the table had legitimate values and used a One-Hot-Encoder in order to transform the Contains column to either a 0 or 1 so our output wouldn't be a string. Furthermore, we also made sure that our input data was encoded using a CountVectorizer so that our rows would be treated as vectors. The vectorizer is used in both our model and our web app so that our data can be processed more easily by the model. Since we were dealing with a smaller data set we realized that it was important for us to be able to give our model as much data as possible. In the case of Logistic Regression specifically, we used SMOTE so that we could generate more data from the minority class. This would allow our model to be much more robust because it would be able to have sizeable amounts of data from both classes. SMOTE helped relieve the issues we had with a small data set and also helped balance out our data. In the case of the neural network, we will certainly need to perform some sort of hyperparameter tuning based on the results we obtained from our original model. We realized that the values and layers we used were values that would serve as good baselines rather than for a final result. After we ran the initial models then we could alter parameters and see if our results got better.

We initially wanted to create a simple web app that would display the results of a pre-trained model. Instead, we tried to make our web application look more appealing and also used it to obtain user input that would be used to perform classification based on user input. The user interface was then improved so that it was easier to use.

Experimental Results

Logistic Regression:

The first model we created was a Logistic Regression model using Scikit-learn's linear model. We performed a split of our data so that our training set was 80% of the total dataset, while the testing set was 20% of the original data set.

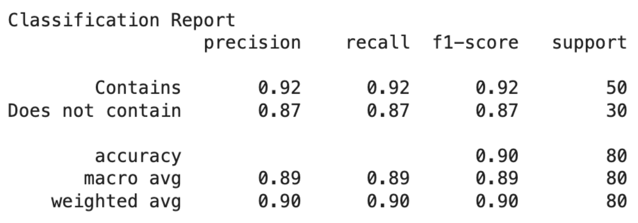

```
Classification Report
                  precision    recall  f1-score   support

        Contains       0.92      0.92      0.92        50
Does not contain       0.87      0.87      0.87        30

        accuracy                           0.90        80
       macro avg       0.89      0.89      0.89        80
    weighted avg       0.90      0.90      0.90        80
```

Fig. 2. Logistic Regression Base Model Report

When we tested the Logistic regression model we used most of the default parameters, but we made the number of max iterations at 10000, and we also used the L2 Penalty for our loss function. The L2 loss function which is calculated as the square root of the sum of the squared vector values. The reason we used L2 loss was because it is generally better at preventing over-fitting than other loss methods. We only ran the Logistic Regression model once because we ended up obtaining an accuracy of 90%. The model performed well in general with both the training and testing data.

After we performed classification with the original model we decided to tune the model by using Grid Search and SMOTE. We first used SMOTE because we have a relatively small, uneven data set and we wanted to be able to generate a balance between samples. SMOTE would allow our model to see more data and increase the likelihood that it would be able to draw generalizations. After applying SMOTE we then decided to use Grid Search because we wanted to test out various hyperparameters. In the Grid Search, we primarily tested out different values of C. The tuning process had a max number of iterations set at 10000 and had a CV of 5. We wanted to make sure that our balance and scoring were fair which is why our scoring used the f1-score metric and our loss was the L2 Penalty. Once we got the results from the Grid Search we decided to use that model to perform classification. We set a threshold based on the results we obtained from our model and decided that the result from our model would be based on whether the output we received was above or below that threshold. Based on the reports

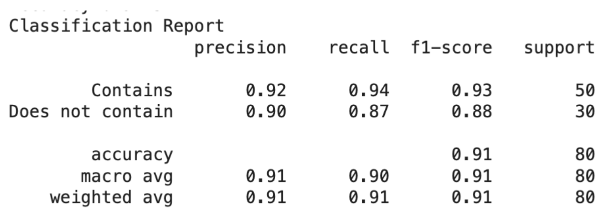
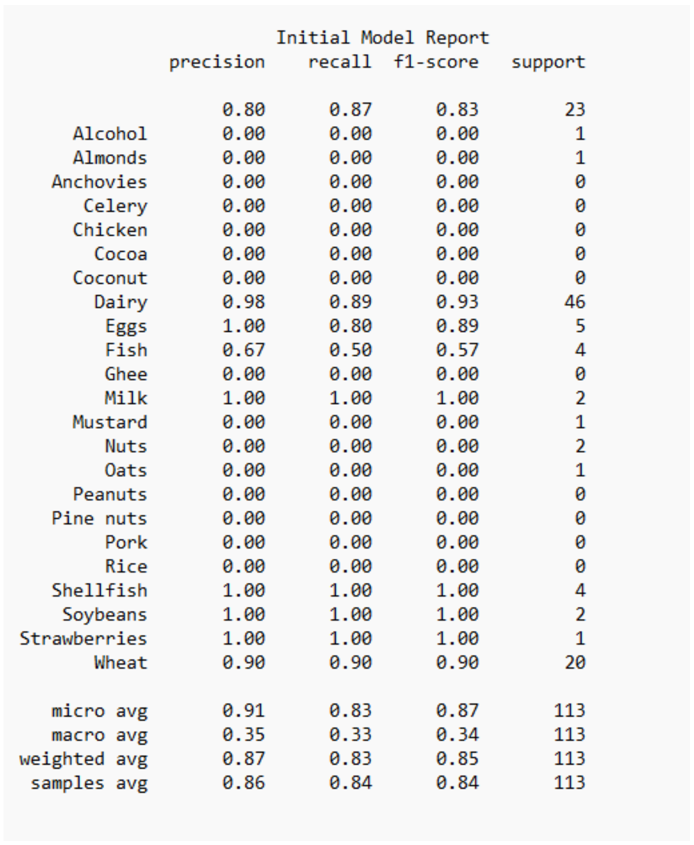

```
Classification Report
                  precision    recall  f1-score   support

        Contains       0.92      0.94      0.93        50
Does not contain       0.90      0.87      0.88        30

        accuracy                           0.91        80
       macro avg       0.91      0.90      0.91        80
    weighted avg       0.91      0.91      0.91        80
```

Fig. 3. Logistic Regression Tuned Model Report

we received after running Logistic Regression we can see that our hyper-trained model is generally better at drawing generalizations from our data. Although the changes are marginal it is worth noting that in this we are dealing with a smaller data set, so the difference in the models may be more prevalent if we increased the size of our data set. The accuracy of the two models are similar and hover between 89%-92%, Based on the amount of iterations we ran with Grid Search it is very likely that the model was able to capture the best possible C-value. You can see from the reports that there is some minor imbalance visible because the model does better at classifying foods that contain allergies than those that don't contain an allergen. This is once again explained by the fact that our data set still has some class imbalance, and SMOTE wasn't able to completely resolve these issues.

Multi-Layer Neural Network:

The goal of the Multi-Layer neural network was to see if we could outperform the Logistic Regression model. Based on the literature we read, we felt that deep learning models showed a lot of promise and we decided to use a neural network to see if it could capture features from the data.



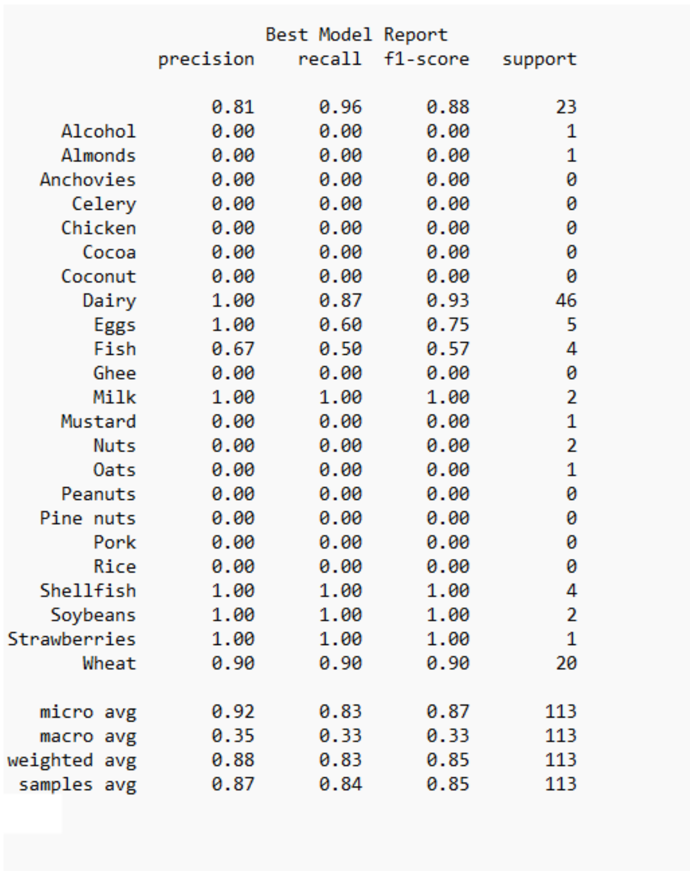
```
                    Initial Model Report
              precision    recall  f1-score   support

                   0.80      0.87      0.83        23
     Alcohol       0.00      0.00      0.00         1
     Almonds       0.00      0.00      0.00         1
   Anchovies       0.00      0.00      0.00         0
      Celery       0.00      0.00      0.00         0
     Chicken       0.00      0.00      0.00         0
       Cocoa       0.00      0.00      0.00         0
     Coconut       0.00      0.00      0.00         0
       Dairy       0.98      0.89      0.93        46
        Eggs       1.00      0.80      0.89         5
        Fish       0.67      0.50      0.57         4
        Ghee       0.00      0.00      0.00         0
        Milk       1.00      1.00      1.00         2
     Mustard       0.00      0.00      0.00         1
        Nuts       0.00      0.00      0.00         2
        Oats       0.00      0.00      0.00         1
     Peanuts       0.00      0.00      0.00         0
   Pine nuts       0.00      0.00      0.00         0
        Pork       0.00      0.00      0.00         0
        Rice       0.00      0.00      0.00         0
   Shellfish       1.00      1.00      1.00         4
    Soybeans       1.00      1.00      1.00         2
Strawberries       1.00      1.00      1.00         1
       Wheat       0.90      0.90      0.90        20

   micro avg       0.91      0.83      0.87       113
   macro avg       0.35      0.33      0.34       113
weighted avg       0.87      0.83      0.85       113
 samples avg       0.86      0.84      0.84       113
```

Fig. 4. Initial Model Report

Initial Model Feature Importance

—	Feature	Importance
117	eggs	28.375782
179	lasagna	25.040493
111	dough	23.941013
293	ratatouille	23.380836
131	flour	23.353924
252	peanut	23.240593
281	quesadilla	23.154951
97	croutons	22.877419
276	prawns	22.191948
275	prawn	22.046999

The neural network we used has 6 layers in total. The size of the input layer corresponds to the number of features in the training set of the dependent variable. The first hidden layer has 128 neurons using the ReLU activation function. Then there is a dropout layer with a dropout rate of 0.5. After the dropout layer, there is a second hidden layer with 64 neurons that also uses the ReLU activation function followed by a Dropout layer with a dropout rate of 0.5. The final layer is the output layer and has a variable number of neurons that correspond to the unique labels and use a sigmoid activation function. We used this model to run 1000 epochs to get an accuracy of 90%. Figure 3 is the untuned model report. From here we can see that it performs very similar to the hypertuned version of our Logistic Regression model. The metrics are fairly similar, however since the neural network is evaluating features the results are slightly different from what we got with the Logistic Regression model.



```
                    Best Model Report
              precision    recall  f1-score   support

                   0.81      0.96      0.88        23
     Alcohol       0.00      0.00      0.00         1
     Almonds       0.00      0.00      0.00         1
   Anchovies       0.00      0.00      0.00         0
      Celery       0.00      0.00      0.00         0
     Chicken       0.00      0.00      0.00         0
       Cocoa       0.00      0.00      0.00         0
     Coconut       0.00      0.00      0.00         0
       Dairy       1.00      0.87      0.93        46
        Eggs       1.00      0.60      0.75         5
        Fish       0.67      0.50      0.57         4
        Ghee       0.00      0.00      0.00         0
        Milk       1.00      1.00      1.00         2
     Mustard       0.00      0.00      0.00         1
        Nuts       0.00      0.00      0.00         2
        Oats       0.00      0.00      0.00         1
     Peanuts       0.00      0.00      0.00         0
   Pine nuts       0.00      0.00      0.00         0
        Pork       0.00      0.00      0.00         0
        Rice       0.00      0.00      0.00         0
   Shellfish       1.00      1.00      1.00         4
    Soybeans       1.00      1.00      1.00         2
Strawberries       1.00      1.00      1.00         1
       Wheat       0.90      0.90      0.90        20

   micro avg       0.92      0.83      0.87       113
   macro avg       0.35      0.33      0.33       113
weighted avg       0.88      0.83      0.85       113
 samples avg       0.87      0.84      0.85       113
```

Fig. 5. Best Model Report

Then we used the Optuna framework to perform hyper-parameter tuning. When we performed the hyperparameter tuning we looked into all of the different parameters that could be altered. The parameters we altered during the hyper-tuning process were the number of neurons in the hidden layers, the dropout rate in the dropout layers, the activation function of two hidden layers, and the learning rate. We also tested different activation functions for our hidden layers and the learning rate of our model. In the end, we found that the best hyperparameters for our model were: dropout-rate1: 0.21584836487802678, dropout-rate2: 0.6692543310734468, learning rate: 0.0028785544123480908, units-layer-1: 202, units-layer-2: 124, activation1: relu, activation2: relu. The hyperparameter tuning process was quite extensive we needed to make sure that a variety of parameters were being tested and that we were truly capturing the best model. There were a large amount of parameters we wanted to change and we made sure that our parameter grid accounted for this. We were able to alter the learning rates as well as the dropout rates when we performed the hyperparameter tuning. Furthermore, we used the literature that we read prior to starting this project in order to decide what arguments we should change within the parameters grid. The goal of hyperparameter tuning was to see if we could generate even better metrics from the base model, by only changing a couple of parameters within that model. When we ran the initial model, we ran around 1000 epochs to see if our results would be consistent across epochs, and in the end, by hyper-tuning our neural network we were able to get an accuracy of around 93.75%. Figure 5 is the report of the best model tuned by Optuna. In the report, the best model has higher precision and recall values on some of the classifications. That is the reason why this model can get higher accuracy than the initial model. The table below shows the most important features as determined by our model:

Best Model Feature Importance

—	Feature	Importance
117	eggs	60.963745
131	flour	46.648193
385	wrap	43.477741
327	shrimp	43.449097
97	croutons	42.760468
275	prawn	42.303341
276	prawns	42.185421
230	olives	41.294163
252	peanut	40.831917
260	pesto	40.778690

Model Comparison: By manually trying some input to the logistic regression model, we discovered that the logistic regression model is overfitting. When we input ”water, water, none, none, none”, it will return ”contain”. By trying different inputs, we discovered this model will only return ”does not contain” When the data entered is very close to the data in the data set that the report does not contain. That is because the data set is small and imbalanced. There are some outliers we discovered from the data set. A few lines of data report no allergens, but also report ”contain”. The neural network model handles more complex tasks than the logistic regression. The neural network not only can predict whether the food product contains or does not contain allergens, but also classifies which allergens are likely to be present in the food products. The best model, which is the tuned model can learn faster than the initial model. In Figure 6, we can also discover that the accuracy of the tuned model is also higher than the initial model during training. The neural network also helped us realize that there was some form of overfitting that occurred in our Logistic Regression model. We can tell from the graph at the top of this page that the models have similar performance and it is capable of providing accurate feedback.

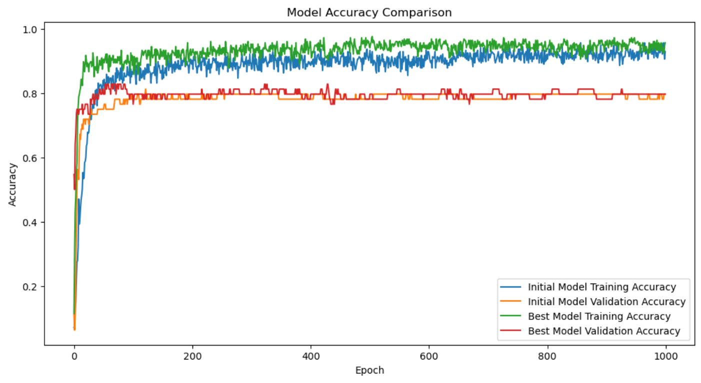


Fig. 6. Model Comparison

Software Implementation: After we developed our 3 models and performed hyper-tuning we moved on to making the web app. We used Flask and the Python backend in order to create a working web page. Flask builds both the back end and front end, and it uses HTML in order to build the front end and obtain input. Then it uses a vectorizer and sends the user input to the model to predict the values. Using the values returned it prints them on a different page using result.HTML which not only shows the result but also has a button to go back and provide another prediction. Currently, the model uses tabular data in order to detect whether an allergen is present in a specific food. The project had 3 components: the web page is what the user says, and the HTML and Python code is what is able to parse the user input and run the model. Once the model is run the output is sent back so that it can be displayed on the website. It was important that we used HTTP requests so that we could obtain the input from the website and return the result from the model back to the website. The full implementation diagram is shown below:

Based on the diagram above we had a pretty simple implementation for our web application. The most important aspect of the web application was to determine what our model would output when it would receive input it hadn’t seen in the original

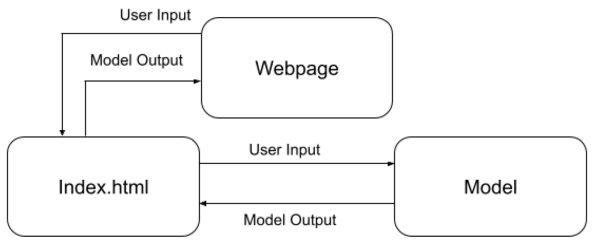


Fig. 7. Diagram of the main components of our web app

data set. The web application allowed us to test our model even more because it gave us the ability to see what the model would output when it was given a food product it hadn't seen in the training set. One thing that is worth noting is that our data set is fairly small, which is why we are thinking about performing synthetic data generation so that we can have a larger data set. In the future, we want to connect the models with image classification. Right now our input is text data, and we are hoping that eventually we can input pictures of foods into our models and it can not only determine what the food is called, but also the ingredients that are present in that food.

Conclusion and Discussion

1. Through using the Allergen Status of Food Products which contains a list of common food products and their main ingredients we were able to build a model with an accuracy of 93.75%. This model was a multi-layer neural network that used a combination of hidden and dropout layers. We felt that dropout and hidden layers helped our model because our data set was relatively small, but in the future, we expect to use synthetic data generation and collect more data so that we can create a model that is capable of performing even better generalization. Even though there were more rows that contained an allergen than didn't contain an allergen. The numbers were fairly close and both made up sizeable portions of the data set.

2. Through the use of our models, we were able to generate favorable results. Although our data set was rather small we were able to use SMOTE and hyperparameter tuning in order to optimize the model even though we didn't have the most optimal data set. Furthermore, we can also realize that when comparing the two tuned models there wasn't any significant difference. However, because we noticed overfitting within Logistic Regression we feel that the results obtained from the neural network are stronger.

3. Overall, it was very important that we cleaned our dataset and encoded it because much of our data was categorical. The use of encoders made it much easier for our model to be able to parse the input and draw useful connections with the output. The Logistic Regression and Neural Network models both performed well, both receiving an accuracy above 85%, however, we noticed that in the case of Logistic Regression specifically there were instances of overfitting most likely due to our data not being large enough. However, we were able to generate strong results from both models because of hyperparameter tuning which allowed us to test numerous combinations of parameters and compare the results we got with previous iterations. In the future, we could potentially utilize larger data sets in order to reinforce the generalizations drawn from our models.

4. In order to display the ability of the model we created a web application so that a user could enter input and then through that could see if a certain food they entered contained an allergen or not. The website collects user input and then through the use of HTML is able to call the model in the backend and determine whether the food the user entered contains an allergen. The model then returns the result which is displayed by the website. We believe the tuned neural network can serve as a strong baseline for future detection of allergens in food products.

Week	**Activity**
1 & 2	Decide which data set to use and what kind of ML technique we plan on using either Supervised or Unsupervised, Regression, etc.
3	Clean/Preprocess data. Data Analysis. Create the correlation matrix to see what factors have the greatest relationships with each other. Plot data and compare. Primarily a categorical study in this case.
4	Create the visuals based on the classification we perform and also decide if we plan on using some form of image classification paired with the original data set i.e. upload an image and from that image perform classification to determine allergens.
5	Perform classification on the image if possible to determine allergen.
6	Get the accuracy of models and begin hyper-tuning.
7	Start making deliverables ie reports, and continue hyper-tuning if possible.
8	Start a web application to display data and then create a single-page app that can use the ML model to classify.
9	Complete the report and describe how the project was made. Collect all data/visuals.
10	Final touches, literature review, proofreading, styling web app.

References

[1] Wang, Liyang et al. "A Comparative Analysis of Novel Deep Learning and Ensemble Learning Models to Predict the Allergenicity of Food Proteins." Foods (Basel, Switzerland) vol. 10,4 809. 9 Apr. 2021, doi:10.3390/foods10040809

[2] Kumar, Arun, and Prashant Singh Rana. "A deep learning based ensemble approach for protein allergen classification." PeerJ. Computer science vol. 9 e1622. 12 Oct. 2023, doi:10.7717/peerj-cs.1622

[3] B. Rohini, D. M. Pavuluri, L. Naresh Kumar, V. Soorya and J. Aravinth, "A Framework to Identify Allergen and Nutrient Content in Fruits and Packaged Food using Deep Learning and OCR," 2021 7th International Conference on Advanced Computing and Communication Systems (ICACCS), Coimbatore, India, 2021, pp. 72-77, doi: 10.1109/ICACCS51430.2021.9441800.