ANKARA UNIVERSITY
COMPUTER ENGINEERING

DietBOT

Student Number: 19290344-20290310
Student Name SURNAME: Çağrı Zengin-Mehmet Alpay
Supervisor Name SURNAME : Begüm Mutlu Bilge

Ankara date

CONTENTS

1. INTRODUCTION

1.1. Problem Definition

Aftermath of pandemic has resulted in the intake of unhealthy food. Obesity, diabetes, high blood pressure, and other health issues are all a result of this. It has become extremely fundamental for individuals to have a decent adjusted to a healthy, nutritional, and balanced diet. People are actively planning diet and fitness programs. Unfortunately, many of these programs encounter challenges when it comes to practical implementation in real life. To overcome this problem, we offer you DietBOT!

1. INTRODUCTION

1.2. Aim of the Project

The goal is to provide users with a practical and dynamic recommender that can assist in planning and adhering to health-related activities within the constraints of their daily schedule. The proposed system leverages Natural Language Processing (NLP) techniques and recommendation systems to offer tailored suggestions for meals and exercise routines. During the project, advanced applications developed in the healthcare field were analyzed, the recommendation system process was detailed, and chatbot examples were created using natural language processing techniques.

2. LITERATURE

Gao et al. (2017) introduced a computational framework for a personalized diet recommendation system. The system used Bayesian personalized ranking along with matrix factorization to learn user preferences from a wide range of data. The results indicate that the proposed feature space, when used to train forest-based models, achieves good performance and enables interpretability.

Butti Gouthami and Malige Gangappa (2020) The USDA nutrition dataset will be used to determine the user's suggested diet. A set of grocery shop information that takes the user's preferred food intake into account. The methods used in the project are Memory-based collaborative filtering,content-based filtering and clustering.

Thi Ngoc Trang Tran et al. (2021) show that their method can be used to present a comprehensive review of healthcare recommender system research: Besides, our exploration recognizes from past important outlines concentrates that it gives knowledge for suggested circumstances and approaches. In the results, forest-based models on the training data which is capable of predicting with decent performance and interpreting the healthcare-related sentences by extracting the important features used in the decision rules, ranked by their contributions, and the discriminative patterns consist of the decision rules.

2. LITERATURE

CHARLIE (A Chatbot That Recommends Daily Fitness and Diet Plans) (2023) designed to help users plan their fitness and diet goals around their schedule. The chatbot utilizes Natural Language Processing (NLP) techniques to understand user input, combines it with their calendar data, maps it with calories burnt data, and recommends personalized fitness and diet plans. A weighted scoring system was set up to determine the best recipe for each meal based on nutritional goals. Training models showed good loss patterns, with a final loss of about 0.001 for the chatbot response model and 0.1037 for the recipe sorting model.

Iigo Orue Saiz (2021) undertook an investigation to identify existing research and recommendation systems employed in major databases for the specified purpose within the last five years. The findings led to the following conclusion: Previous studies tend to emphasize recommendation systems, particularly focusing on collaborative filtering, rather than providing detailed information about data or sample descriptions. Additionally, there is uncertainty regarding the indices utilized for calculating calories or nutrients. Therefore, for feasibility, it is essential to work with openly available or well-described information, enabling the replication of experiences by different groups or, at the very least, ensuring comparability.

RECOMMENDER PART

3. MATERIAL AND METHOD

3.0. What is Recommendation System?

A recommendation system is a type of software application or algorithm designed to suggest items or content to users based on their preferences, behavior, or characteristics. These systems are commonly used in various online platforms, such as e-commerce websites, streaming services, social media, and more. Recommendation systems aim to enhance user experience by providing personalized suggestions, thereby helping users discover new items, products, or content that align with their interests and preferences. They typically leverage user data, such as past behavior, ratings, and demographics, as well as item characteristics to generate relevant recommendations.

3. MATERIAL AND METHOD

3.1. Dataset

The dataset we used in our project is adapted version of U.S. Department of Agriculture nutrition dataset for Turkish meals. Unfortunately, we could not find a dataset containing the nutritional values of Turkish food, so we decided to produce our own dataset. Every single data we collect is from Turkish National food composition database. It includes three different class according to nutrient species.

3. MATERIAL AND METHOD

1	Food_iten	Breakfast	Lunch	Dinner	VegNovVe	Calories	Fats	Proteins	Iron	Calcium	Sodium	Potassium	Carbohydı	Fibre	VitaminD	Sugars
2	Baklava	0	0	1	0	300	20	5	1	50	10	100	30	2	0	20
3	Kebab (Mi	0	0	1	0	400	25	20	2	30	20	150	15	3	0	2
4	Meze Plat	0	1	1	0	250	15	10	1	20	15	120	10	2	0	5
5	Hummus	0	1	1	1	150	10	5	1	15	5	70	10	4	0	1
6	Kofte	0	1	1	0	350	15	18	2	25	15	130	20	3	0	3
7	Pide	0	1	1	0	280	12	10	1	20	10	100	25	2	0	1
8	Manti	0	0	1	0	300	15	12	1	30	12	120	30	4	0	2
9	Menemer	1	0	1	1	200	15	8	1	20	10	80	15	2	0	5
10	Turkish De	0	0	1	0	200	0	2	0.5	10	5	80	50	1	0	40

3. MATERIAL AND METHOD

3.2. Methods

- Data Preprocessing

Separating the items into 3 groups according to their breakfast, lunch and dinner features. The separation of food items for breakfast, lunch, and dinner based on their presence in the data serves the purpose of categorizing and analyzing the user's dietary choices during different meals throughout the day. This segregation allows the program to perform specific analyses and make personalized recommendations for each meal based on the user's input and dietary preferences.

3. MATERIAL AND METHOD

```
data = pd.read_csv('meals.csv')

bfdata = data['Breakfast'].to_numpy()
ldata = data['Lunch'].to_numpy()
ddata = data['Dinner'].to_numpy()
Food_itemsdata = data['Food_items']

bfsp = []
lsp = []
dsp = []
bfspid = []
lspid = []
dspid = []

for i in range(len(Food_itemsdata)):
    if bfdata[i] == 1:
        bfsp.append(Food_itemsdata[i])
        bfspid.append(i)
    if ldata[i] == 1:
        lsp.append(Food_itemsdata[i])
        lspid.append(i)
    if ddata[i] == 1:
        dsp.append(Food_itemsdata[i])
        dspid.append(i)
```

Separating the items into 3 groups according to their breakfast, lunch and dinner features.

3. MATERIAL AND METHOD

3.2. Methods

- Data Preprocessing

Transpose data and select specific rows and columns. In the provided code, the iloc (integer-location based indexing) method is used to retrieve specific rows of meal data for breakfast, lunch, and dinner. The iloc method is a way to access rows and columns in a DataFrame by using their integer indices. Using iloc for row selection depends on the specific structure of the DataFrame and the goals of the analysis.

3. MATERIAL AND METHOD

```
bfspid_data = data.iloc[bfspid]
bfspid_data = bfspid_data.T
val = list(np.arange(5, 15))
Valapnd = [0] + val
bfspid_data = bfspid_data.iloc[Valapnd]
bfspid_data = bfspid_data.T
```

Transpose data and select specific rows and columns for Breakfast data.

3. MATERIAL AND METHOD

3.2. Methods

- User Input

The diet list given in this project is based on the food preferences, age and body mass index of the users. A food recommendation system made through Body Mass Index (BMI) can utilize individual health and dietary information to provide personalized food suggestions. Getting age, weight and height information from user. Calculating BMI and classifying users according to BMI and Age intervals.

3. MATERIAL AND METHOD

```
bmi = weight / ((height / 100) ** 2)

for lp in range(0, 80, 20):
    test_list = np.arange(lp, lp + 20)
    for i in test_list:
        if (i == age):
            agecl = round(lp / 20)

if (bmi < 16):
    clbmi = 4
elif (bmi >= 16 and bmi < 18.5):
    clbmi = 3
elif (bmi >= 18.5 and bmi < 25):
    clbmi = 2
elif (bmi >= 25 and bmi < 30):
    clbmi = 1
elif (bmi >= 30):
    clbmi = 0
```

BMI and Age intervals.

3. MATERIAL AND METHOD

3.2. Methods

- Clustering Method

In our project we use K-Means clustering method for classifying data according to their calories. K-Means clustering is a populer unsupervised machine learning algorithm used for partitioning a dataset into K distinct, non-overlapping subsets (clusters). The goal of K-means is to group similar data points into clusters, where similar is defined by proximity of data points to the mean (centroid) of the cluster.

3. MATERIAL AND METHOD

```
Dinner_calorie = Dinner_IO[1:, 1:len(Dinner_IO)]
X = np.array(Dinner_calorie)
kmeans = KMeans(n_clusters=3, random_state=0, n_init=5).fit(X)
dinner_label = kmeans.labels_
```

Ex: K-means clustering for Dinner Calorie

3. MATERIAL AND METHOD

3.2. Methods

- Preparation

This code block is preparing labeled data for three different categories, each with five repetitions. The data is being stored in separate arrays (wl_nut(weight loss nutritions), wg_nut(weight gain nutritions), and h_nut(healthy_nutritions)), and the labels are appended to each row of data. The labels come from the clbmi(Range of users' BMI) variable and creation of dataset for diet list.

3. MATERIAL AND METHOD

```
for i in range(5):
    for j in range(len(wl)):
        valloc = list(wl[j])
        valloc.append(bmicls[i])
        valloc.append(agecls[i])
        wl_nut[t] = np.array(valloc)
        if j < len(brklbl):
            yt.append(brklbl[j])
        t += 1
    for j in range(len(wg)):
        valloc = list(wg[j])
        valloc.append(bmicls[i])
        valloc.append(agecls[i])
        wg_nut[r] = np.array(valloc)
        if j < len(lnchlbl):
            yr.append(lnchlbl[j])
        r += 1
    for j in range(len(h)):
        valloc = list(h[j])
        valloc.append(bmicls[i])
        valloc.append(agecls[i])
        h_nut[s] = np.array(valloc)
        if j < len(dnrlbl):
            ys.append(dnrlbl[j])
        s += 1
```

Regularization

3. MATERIAL AND METHOD

3.2. Methods

- Classification

According to the user's preference, a diet recommendation is made using RandomForestClassifier. Random Forest Classifier is an ensemble learning algorithm commonly used for both classification and regression tasks. It operates by constructing a multitude of decision trees during training and outputs the mode (classification) or mean (regression) prediction of the individual trees for a given input. A random forest is nothing more than a series of decision trees with their findings combined into a single final result. They are so powerful because of their capability to reduce overfitting without massively increasing error due to bias.

3. MATERIAL AND METHOD

```
clf = RandomForestClassifier(n_estimators=100)

X_test = np.zeros((len(h) * 5, 9), dtype=np.float32)

for i in range(len(h)):
    valloc = list(h[i])
    valloc.append(agecl)
    valloc.append(clbmi)
    X_test[i] = np.array(valloc) * ti

X_train = h_nut[:len(ys)]
y_train = ys

clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
```

Ex:Random Forest Classifier

CHATBOT PART

3. MATERIAL AND METHOD

-Preprocess

Preprocessing operation of prompt which we desired.

Ex: Greetings prompt for intents.

```
{
  "intents": [
    {
      "tag": "greeting",
      "patterns": [
        "Hi",
        "Hey",
        "How are you",
        "Is anyone there?",
        "Hello",
        "Good day"
      ],
      "responses": [
        "Hey :-)",
        "Hello, thanks for visiting!",
        "Hi there, what can I do for you?",
        "Hi there, how can I help?"
      ]
    },
    {
      "tag": "goodbye",
      "patterns": ["Bye", "See you later", "Goodbye"],
      "responses": [
        "See you later, thanks for visiting!",
        "Have a nice day",
        "Bye! Come back again soon."
      ]
    },
```

3. MATERIAL AND METHOD

-Tokenization

Tokenization of each word in the given sentence and create a list that every tag match with patterns

```
for intent in intents['intents']:
    tag = intent['tag']
    # add to tag list
    tags.append(tag)
    for pattern in intent['patterns']:
        # tokenize each word in the sentence
        w = tokenize(pattern)
        # add to our words list
        all_words.extend(w)
        # add to xy pair
        xy.append((w, tag))
```

3. MATERIAL AND METHOD

-Stemmization and Lower case

Stemming the words to get their root form and lower case every letter. Then removing duplicated words and sorting.

```
# stem and lower each word
ignore_words = ['?', '.', '!']
all_words = [stem(w) for w in all_words if w not in ignore_words]

# remove duplicates and sort
all_words = sorted(set(all_words))
tags = sorted(set(tags))
```

3. MATERIAL AND METHOD

-TF-IDF function and Labelization

Performing Tf-idf for which already tokenized. This function firstly calculates term frequency of each word. Then, seperates them according to frequency. In the end, labelization.

```
# create training data
X_train = []
y_train = []
#tf için for döngüsü
for (pattern_sentence, tag) in xy:
    # X: term frequency for each pattern_sentence, which is already tokenized
    tf_vector = tf(pattern_sentence, all_words)
    X_train.append(tf_vector)
    # y: PyTorch CrossEntropyLoss needs only class labels(#'s for all labels), not one-hot
    label = tags.index(tag)
    y_train.append(label)
```

3. MATERIAL AND METHOD

-Preparation of Dataset

Splitting dataset as %20 test and %80 train set.

```
X_train = np.array(X_train)
y_train = np.array(y_train)
X_train, X_test, y_train, y_test = train_test_split(X_train, y_train, test_size=0.2, random_state=42, shuffle=True)
```

3. MATERIAL AND METHOD

-Model

This is our neural network model. It contains a structure consisting of three layers and ReLU as the activation function.

```python
import torch
import torch.nn as nn

#feed forward neural net with 2 hidden layers
#bag of woods is input
class NeuralNet(nn.Module):
    def __init__(self, input_size, hidden_size, num_classes, dropout_rate=0.5):
        super(NeuralNet, self).__init__()
        self.l1 = nn.Linear(input_size, hidden_size)
        self.dropout1 = nn.Dropout(p=dropout_rate)
        self.l2 = nn.Linear(hidden_size, hidden_size)
        self.dropout2 = nn.Dropout(p=dropout_rate)
        self.l3 = nn.Linear(hidden_size, num_classes)
        self.relu = nn.ReLU()

    def forward(self, x):
        out = self.l1(x)
        out = self.relu(out)
        out = self.dropout1(out)
        out = self.l2(out)
        out = self.relu(out)
        out = self.dropout2(out)
        out = self.l3(out)
        # we don't want activation and no softmax at the end because later on there will be cross entropy loss
        return out
```

3. MATERIAL AND METHOD

-Model

This model has a typical multilayer perceptron (MLP) structure. The first layer takes the input data, then passes through the hidden layers and finally produces a prediction in the output layer. We did not use activation and softmax on forward part because later there will be cross entropy loss.

3. MATERIAL AND METHOD

-Model Training

Model is trained by 1000 epochs with Adam optimizer.

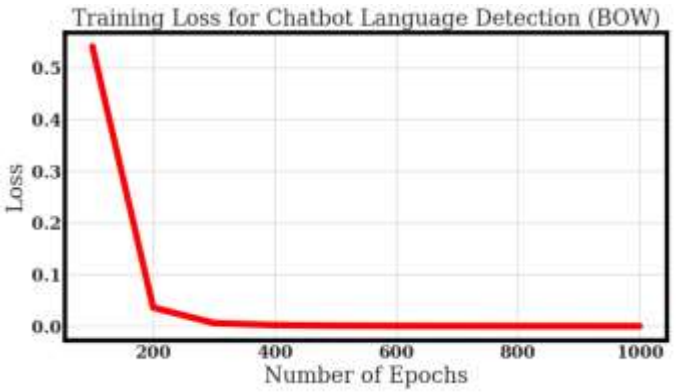
```
num_epochs = 1000
batch_size = 9
learning_rate = 0.001
input_size = len(X_train[0])
hidden_size = 9
output_size = len(tags)
```

HyperParameters

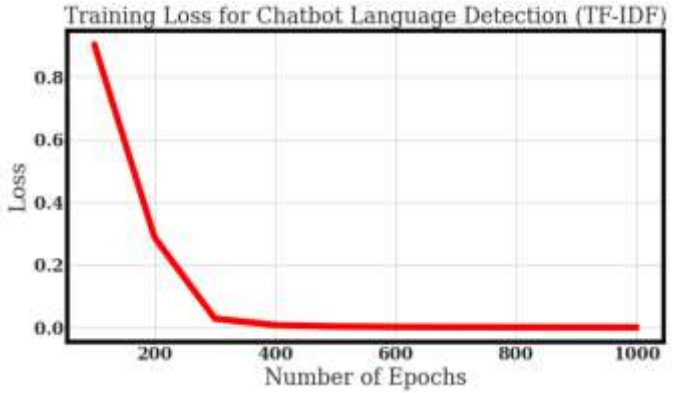
```
# Train the model
for epoch in range(num_epochs):
    total_accuracy = 0
    total_precision = 0
    total_recall = 0
    total_f1 = 0

    for (words, labels) in train_loader:
        words = torch.tensor(words, dtype=torch.float32).to(device)
        labels = torch.tensor(labels, dtype=torch.long).to(device)

        # Forward pass
        outputs = model(words)
        loss = criterion(outputs, labels)

        # Accuracy metriği
        predictions = torch.argmax(outputs, dim=1)
        batch_accuracy = accuracy_metric(predictions, labels)
        total_accuracy += batch_accuracy.item()

        # Precision, Recall, F1 Score metrikleri
        precision_metric.update(predictions, labels)
        recall_metric.update(predictions, labels)
        # f1_metric.update(predictions, labels)

        # Backward and optimize
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

    if (epoch+1) % 100 == 0:
        average_accuracy = total_accuracy / len(train_loader)
        average_precision = precision_metric.compute()
        average_recall = recall_metric.compute()
        # average_f1 = f1_metric.compute()
        f1_tensor = multiclass_f1_score(predictions, labels, num_classes=output_size)

        print("Train Results:")
        print(f'Epoch [{epoch+1}/{num_epochs}], Loss: {loss.item():.4f}, Accuracy: {average_accuracy:.4f}, Precision: {average_
        epoch_list.append((epoch+1))
        loss_list.append(round(loss.item(), 4))
```

4. EXPERIMENTS

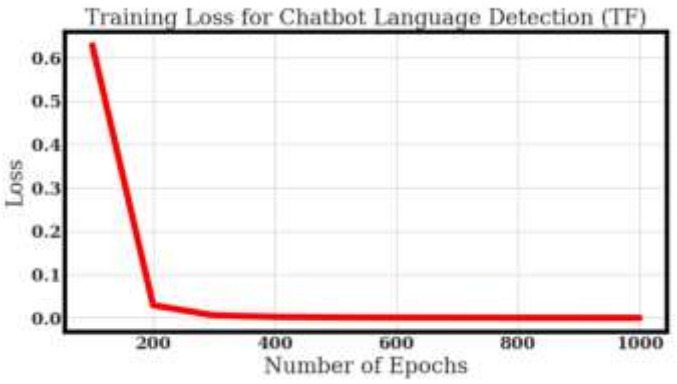
4.Metric Results



Loss-Epochs Graph (BoW)



Loss-Epochs Graph (TF-IDF)



Loss-Epochs Graph (TF)

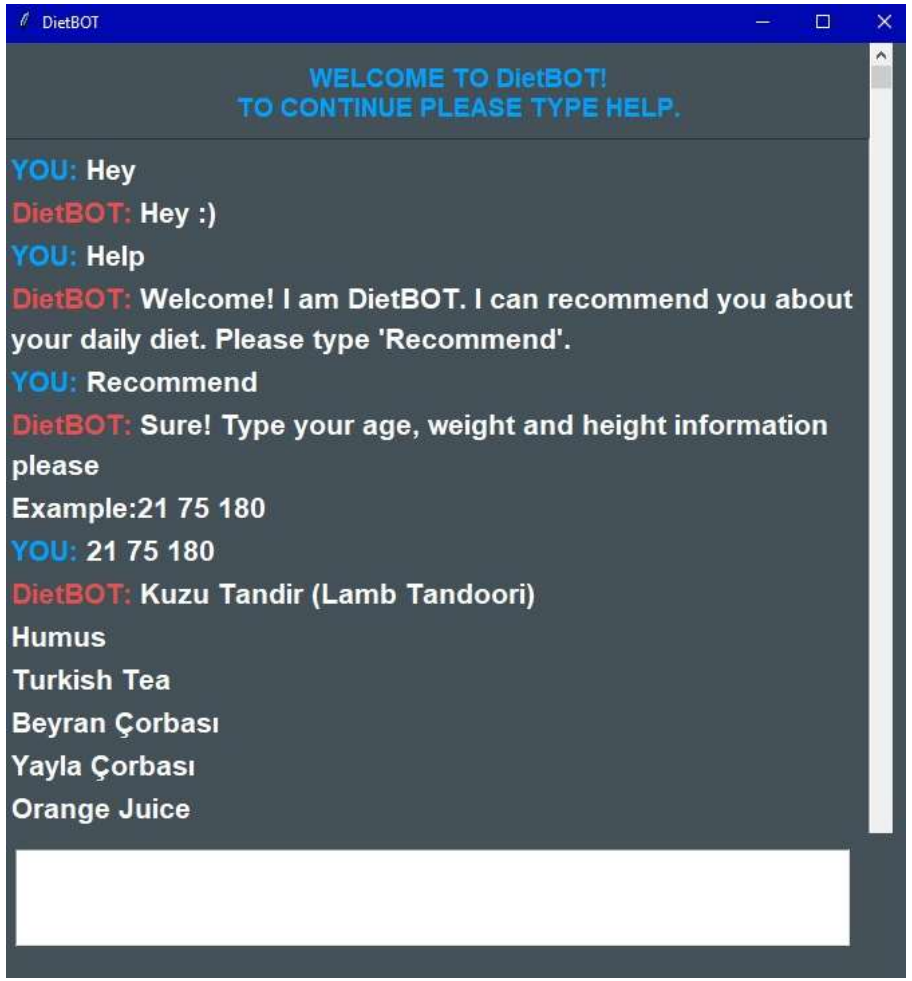
4. EXPERIMENTS

4.Metric Results

Model Name (Train Set)	Accuracy	Precision	Recall	F1
NeuralNetworkwithTFIDF	1.0	0.92	0.91	1.0
NeuralNetworkwithTF	1.0	0.94	0.93	1.0
NeuralNetworkwithBoW	0.60	1.0	0.60	0.66
RandomForestClassifier	0.90	0.97	0.90	0.90

Model Name (Test Set)	Accuracy	Precision	Recall	F1
NeuralNetworkwithTFIDF	0.83	0.80	0.70	0.73
NeuralNetworkwithTF	0.22	0.46	0.40	0.42
NeuralNetworkwithBoW	0.77	0.70	0.60	0.64
RandomForestClassifier	0.6	1.0	0.6	0.66

4. EXPERIMENTS



This is sample conversation of our DietBOT

5. CONCLUSION

-Result

Even though our project works in the end, we must express that it is not exactly what we wanted. We could not complete sections such as diseases and meal classification of breakfast, lunch and dinner for the diet recommender section, and we think that the Chatbot section is still very basic and may have an overfitting problem. In the future, we will solve these troublesome parts with talking some serious nutritionist and develop a chatbot that covers people's diseases, preferences and has more conversational functions.

REFERENCES

- Jun Gao, Ninghao Liu, Mark Lawley, and Xia Hu, “An Interpretable Classification Framework for Information Extraction from Online Healthcare Forums”, Journal of Healthcare Engineering, Cui Tao, vol. 2017, Article ID 2460174

- Thi Ngoc Trang Tran, Alexander Felfernig, Christoph Trattner and Andreas Holzinger, “Recommender systems in the healthcare domain: state-of-the-art and research issues”, Journal of Intelligent Information Systems, vol. 57, 2020, pp. 171–201, doi: 10.1007/s10844-020- 00633-6.

- Butti Gouthami and Malige Gangappa, “NUTRITION DIET RECOMMENDATION SYSTEM USING USER’S INTEREST”, International Journal of Advanced Research in Engineering and Technology (IJARET), vol. 11, 2020, pp. 2910-2919, doi: 10.34218/IJARET.11.12.2020.272

- D. Chowdhury, A. Roy, S. R. Ramamurthy and N. Roy, "CHARLIE: A Chatbot That Recommends Daily Fitness and Diet Plans," *2023 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*, Atlanta, GA, USA, 2023, pp. 116-121, doi: 10.1109/PerComWorkshops56833.2023.10150359.

REFERENCES

- A. S. Fauziah, F. Renaldi and I. Santikarama, "Combining Medical Records, Daily Diets, and Lifestyle to Detect Early Stage of Diabetes using Data Mining Random Forest Method," *2022 IEEE Creative Communication and Innovative Technology (ICCIT)*, Tangerang, Indonesia, 2022, pp. 1-5, doi: 10.1109/ICCIT55355.2022.10118711.

- R. Sajith, C. Khatua, D. Kalita and K. B. Mirza, "Nutrient Estimation from Images of Food for Diet Management in Diabetic Patients," *2023 World Conference on Communication & Computing (WCONF)*, RAIPUR, India, 2023, pp. 1-6, doi: 10.1109/WCONF58270.2023.10235177.

- W. Zhou and X. Peng, "Linear Programming Method and Diet Problem," *2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI)*, Changchun, China, 2022, pp. 761-764, doi: 10.1109/ICETCI55101.2022.9832221.

- C. A. Iheanacho and O. R. Vincent, "Classification and recommendation of food intake in West Africa for healthy diet using Deep Learning," *2022 5th Information Technology for Education and Development (ITED)*, Abuja, Nigeria, 2022, pp. 1-6, doi: 10.1109/ITED56637.2022.10051387.

ANKARA UNIVERSITY
COMPUTER ENGINEERING

THANK YOU

Student Number: 19290344-20290310
Student Name SURNAME: Çağrı Zengin-Mehmet Alpay
Supervisor Name SURNAME : Begüm Mutlu Bilge