Chair in
Information and Service Systems

Saarland University

Information and Service Systems (ISS)

# Project 3 Nutrition and immune system

*Authors:*
Abdallah Bashir 2577831
Yogesh Kumar Baljeet Singh 2576034
Hassan Soliman 2576774
Rayhanul Islam Rumel 2576541

***Chair:*** Univ.-Prof. Dr.-Ing. Wolfgang Maaß Chair in Information and Service Systems, Campus A5 4, 66123 Saarbrücken

Date of submission
Jul 17th

# Contents

## 1 Introduction

COVID-19 is a recently discovered coronavirus-caused infectious disease. Most people diagnosed with the COVID-19 virus will develop mild to moderate respiratory illness and recover without any medical care being needed. Using Machine Learning, We try to analyze the food consumption from countries all over the world to see whether there's a relationship between given country food habits and their recovery rate.

## 2 Dataset

### 2.1 Data Explanation

Data is divided into four datasets, they all have the same 170 rows, which are all countries, and same columns, which are 23 different food categories and COVID-19 statistics for each country. All four datasets share the same values of the columns of COVID-19 statistics for each country, the difference between them is explained as follows:-

- Fat Supply Quantity Data:- for each country, it demonstrates the fat consumption percentages distributed amongst all food categories.
- Food Supply Quantity in Kg:- for each country, it demonstrates the food supply quantity consumption percentages distributed amongst all food categories.
- Protein Supply Quantity Data:- for each country, it demonstrates the protein consumption percentages distributed amongst all food categories.
- Food Supply Data in Kcal:- for each country, it demonstrates the energy intake consumption percentages distributed amongst all food categories.

## 3 Exploratory data analysis (EDA)

To start with, we first ranked the countries based on their recovery rate (descending order) for all the datasets. Then we came up with this idea to consider the countries which had Total Number of Confirmed Cases greater than 125,000. The reason behind this threshold is to get variations in the recovery rate, as most of the countries with lower confirmed cases had a very high recovery rate which was clustered around 95%. After ranking countries based on their recovery rate, each team member focused on comparing two pairs of countries (two countries with a high & low recovery rate). So, in total, we had 16 countries to compare for all the 4 datasets.

Due to the length constraints, we would like to compare only one pair of countries (Bangladesh - Mexico) for a single dataset (Food supply in Fat). Comparisons can be found in Figure 3 and Figure 4

Bangladesh with more recovery rates, consumes more fat from the following sources: Vegetal Products, Vegetable Oils, Animal Products, Cereals, and Milk. These food sources seem to have a positive impact on the recovery rate which leads this country to have a better recovery rate than Mexico.

Mexico, on the other hand with a low recovery rate, consumes around 30% less Vegetal Products, around 40% less Vegetable Oils, 2.5 times more Animal Products, 30% more milk than Bangladesh for their fat intake as their food habits are different from Bangladesh.

We used a similar approach for all the comparisons and then we proceeded to model the datasets so as to compare the results from EDA and ML models.

# 4 Procedure

## 4.1 What to Model!

We want to investigate the different food categories consumption rates and their significance regarding COVID-19 recovery rates. The task here is a regression problem where input is all food categories percentages, and the output is the recovery rare. Thus why the loss function is Root mean square error. The main motivation behind choosing this type of modeling is that we can detect the important features "food categories" regarding predicting recovery rate using feature importance techniques. We split the data into train and test datasets.

## 4.2 Modeling

We choose multiple variants of the Decision trees family for each dataset to try to predict the Recovery rate. We then bench-marked the results for all models and choose the best model for each given dataset. The models we choose were:

- Decision Tree.
- Random Forest.
- Gradient Boosting.
- Ada-boost with Decision Trees.

We choose the Decision Tree models family because it suits our small size of the data the best. Another important factor was the running time considering we had to run a lot of experiments for the four models and each of the four datasets.

### 4.2.1 Feature importance permutation

- Default Feature Importance:- it works by calculating the mean decrease in impurity cased by each feature, then rank them in descending order.

- Permutation Feature Importance:- It works by calculating the difference between the RMSE of the model after permutating the values of one feature, and the RMSE of the baseline model without any permutation, do that for each feature and rank them in descending order.

We adopted Permutation Feature Importance approach due to higher reliability (Lewinson).

### 4.2.2 Hyper parameter tuning

To choose the best Hyper-parameters for any given model we used both Random and Grid Search. At first we use random search to narrow the search region, then we forward that region to Grid Search to return the best combination of Hyper-parameters. This procedure helped us by decreasing the training time and reduced the test error as well.

## 5 Results and Discussion

Table 1 shows the four models test error (RMSE) results on the four datasets. As expected, GradBoost and Random Forest topped the list of the models with minimum test errors.

After listing the most significant features found after modeling across the Validation data which is the countries with the highest and the lowest recovery rate that have been analysed in the EDA section, we came to the following table which describes about each dataset's distinguishing features. We separated the features as Good and Bad by taking the maximum number of its occurrences in high and low recovery rate countries respectively.

Based on the information from Table 2, we would like to recommend that to have a healthy immune system amidst this COVID-19 pandemic, that countries manufacture or import and feed more of Vegetal Products, cereals, Starchy Roots, Alcoholic Beverages, Sugar and Sweeteners, and Meat. Also, the countries should try limiting their citizen's food intake of Animal Products, Animal Fats, Fruits - Excluding Wine, Pulses, Treenuts, Fish, Vegetal Oils Vegetables, Eggs and Oilcrops to a minimum, as they have a negative impact on the immune system which is directly linked to the recovery rate of the Country.

However we had some limitation in our study. There are some features which we couldn't classify e.g. Milk, stimulants as it appeared as Good feature in one dataset and Bad in another dataset. This can be used as a pre-cursor to work on Individual's immune system analysis (if such a dataset is developed in the near future), which would then in turn help in verifying the claims that we make in this report about what kind of food makes a COVID-19 patient should intake to boost his immune system to fight against this virus along with more insights of these features.

| ***model/dataset*** | Fat | Protein | KG Quantity | Kcal Quantity |
|---|---|---|---|---|
| Decision Tree | 13.51 | 14.35 | 5.76 | 14.35 |
| Adaboost | 15.84 | 8.91 | 6.9 | 8.91 |
| Gradboost | 16.96 | **5.14** | **5.23** | **5.43** |
| Random Forest | **8.98** | 8.13 | 8.01 | 8.75 |

**Table 1:** QDA confusion matrix

| Dataset | Good | Bad |
|---|---|---|
| Protein Supply | *Alcoholic Beverages<br>*Starchy Roots<br>*Vegetal Products | *Animal Products<br>*Pulses<br>*Stimulants |
| Fat Supply | *Vegetal Products<br>*Cereals - Excluding Beer<br>*Meat | *Animal Fats<br>*Fruits - Excluding Wine<br>*Pulses<br>*Treenuts<br>*Eggs<br>*Milk- Excluding Butter<br>*Fish<br>*Animal Products |
| Kcal Supply | *Milk - Excluding Butter | *Fish<br>*Seafood<br>*Fruits - Excluding Wine<br>*Animal Fats<br>*Eggs<br>*Vegetal Oils |
| KG Supply | *Sugar and Sweeteners<br>*Stimulants<br>*Meat | *Vegetables<br>*Eggs<br>*Oilcrops<br>*Animal Fats<br>*Fruits - Excluding Wine |

**Table 2:** Significant Features in each dataset

# A Appendix

All code and more elaborated notebooks are available in our repo.

## A.1 Data Collection and Filtering

### A.1.1 Data Origin and Sources

Data is attributed to (3). Data comes from three sources as follows:-

- FAO (5):- It has data for different food categories and their consumption percentages for each country.
- Johns Hopkins Center for Systems Science (4):- It has data for the different confirmed, deaths and recovered rates of COVID-19.
- Population Reference. Bureau (1):- It has data for population numbers for each country.

### A.1.2 Data Preprocessing

In this regard, we wanted to calculate the recovery rate for each country using the statistics we already have in the datasets as follows:-

$$Recovery\ Rate = \frac{Recovered}{Confirmed - Active}$$

Then Countries are arranged descendingly according to the newly added column "Recovery Rate" In addition, we needed to calculate the total confirmed number of cases as follows:-

$$Total\ Confirmed = \frac{Confirmed}{100} * Population$$

Few countries have missing confirmed rates, so their total confirmed number of cases is fetched using official statistics (2). Finally, for some countries, there is 'NaN' values in two columns, which are then interpolated using linear regression.

### A.1.3 Threshold on considering Data points

Our problem domain was to recommend better dietary systems for individuals. of course this would have required further analysis for individual cases and study their reaction towards different food groups. Unfortunately the given data analyse food consumption based on countries and we could not find any study or data that was based on individuals food consumption and their COVID-19 status.

Basing our analysis purely on the given data which only considers countries food consumption will be built on the assumption that a given country consuming this amount of these food groups will lead to a better or worse recovery rate. Taking this assumption alone into consideration is wrong. A country might have a bad policy in fighting the COVID-19 pandemic, for e.g late response with wearing masks or applying lock-down. Taking this into mind we decided to include only countries with high portions on confirmed cases, since we will be building our analysis on recovery rate. we noticed that countries with low total confirmed cases always have recovery rate near 100% which entails that it is very early to judge whether this country behaviour in food consumption actually helps in fighting COVID-19. Figure 1 and 2 shows the top 10 and lower 10 countries in total confirmed cases numbers, we can see that after applying the thresh-hold of taking countries above 5000 cases: 82 countries out of total 170 remained after applying the filter and The minimum of Total Confirmed' cases is 5017 while it's zero before applying the threshold.

## A.2 EDA

We used EDA to approach and analyse datasets to summarize the main characteristics using visual graphs as described in the functions below:

### A.2.1 Code

```
#Calculate recovery rate of each country

def prepare_rec_rate_and_total_confirmed(df):
    df['Rec_Rate'] = df[['Confirmed', 'Active', 'Recovered']].
    apply(lambda x: (x['Recovered']/(x['Confirmed']-x['Active']))*100,
    axis=1)
    df = df.sort_values(by=['Rec_Rate'], ascending=False)

    df['Total Confirmed'] = df[['Confirmed', 'Population']].
    apply(lambda row: (row['Confirmed']/100)*row['Population'], axis=1)

    return df

#Returns filtered data with respect to country:

def prepare_data_for_each_country(country):

    df_kg_country = df_kg[df_kg["Country"] == country]
    df_kcal_country = df_kcal[df_kcal["Country"] == country]
    df_fat_country = df_fat[df_fat["Country"] == country]
    df_protein_country = df_protein[df_protein["Country"] == country]

    return df_kg_country, df_kcal_country,
    df_fat_country, df_protein_country

#Returns filtered data with respect to features:

def prepare_data_for_each_country_for_plotting(df_kg, df_kcal, df_fat,
df_protein):


    df_column_names=list(df_kg)[1:24]
    df_kg_percentages=df_kg.iloc[:,1:24]
    df_kcal_percentages=df_kcal.iloc[:,1:24]
    df_fat_percentages=df_fat.iloc[:,1:24]
    df_protein_percentages=df_protein.iloc[:,1:24]

    return df_column_names, df_kg_percentages, df_kcal_percentages,
    df_fat_percentages, df_protein_percentages

# Returns data for a pair of countries for plotting:

def get_data_for_two_countries(c1, c2):

    df_kg_filtered = df_kg[df_kg["Country"].isin([c1,c2])]
    df_kcal_filtered = df_kcal[df_kcal["Country"].isin([c1,c2])]
    df_fat_filtered = df_fat[df_fat["Country"].isin([c1,c2])]
    df_protein_filtered = df_protein[df_protein["Country"].isin([c1,c2])]

    df_kg_c1, df_kcal_c1, df_fat_c1, df_protein_c1 =
    prepare_data_for_each_country(c1)

    df_kg_c2, df_kcal_c2, df_fat_c2, df_protein_c2 =
    prepare_data_for_each_country(c2)

    df_c1_column_names, df_kg_c1_percentages, df_kcal_c1_percentages,
    df_fat_c1_percentages, df_protein_c1_percentages =
    prepare_data_for_each_country_for_plotting(df_kg_c1,
    df_kcal_c1, df_fat_c1, df_protein_c1)
    df_c2_column_names, df_kg_c2_percentages,
    df_kcal_c2_percentages, df_fat_c2_percentages,
    df_protein_c2_percentages =
    prepare_data_for_each_country_for_plotting(df_kg_c2, df_kcal_c2,
    df_fat_c2, df_protein_c2)


    return df_kg_filtered, df_kcal_filtered,
    df_fat_filtered,  df_protein_filtered, df_kg_c1,
    df_kcal_c1, df_fat_c1, df_protein_c1, df_kg_c2,
    df_kcal_c2, df_fat_c2, df_protein_c2,
    df_c1_column_names, df_kg_c1_percentages,
    df_kcal_c1_percentages, df_fat_c1_percentages,
    df_protein_c1_percentages, df_c2_column_names,
    df_kg_c2_percentages, df_kcal_c2_percentages,
    df_fat_c2_percentages, df_protein_c2_percentages

# For drawing Scattered plots:

def draw_scatter_plot(df, title):

    return (ggplot(df)
     + aes(x='Country', y = 'Rec_Rate', color='Country')
     + p9.geom_point(size = 7)
     + labs(title=title, x='Countries', y='Recovery Rate')
     + geom_text(aes(x="Country", y="Rec_Rate", label='Rec_Rate'),
                 nudge_y=2.7,
         stat='identity',
         format_string='{:.1f}% ')
    )

# For drawing Pie chart plot:

def draw_pie_chart(df_percentages, df_column_names, title):


    explode = (0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,
    0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05,0.05)

    from random import randint
    color = []
    n = 23
    for i in range(n):
        color.append('#%06X' % randint(0, 0xFFFFFF))

    plt.pie(df_percentages, labels=df_column_names, labeldistance=1.05,
    autopct='%1.1f%%', startangle=10, pctdistance=0.85,
    explode = explode, colors=colors_list_filtered)
    centre_circle = plt.Circle((0,0),0.70,fc='white')
    fig = plt.gcf()
    fig.gca().add_artist(centre_circle)
    plt.axis('equal', radius=1200)
    plt.legend(df_column_names, title="Food Categories",
             loc="right", bbox_to_anchor=(1.5, 0.5))
    plt.title(title, pad=20, fontsize= 20)
    plt.figure(figsize=[30, 30])
    plt.tight_layout()
    plt.rcParams.update({'font.size': 12})
    fig_size = plt.rcParams["figure.figsize"]
    fig_size[0] = 8
    fig_size[1] = 8
    plt.rcParams["figure.figsize"] = fig_size
    plt.show()

# For drawing Bar Chart plot:

def draw_bar_chart(df_percentages, df_column_names, title):

    df_percentages_list = df_percentages.values.tolist()
    df_percentages_list_flattened =
    list(np.concatenate(df_percentages_list).flat)

    demo = pd.DataFrame({"x-axis": df_column_names,
                         "y-axis": df_percentages_list_flattened})

    return ggplot(data=demo) +\
    geom_bar(mapping=aes(x="x-axis", y="y-axis", fill='x-axis'),
    stat="identity", size=20) +\
    coord_flip() +\
    labs(title=title, x='Food Categories', y='Percentages') +\
    geom_text(
        aes(x="x-axis", y="y-axis", label='y-axis'),
        stat='identity',
        nudge_y=2.3,
        format_string='{:.1f}% '
    )

# Plots cross-correlations of all features against one another
def correlation(df, title):
    sns.set(font_scale=1)
    corr = df.corr()

    mask = np.triu(np.ones_like(corr, dtype=np.bool))
    f, ax = plt.subplots(figsize=(11, 9))
    cmap = sns.diverging_palette(20, 220, n=200)
    sns.heatmap(corr, mask=mask, cmap=cmap, vmax=1, vmin=-1, center=0,
                square=True, linewidths=.5, cbar_kws={"shrink": .5})
    ax.set_title(title, pad=20, fontsize= 20)

# Plots correlations of all features against Recovery rate:

def correlation_values(df, title):

    df_correlation = df.corr()
    cmap = sns.diverging_palette(20, 220, n=200)
    # Set up the matplotlib figure
    f, ax = plt.subplots(figsize=(11, 9))
    sns.heatmap(df_correlation[['Rec_Rate']].sort_values(by=['Rec_Rate'],
                vmin=-1,
                vmax=1, center=0,
                cmap=cmap,
                annot=True,linewidths=1.5, cbar_kws={"shrink": .5})
    ax.set_title(title, pad=20, fontsize= 20)
```

## A.3 Modeling

We have built **class** `Model` as the main modeling class. We just need to pass it the name of the data-set. for e.g `Model("fat")`. the main function in the class is `best_ml_approach()` which takes the output of the four models and return the model with the minimum Test error along with the graph of features permutation importance.

### A.3.1 Code

```python
class Model:
    def __init__(self, data):

        self.df = self.clean_data(datasets[data])
        self.data = train_test_split(
            self.df.loc[:, "Alcoholic Beverages":"Vegetables"],
            self.df["Rec_Rate"],
            test_size=0.2,
            random_state=100,
        )
        self.best_ml_appraoch(self.df)

    def clean_data(self, path):
        """
        preprocess the data
        """

        return data

    def plot_feature_importance(self, df, importance):
        """
        plots the most significant features
        """

    def _show_on_single_plot(self, ax):
        """
        annotate the graph with confirmed cases numbers for each country
        """
"""
next four functuions return the results on test data for any given datase
"""
    def decision_tree(self):

    def adaboost(self, tree):

    def gradboost(self):

    def random_forest(self):
```

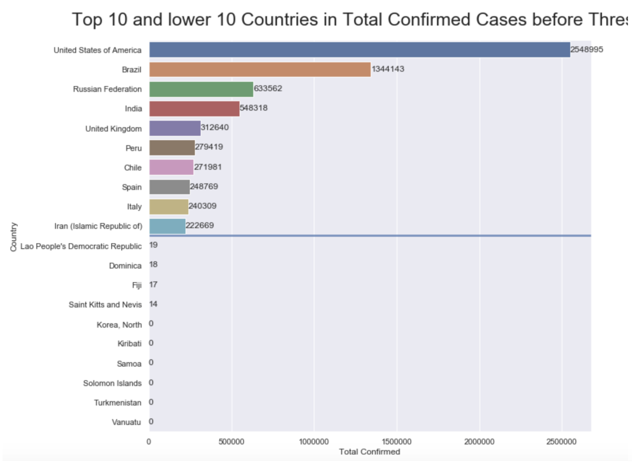
**Figure 1:** Before Threshold

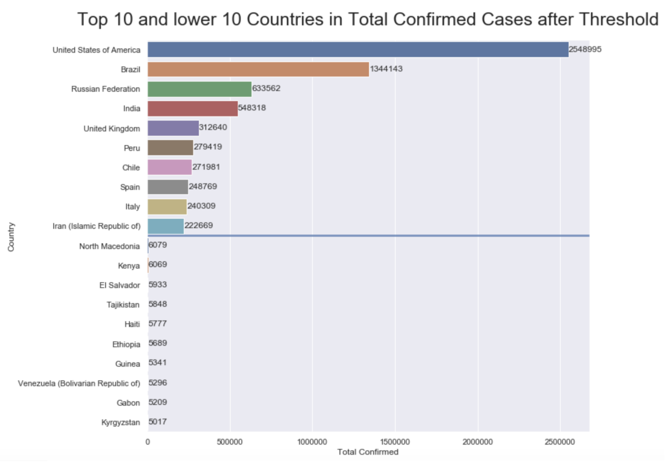
**Figure 2:** After Threshold

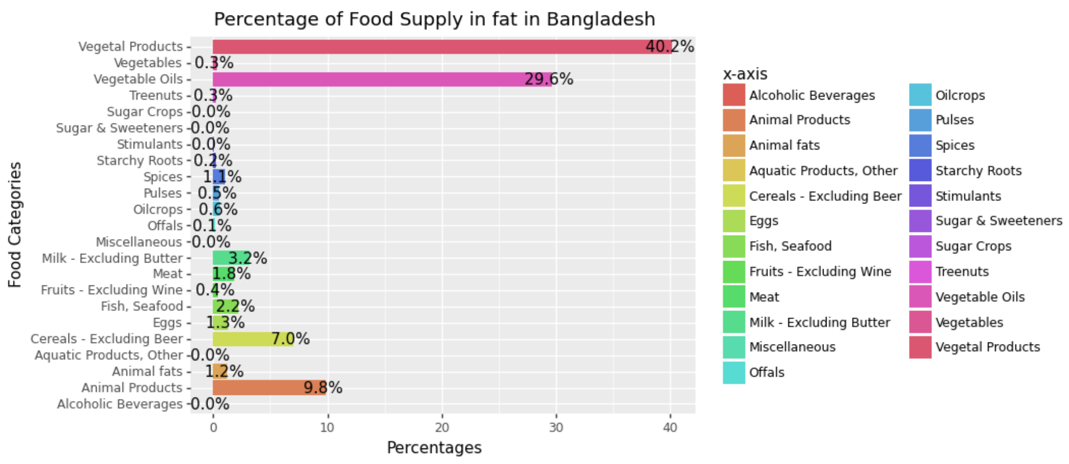
**Figure 3:** Food Supply in Fat for Bangladesh

```
def best_ml_appraoch(self, df):
    """
    return the best "minimum" test error of:
    * Decision Tree
    * Adaboost
    * Random Forest
    * Gradient Boost
    """
```

---

## A.4 Figures

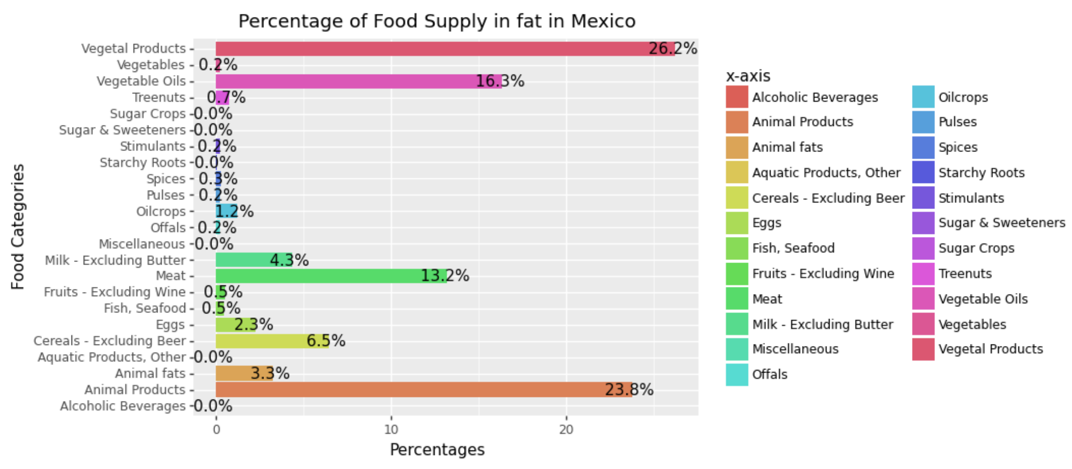

**Figure 4:** Food Supply in Fat for Mexico

## B Declaration of Authorship

I affirm that I have produced the work independently, that I have not used any aids other than those specified and that I have clearly marked all literal or analogous reproductions as such.

**Location: Saarbruecken, Germany, Date: 14th July 2020**

Bashir, Abdallah: [illegible]

Baljeet Singh, Yogesh Kumar: [illegible]

Rumel, Rayhanul Islam: Rumel

Soliman, Hassan: Hassan Soliman

## References

[1] 2019 world population data sheet – population reference bureau. Library Catalog: www.prb.org.

[2] Coronavirus update (live): 13,306,035 cases and 577,205 deaths from COVID-19 virus pandemic - worldometer. Library Catalog: www.worldometers.info.

[3] COVID-19 healthy diet dataset. Library Catalog: www.kaggle.com.

[4] COVID-19 map. Library Catalog: coronavirus.jhu.edu.

[5] Food consumption database. Library Catalog: www.fao.org.

[Lewinson] Lewinson, E. Explaining feature importance by example of a random forest. Library Catalog: towardsdatascience.com.