ZOMBIE DETECTOR: Using ML To Save Lives During A Zombie Apocalypse

Mini Project documentation submitted to

JAWAHARLAL NEHRU TECNOLOGICAL UNIVERSITY, HYDERABAD

In partial fulfillment of the requirements for the award of the degree of

BACHELOR OF TECHNOLOGY

In

COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

Submitted By

AKHIL BOLLABOINA	**(21UK1A6750)**
MANIKARNA JAMPALA	**(21UK1A6714)**
PAVANI JAKKULA	**(21UK1A6713)**
SRUTHI KATKURI	**(21UK1A6761)**

Under the guidance of

DR REKHA GANGULA

Asst.Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

VAAGDEVI ENGINEERING COLLEGE

Affiliated to JNTUH, HYDERABAD

BOLLIKUNTA, WARANGAL (T.S) – 506005

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

(DATA SCIENCE)

VAAGDEVI ENGINEERING COLLEGE WARANGAL

CERTIFICATE

This is to certify that the Mini project report entitled "**ZOMBIE DETECTOR: Using ML To Save Lives During A Zombie Apocalypse**" is being submitted by **AKHIL BOLLABOINA(21UK1A6750), MANIKARNA JAMPALA(21UK1A67), PAVANI JAKKULA(21UK1A67), SRUTHI KANDURI (21UK1A6762)**, partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Computer Science & Engineering to Jawaharlal Nehru Technological University Hyderabad during the academic year 2024-2025.

Project Guide	**Head of the Department**
G.REKHA	**Dr. K. SHARMILAREDDY**
(Asst.professor)	**(Professor)**

ACKNOWLEDGEMENT

We wish to take this opportunity to express our sincere gratitude and deep sense of respect to our beloved **Dr. P. Prasad Rao**, Principal, Vaagdevi Engineering College for making us available all the required assistance and for his support and inspiration to carry out this mini project in the institute.

We extend our heartfelt thanks to **Dr. K. SHARMILAREDDY,** Head of the Department of CSD, Vaagdevi Engineering College for providing us necessary infrastructure and there by giving us freedom to carry out the mini project.

We express heartfelt thanks to the guide **G.REKHA**, Asst. Professor, Department of CSD for her constant support and giving necessary guidance for completion of this mini project.

Finally, we express our sincere thanks and gratitude to our family members, friends for their encouragement and outpouring their knowledge and experiencing throughout thesis.

AKHIL BOLLABOINA	**(21UK1A6750)**
MANIKARNA JAMPALA	**(21UK1A6714)**
PAVANI JAKKULA	**(21UK1A6713)**
SRUTHI KATKURI	**(21UK1A6761)**

ZOMBIE DETECTOR: Using ML To Save Lives During A Zombie Apocalypse

ZOMBIE DETECTOR: Using ML To Save Lives During A Zombie Apocalypse

News reports suggest that the impossible has become possible…zombies have appeared on the streets! What should we do? The Centers for Disease Control and Prevention (CDC) zombie preparedness website recommends storing water, food, medication, tools, sanitation items, clothing, essential documents, and first aid supplies. The CDC analysts will be prepared, but it may be too late for others! The purpose of this project is to develop a machine learning model that can predict the likelihood of a person turning into a zombie based on their age, sex, location, and available supplies. The project simulates a zombie outbreak and focuses on identifying supplies that are associated with safety during such an event. The Flask app developed as part of the project allows users to input their own data and get a prediction in real-time.

In summary, the purpose of this project is to use machine learning to identify supplies that are associated with safety during a zombie outbreak, raise awareness about emergency preparedness, and assist emergency responders in making informed decisions.

Technical Architecture

Define Problem / Problem Understanding

Data Collection & Preparation

ML depends heavily on data. It is the most crucial aspect that makes algorithm training possible. So, this section allows you to download the required dataset.

Collect The Dataset

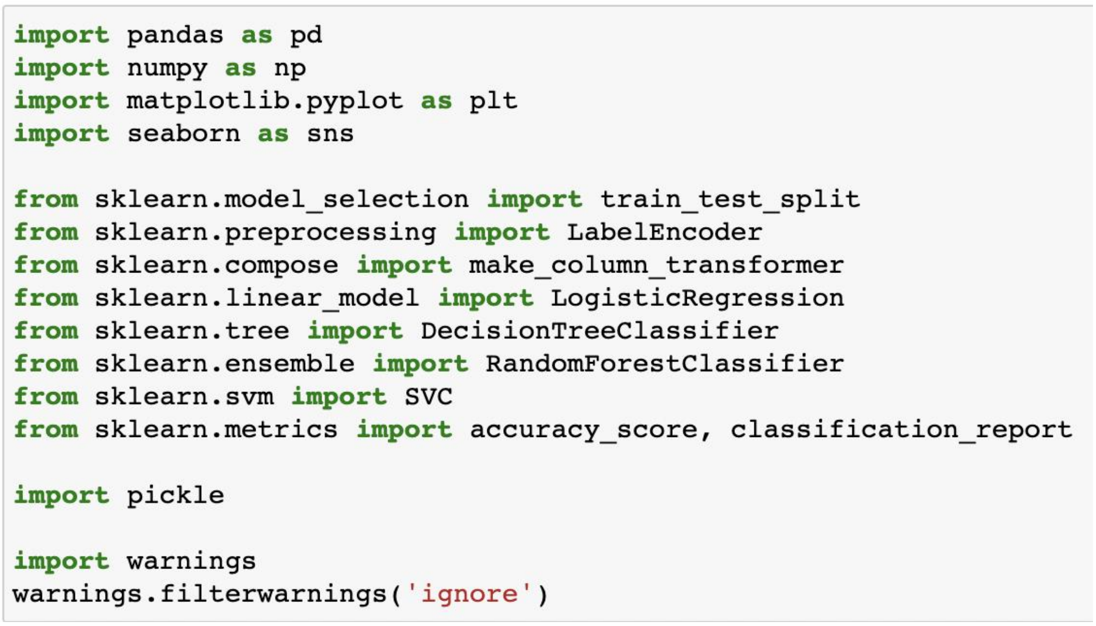
There are many popular open sources for collecting the data. Eg: kaggle.com, UCI repository, etc.
In this project, we have used .csv data. This data is downloaded from kaggle.com. Please refer to the link given below to download the dataset.

Link: https://www.kaggle.com/datasets/kingabzpro/zombies-apocalypse

As the dataset is downloaded. Let us read and understand the data properly with the help of some visualisation techniques and some analysing techniques.
Note: There are a number of techniques for understanding the data. But here we have used some of it. In an additional way, you can use multiple techniques.

Importing the libraries

Import the necessary libraries as shown in the image.

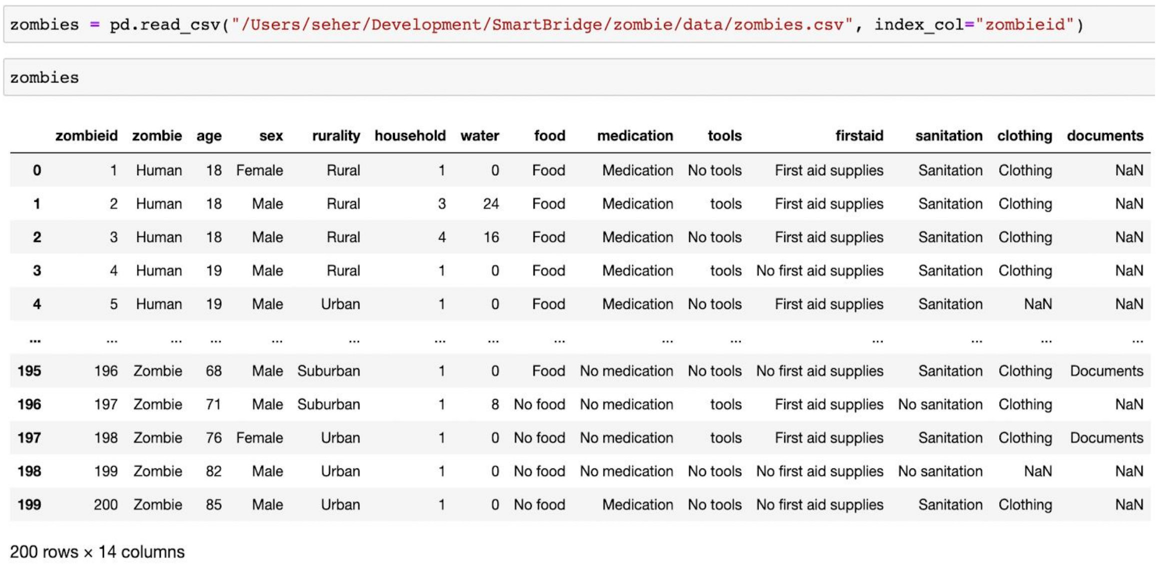


```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from sklearn.compose import make_column_transformer
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report

import pickle

import warnings
warnings.filterwarnings('ignore')
```

Read the Dataset

Our dataset format might be in .csv, excel files, .txt, .json, etc. We can read the dataset with the help of pandas.

In pandas, we have a function called read_csv() to read the dataset. As a parameter, we have to give the directory of the csv file.



```
zombies = pd.read_csv("/Users/seher/Development/SmartBridge/zombie/data/zombies.csv", index_col="zombieid")
```

```
zombies
```

	zombieid	zombie	age	sex	rurality	household	water	food	medication	tools	firstaid	sanitation	clothing	documents
0	1	Human	18	Female	Rural	1	0	Food	Medication	No tools	First aid supplies	Sanitation	Clothing	NaN
1	2	Human	18	Male	Rural	3	24	Food	Medication	tools	First aid supplies	Sanitation	Clothing	NaN
2	3	Human	18	Male	Rural	4	16	Food	Medication	No tools	First aid supplies	Sanitation	Clothing	NaN
3	4	Human	19	Male	Rural	1	0	Food	Medication	tools	No first aid supplies	Sanitation	Clothing	NaN
4	5	Human	19	Male	Urban	1	0	Food	Medication	No tools	First aid supplies	Sanitation	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
195	196	Zombie	68	Male	Suburban	1	0	Food	No medication	No tools	No first aid supplies	Sanitation	Clothing	Documents
196	197	Zombie	71	Male	Suburban	1	8	No food	No medication	tools	First aid supplies	No sanitation	Clothing	NaN
197	198	Zombie	76	Female	Urban	1	0	No food	No medication	tools	First aid supplies	Sanitation	Clothing	Documents
198	199	Zombie	82	Male	Urban	1	0	No food	No medication	No tools	No first aid supplies	No sanitation	NaN	NaN
199	200	Zombie	85	Male	Urban	1	0	No food	Medication	No tools	No first aid supplies	Sanitation	Clothing	NaN

200 rows × 14 columns

Zombies Apocalypse | Kaggle..

To figure out which supplies are associated with safety in Zombie Apocalypse..

https://www.kaggle.com/datasets/kingabzpro/zombies-apocalypse

Data Preparation

As we have understood how the data is, let's pre-process the collected data.

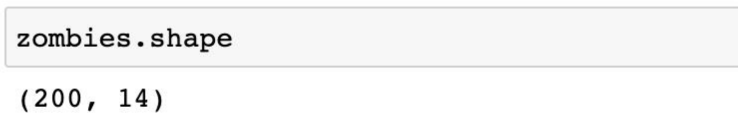
The download data set is not suitable for training the machine learning model as it might have so much randomness so we need to clean the dataset properly in order to fetch good results. This activity includes the following steps.

- Handling missing values
- Handling categorical data

Note: These are the general steps of pre-processing the data before using it for machine learning. Depending on the condition of your dataset, you may or may not have to go through all these steps.

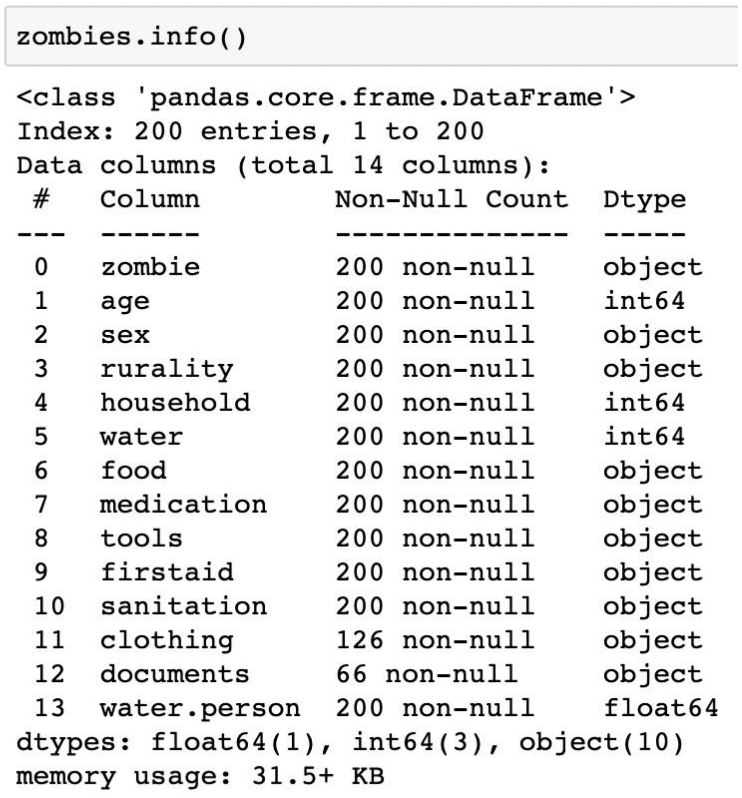
Handling missing values

Let’s find the shape of our dataset first. To find the shape of our data, the .shape method is used. To find the data type, .info() function is used.



```
zombies.shape
```

```
(200, 14)
```



```
zombies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 200 entries, 1 to 200
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   zombie        200 non-null    object
 1   age           200 non-null    int64
 2   sex           200 non-null    object
 3   rurality      200 non-null    object
 4   household     200 non-null    int64
 5   water         200 non-null    int64
 6   food          200 non-null    object
 7   medication    200 non-null    object
 8   tools         200 non-null    object
 9   firstaid      200 non-null    object
 10  sanitation    200 non-null    object
 11  clothing      126 non-null    object
 12  documents     66 non-null     object
 13  water.person  200 non-null    float64
dtypes: float64(1), int64(3), object(10)
memory usage: 31.5+ KB
```

For checking the null values, .isnull() function is used. To sum those null values we use .sum() function. From the below image we found that there are 200 null values present in our dataset in broad_impact. So we can drop the column.

```
#Check for null values
np.sum(zombies.isnull())
```

```
zombie            0
age               0
sex               0
rurality          0
household         0
water             0
food              0
medication        0
tools             0
firstaid          0
sanitation        0
clothing         74
documents       134
water.person      0
dtype: int64
```

We can check for data types to find out categorical data

```
#Check datatype of all features
dataTypeSeries = zombies.dtypes
print('Data type of each column of timesData Dataframe :')
print(dataTypeSeries)
```

```
Data type of each column of timesData Dataframe :
zombie           object
age               int64
sex              object
rurality         object
household         int64
water             int64
food             object
medication       object
tools            object
firstaid         object
sanitation       object
clothing         object
documents        object
water.person    float64
dtype: object
```

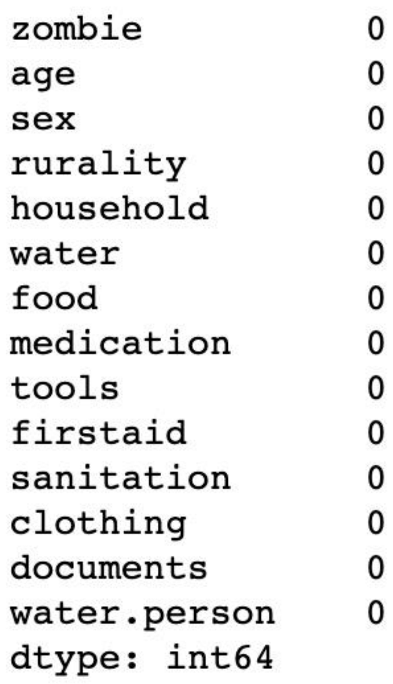
```
#Convert Columns with null values into data type category
for col in ['clothing', 'documents']:
    zombies[col] = zombies[col].astype('category')
```

```python
#Check if conversion successfull
dataTypeSeries = zombies.dtypes
print('Data type of each column of timesData Dataframe :')
print(dataTypeSeries)
```

```
Data type of each column of timesData Dataframe :
zombie             object
age                 int64
sex                object
rurality           object
household           int64
water               int64
food               object
medication         object
tools              object
firstaid           object
sanitation         object
clothing         category
documents        category
water.person      float64
dtype: object
```

```python
# Add new level and recode NA to "No clothing"
zombies["clothing"] = pd.Categorical(zombies["clothing"], categories=pd.unique(zombies["clothing"].cat.categories.appen
zombies['clothing'].fillna('No clothing', inplace=True)

# Add new level and recode NA to "No documents"
zombies["documents"] = pd.Categorical(zombies["documents"], categories=pd.unique(zombies["documents"].cat.categories.ap
zombies['documents'].fillna('No documents', inplace=True)

# Check recoding
np.sum(zombies.isnull())
```

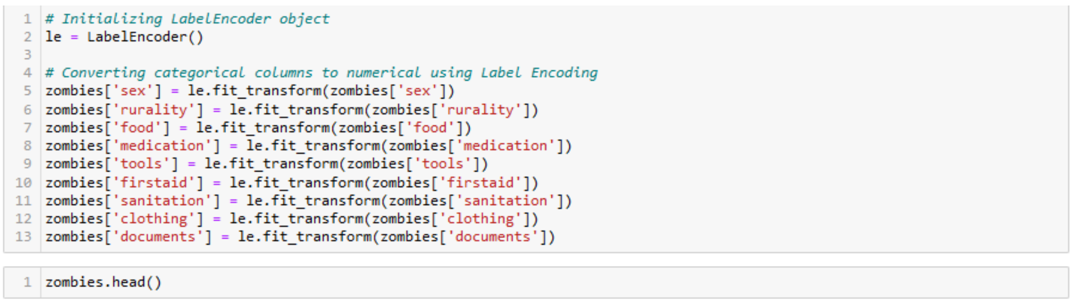
```
zombie          0
age             0
sex             0
rurality        0
household       0
water           0
food            0
medication      0
tools           0
firstaid        0
sanitation      0
clothing        0
documents       0
water.person    0
dtype: int64
```



```
zombie          0
age             0
sex             0
rurality        0
household       0
water           0
food            0
medication      0
tools           0
firstaid        0
sanitation      0
clothing        0
documents       0
water.person    0
dtype: int64
```

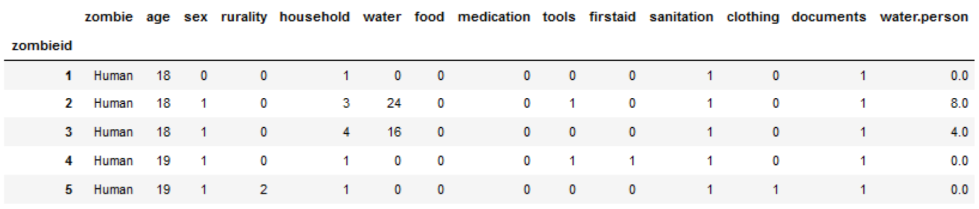
Handling Categorical Values

We will deal with categorical values after splitting the dataset into dependent and Independent variables



```python
# Initializing LabelEncoder object
le = LabelEncoder()

# Converting categorical columns to numerical using Label Encoding
zombies['sex'] = le.fit_transform(zombies['sex'])
zombies['rurality'] = le.fit_transform(zombies['rurality'])
zombies['food'] = le.fit_transform(zombies['food'])
zombies['medication'] = le.fit_transform(zombies['medication'])
zombies['tools'] = le.fit_transform(zombies['tools'])
zombies['firstaid'] = le.fit_transform(zombies['firstaid'])
zombies['sanitation'] = le.fit_transform(zombies['sanitation'])
zombies['clothing'] = le.fit_transform(zombies['clothing'])
zombies['documents'] = le.fit_transform(zombies['documents'])
```

```python
zombies.head()
```



	zombie	age	sex	rurality	household	water	food	medication	tools	firstaid	sanitation	clothing	documents	water.person
zombieid														
1	Human	18	0	0	1	0	0	0	0	0	1	0	1	0.0
2	Human	18	1	0	3	24	0	0	1	0	1	0	1	8.0
3	Human	18	1	0	4	16	0	0	0	0	1	0	1	4.0
4	Human	19	1	0	1	0	0	0	1	1	1	0	1	0.0
5	Human	19	1	2	1	0	0	0	0	0	1	1	1	0.0

Exploratory Data Analysis

Descriptive Statistical

Descriptive analysis is to study the basic features of data with the statistical process. Here pandas has a worthy function called describe. With this describe function we can understand the unique, top and frequent values of categorical features. And we can find mean, std, min, max and percentile values of continuous features.

```
# Descriptive Analytics
print(zombies.describe())
```

```
         age  household  water
count 200.00     200.00 200.00
mean   44.41       2.68   8.75
std    17.37       1.26  12.07
min    18.00       1.00   0.00
25%    29.00       2.00   0.00
50%    42.00       2.50   8.00
75%    58.00       4.00   8.00
max    85.00       6.00  40.00
```

Compare Humans and Zombies - Categorical features

It is important to compare humans and zombies to identify differences in supplies. We need to identify the characteristics and supplies that differ between humans and zombies? Do zombies live in urban areas? Or are they more common in rural areas? Is water critical to staying human? Is food critical to staying human?

```
#Compare categorical Features
zombies_factors = zombies.select_dtypes(include='object')
# Write a function to get percent zombies
perc_zombies = [pd.crosstab(zombies_factors[x],zombies['zombie'], normalize='index')
                for x in zombies_factors.columns
               ]
# Print the data
for df in perc_zombies:
    print(df,"\n")
```

```
zombie  Human  Zombie
zombie
Human    1.00    0.00
Zombie   0.00    1.00

zombie  Human  Zombie
sex
Female   0.63    0.37
Male     0.58    0.42

zombie    Human  Zombie
rurality
Rural      0.82    0.18
Suburban   0.52    0.48
Urban      0.30    0.70

zombie   Human  Zombie
food
Food      0.83    0.17
No food   0.33    0.67

zombie         Human  Zombie
medication
Medication      0.83    0.17
No medication   0.41    0.59

zombie    Human  Zombie
tools
No tools   0.60    0.40
tools      0.61    0.39

zombie                 Human  Zombie
firstaid
First aid supplies      0.63    0.37
No first aid supplies   0.57    0.43

zombie         Human  Zombie
sanitation
No sanitation   0.47    0.53
Sanitation      0.74    0.26
```

```
zombie         Human  Zombie
sanitation
No sanitation   0.47    0.53
Sanitation      0.74    0.26
```

```
zombie    Human  Zombie
clothing
Clothing   0.59    0.41

zombie     Human  Zombie
documents
Documents   0.67    0.33
```

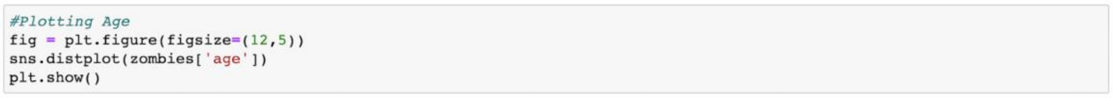
Visual Analysis

Visual analysis is the process of using visual representations, such as charts, plots, and graphs, to explore and understand data. It is a way to quickly identify patterns, trends, and outliers in the data, which can help to gain insights and make informed decisions. We will be using seaborn and plotly packages from python to do so.

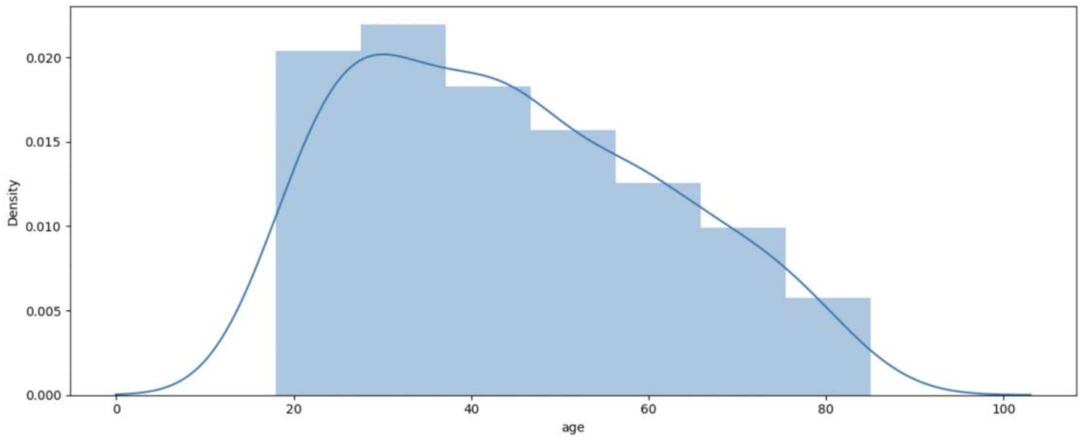
\

Univariate analysis

Seaborn package provides the distplot function. With the help of distplot, we can find the age distribution among the people.



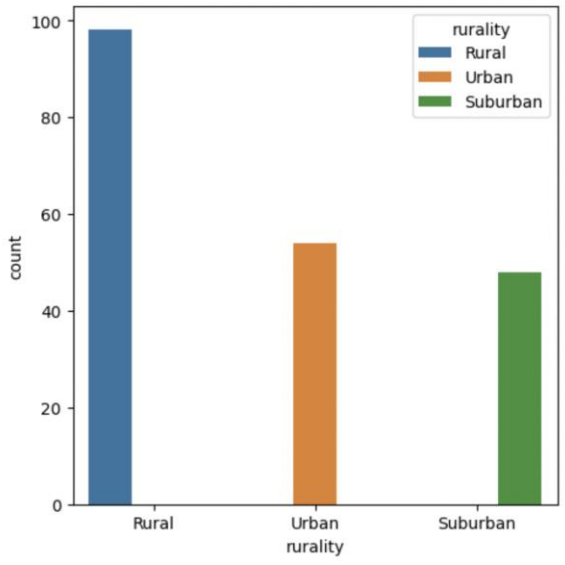
```
#Plotting Age
fig = plt.figure(figsize=(12,5))
sns.distplot(zombies['age'])
plt.show()
```



From the plot we can see that most candidates are between the age 20 to 40.

Plotting categorical data Rurality

```
#Plotting Rurality
fig = plt.figure(figsize=(5,5))
sns.countplot(x=zombies['rurality'], data=zombies['rurality'], hue=zombies['rurality'])
plt.show()
```



From the above graph, we can see that most of the candidates are rural residentials.

Bivariate Analysis

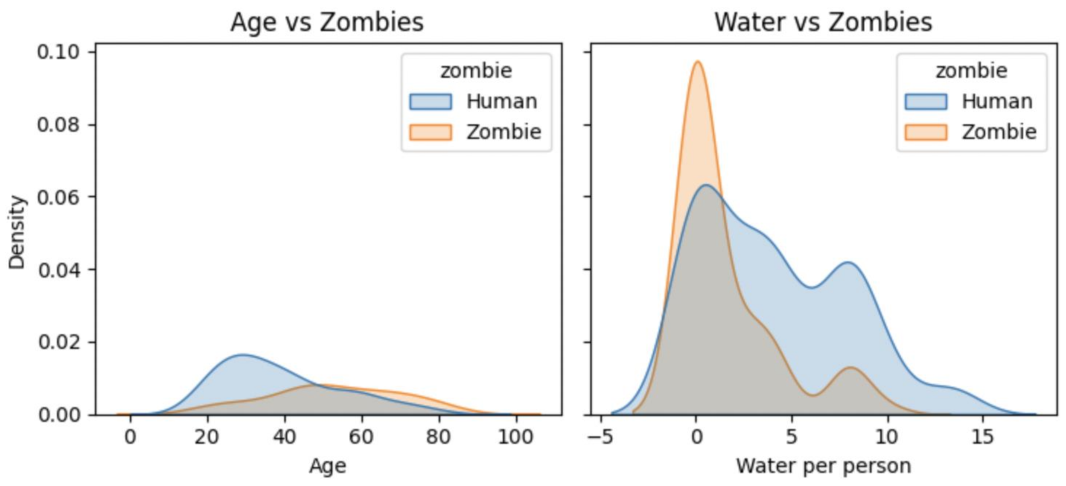
To find the relation between two features we use bivariate analysis. Comparing humans and zombies is important to identify differences in supplies. First we create a new variable water.person that indicates water per person and we examine it.

```
# Create water-per-person
zombies['water.person'] = zombies['water'] / zombies['household']

# Examine the new variable
print(zombies['water.person'].describe())
```

```
count   200.00
mean      3.09
std       3.63
min       0.00
25%       0.00
50%       2.00
75%       5.33
max      13.33
Name: water.person, dtype: float64
```

Then we create and compare age vs zombies and water vs zombies graphs using seaborn kdeplot function.

```
# Create the ageZombies graph
f, (ageZombies, waterPersonZom) = plt.subplots(1, 2, sharey=True)
ageZombies = sns.kdeplot(data = zombies, x = "age", fill = True, hue = "zombie", ax=ageZombies)
ageZombies.set(title="Age vs Zombies", xlabel="Age", ylabel="Density")

# Create the waterPersonZom graph
waterPersonZom = sns.kdeplot(data = zombies, x = "water.person", fill = True, hue = "zombie", ax=waterPersonZom)
waterPersonZom.set(title="Water vs Zombies", xlabel="Water per person", ylabel="Density")

plt.show()
```

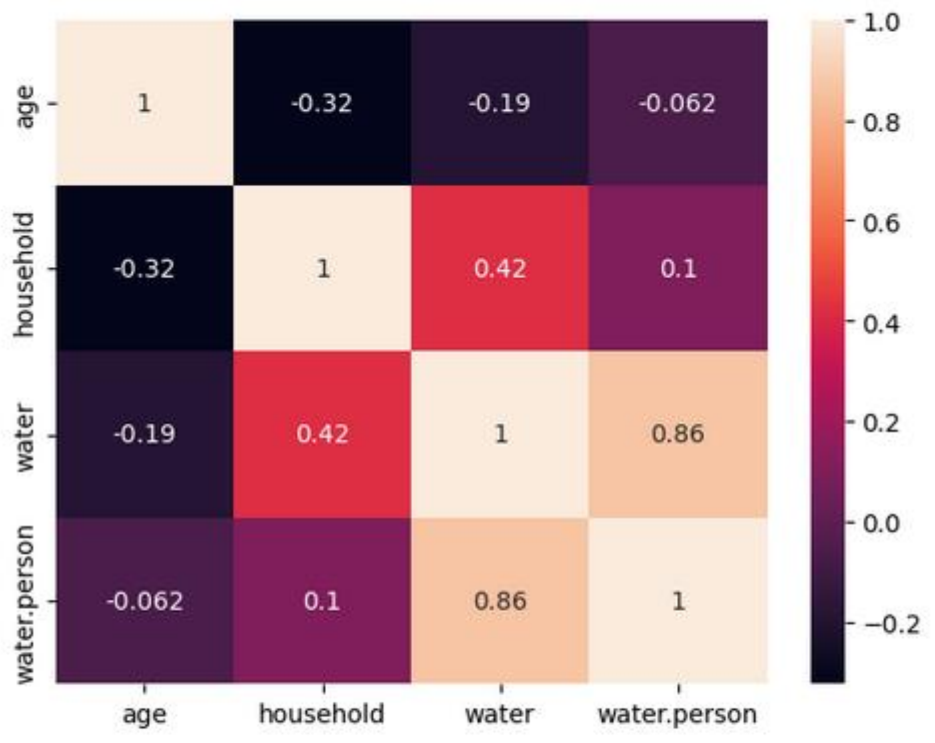


It looks like those who turned into zombies were older and had less available clean water. This suggests that getting water to the remaining humans might help protect them from the zombie hoards! Protecting older citizens is important, so we need to think about the best ways to reach this group.

Multivariate Analysis

```
sns.heatmap(zombies.corr(),annot=True)
```

```
<AxesSubplot:>
```



Splitting data into train and test

Now let’s split the Dataset into train and test sets. First split the dataset into X and y and then split the data set

```
# Splitting data into features and target
X = zombies.drop(['zombie','water.person'], axis=1)
y = zombies['zombie']
```

Here x and y variables are created. On x variable, features are passed. And on y target variable is passed. For splitting training and testing data we are using train_test_split() function from sklearn. As parameters, we are passing x, y, test_size, random_state.

```
#Features
X
```

zombieid	age	sex	rurality	household	water	food	medication	tools	firstaid	sanitation	clothing	documents
1	18	0	0	1	0	0	0	0	0	1	0	1
2	18	1	0	3	24	0	0	1	0	1	0	1
3	18	1	0	4	16	0	0	0	0	1	0	1
4	19	1	0	1	0	0	0	1	1	1	0	1
5	19	1	2	1	0	0	0	0	0	1	1	1
...	...	...	...	...	...	...	...	...	...	...	...	...
196	68	1	1	1	0	0	1	0	1	1	0	0
197	71	1	1	1	8	1	1	1	0	0	0	1
198	76	0	2	1	0	1	1	1	0	1	0	0
199	82	1	2	1	0	1	1	0	1	0	1	1
200	85	1	2	1	0	1	0	0	1	1	0	1

200 rows × 12 columns

```
#Target Variable
y
```

```
zombieid
1        Human
2        Human
3        Human
4        Human
5        Human
         ...
196     Zombie
197     Zombie
198     Zombie
199     Zombie
200     Zombie
Name: zombie, Length: 200, dtype: object
```

```
# Splitting the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Model Building

Training The Model In Multiple Algorithms

Now our data is cleaned and it's time to build the model. We can train our data on different algorithms. For this project we are applying four classification algorithms. The best model is saved based on its performance.

Defining the models

First we define four different machine learning models. All four are classification algorithms. We use their respective classes from sklearn library to define them.
This includes

- LogisticRegression()
- DecisionTreeClassifier()
- RandomForestClassifier() and
- SVC().

We store them in variables lr, dt, rf and svm respectively.

```
# Defining the models
lr = LogisticRegression()
dt = DecisionTreeClassifier()
rf = RandomForestClassifier()
svm = SVC()
```

Training and predicting with each model

A for loop iterates through each of the four models.

Inside the loop, the current model is trained using the fit() method with the training dataset X_train and y_train.

The trained model then makes predictions on the test dataset X_test using the predict() method and the predictions are stored in y_pred.

```
# Training and predicting with each model
for model in [lr, dt, rf, svm]:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)
```

Testing The Model

Here we have tested with the Logistic Regression algorithm. You can test with all algorithms. With the help of predict() function.

```
y_pred = lr.predict([[18,0,0,1,0,0,0,0,0,1,0,1]])
print(y_pred)
```

```
['Human']
```

Performance Testing & Hyperparameter Tuning

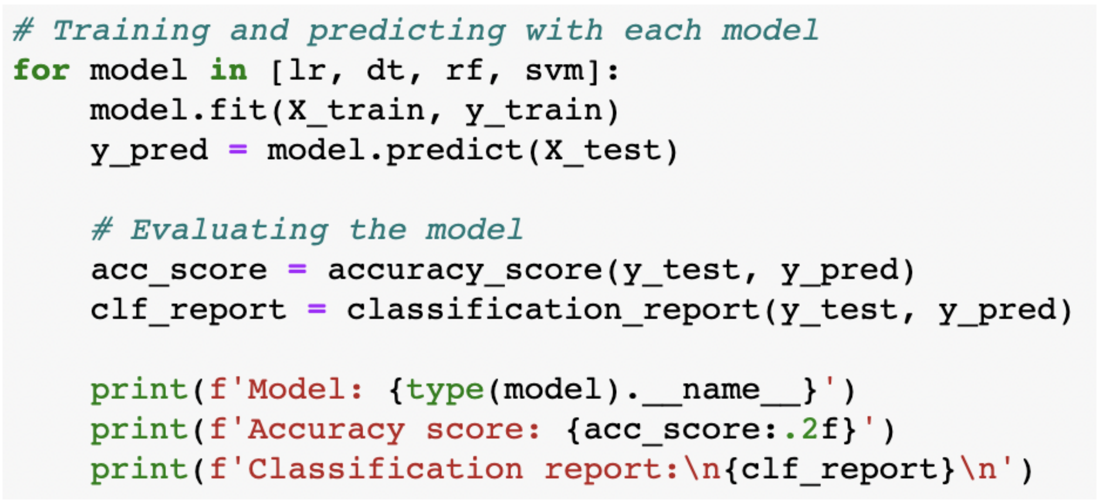
Multiple evaluation metrics means evaluating the model's performance on a test set using different performance measures. This can provide a more comprehensive understanding of the model's strengths and weaknesses. We are using evaluation metrics for classification tasks including accuracy, precision, recall, support and F1-score.

Compare the models

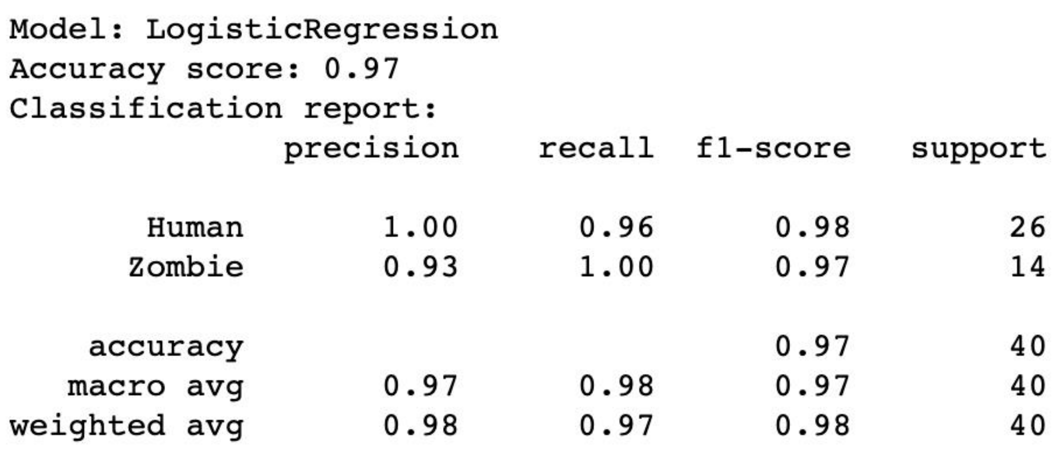
A for loop iterates through each of the four models. Inside the loop, The accuracy of the model is evaluated by comparing the predictions with the actual labels using the accuracy_score() function, and the result is stored in acc_score.

The classification_report() function is used to generate a text report of the main classification metrics (precision, recall, F1-score, and support) for each class, and the result is stored in clf_report.

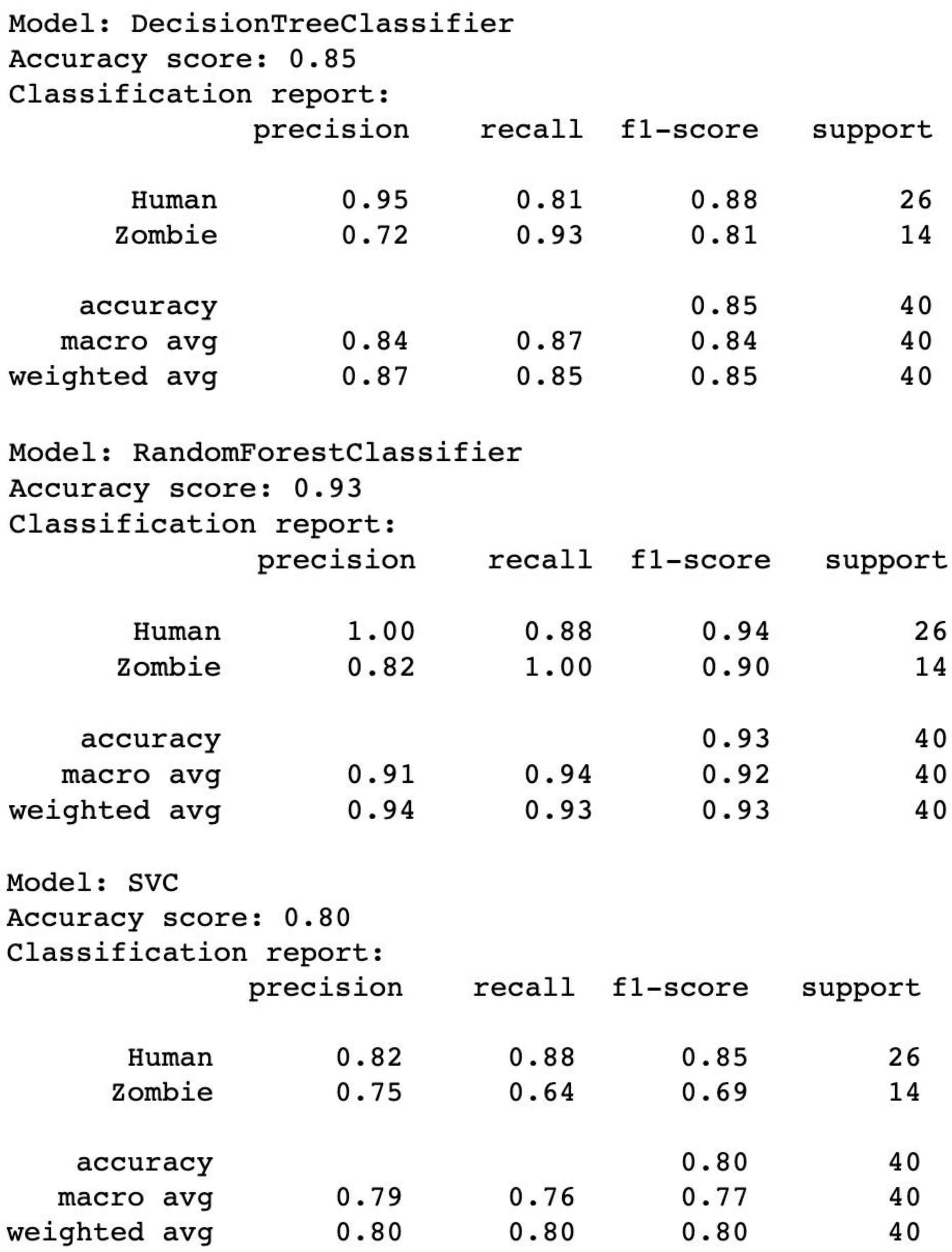
The results are printed to the console using the print() function. The model name, accuracy score, and classification report are displayed for each model, and the loop iterates until all four models have been trained and evaluated.



```python
# Training and predicting with each model
for model in [lr, dt, rf, svm]:
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Evaluating the model
    acc_score = accuracy_score(y_test, y_pred)
    clf_report = classification_report(y_test, y_pred)

    print(f'Model: {type(model).__name__}')
    print(f'Accuracy score: {acc_score:.2f}')
    print(f'Classification report:\n{clf_report}\n')
```



```
Model: LogisticRegression
Accuracy score: 0.97
Classification report:
              precision    recall  f1-score   support

       Human       1.00      0.96      0.98        26
      Zombie       0.93      1.00      0.97        14

    accuracy                           0.97        40
   macro avg       0.97      0.98      0.97        40
weighted avg       0.98      0.97      0.98        40
```



```
Model: DecisionTreeClassifier
Accuracy score: 0.85
Classification report:
              precision    recall  f1-score   support

       Human       0.95      0.81      0.88        26
      Zombie       0.72      0.93      0.81        14

    accuracy                           0.85        40
   macro avg       0.84      0.87      0.84        40
weighted avg       0.87      0.85      0.85        40

Model: RandomForestClassifier
Accuracy score: 0.93
Classification report:
              precision    recall  f1-score   support

       Human       1.00      0.88      0.94        26
      Zombie       0.82      1.00      0.90        14

    accuracy                           0.93        40
   macro avg       0.91      0.94      0.92        40
weighted avg       0.94      0.93      0.93        40

Model: SVC
Accuracy score: 0.80
Classification report:
              precision    recall  f1-score   support

       Human       0.82      0.88      0.85        26
      Zombie       0.75      0.64      0.69        14

    accuracy                           0.80        40
   macro avg       0.79      0.76      0.77        40
weighted avg       0.80      0.80      0.80        40
```

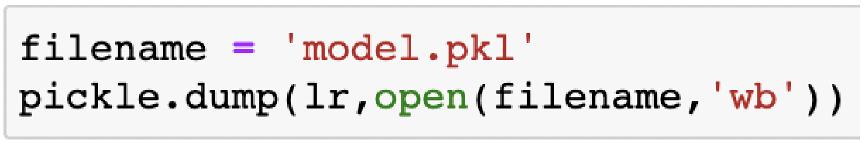
After calling the function, the results of models are displayed as output. From the three models logistic regression is performing well.

Testing Model With Multiple Evaluation Metrics

Model Deployment

Save The Best Model

Saving the best model after comparing its performance using different evaluation metrics means selecting the model with the highest performance and saving its weights and configuration. This can be useful in avoiding the need to retrain the model every time it is needed and also to be able to use it in the future.



```
filename = 'model.pkl'
pickle.dump(lr,open(filename,'wb'))
```

Integrate With Web Framework

In this section, we will be building a web application that is integrated to the model we built. A UI is provided for the uses where he has to enter the values for predictions. The enter values are given to the saved model and prediction is showcased on the UI.

This section has the following tasks

- Building HTML Pages
- Building server-side script
- Run the web application

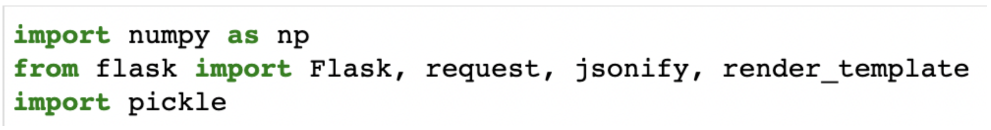
Building Html Pages:

- index.html

For this project create 1 HTML file namely and save them in the templates folder.

Build Python code:

Import the libraries



```
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle
```

Load the saved model. Importing the flask module in the project is mandatory. An object of Flask class is our WSGI application. Flask constructor takes the name of the current module (name) as argument.

```python
# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))
```

Render HTML Page:

```python
@flask_app.route("/")
def Home():
    return render_template("index.html")
```

Here we will be using a declared constructor to route to the HTML page which we have created earlier.

In the above example, ‘/’ URL is bound with the home.html function. Hence, when the home page of the web server is opened in the browser, the html page will be rendered. Whenever you enter the values from the html page the values can be retrieved using POST Method.

Retrieves the value from UI:

```python
@flask_app.route("/predict", methods = ["POST"])
def predict():
    int_features = [int(x) for x in request.form.values()]
    features = [np.array(int_features)]
    prediction = model.predict(features)
    output = prediction[0]
    return render_template("index.html", prediction_text = "{} detected".format(output))
```

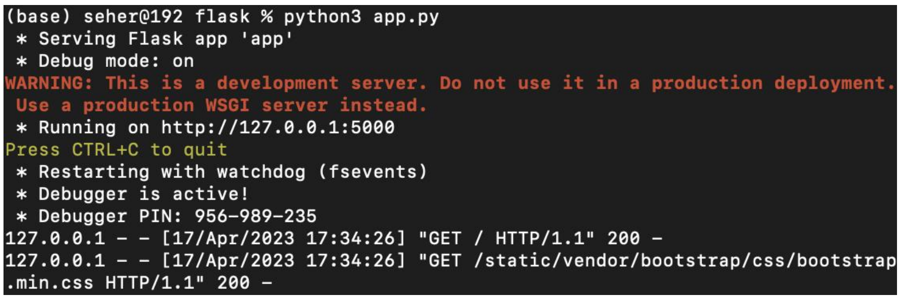
Here we are routing our app to predict() function. This function retrieves all the values from the HTML page using Post request. That is stored in an array. This array is passed to the model.predict() function. This function returns the prediction. And this prediction value will be rendered to the text that we have mentioned in the submit.html page earlier.

Main Function:

```python
if __name__ == "__main__":
    flask_app.run(debug=True)
```

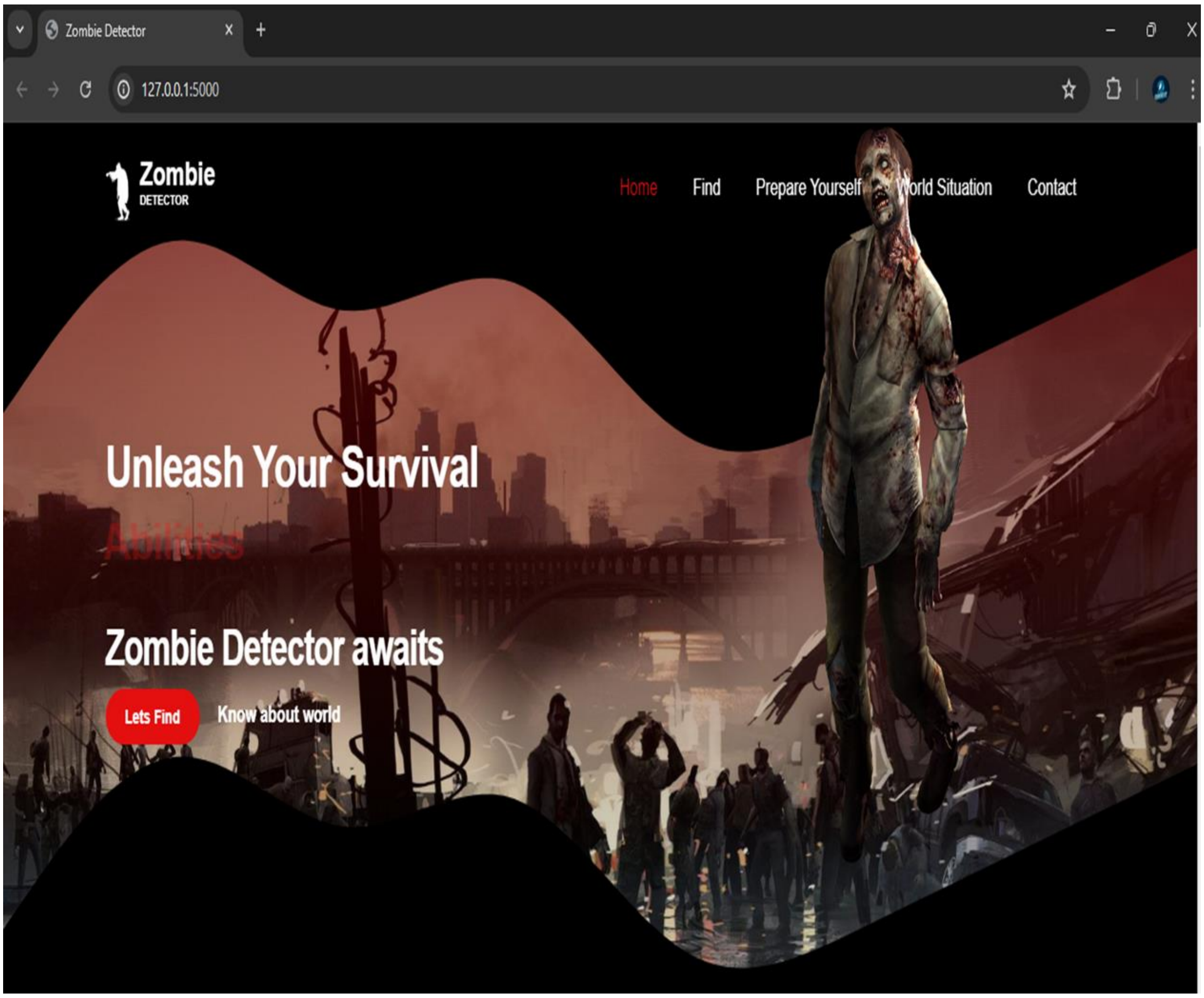
Run the Web Application:

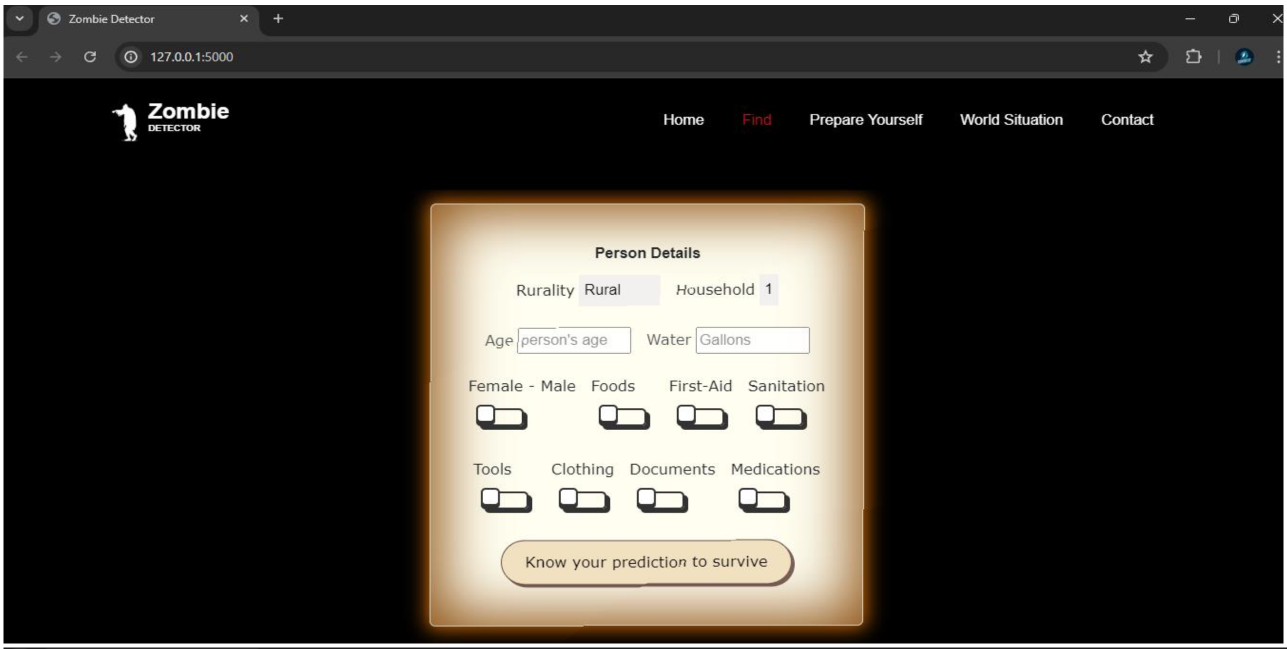
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see the result/prediction on the web.

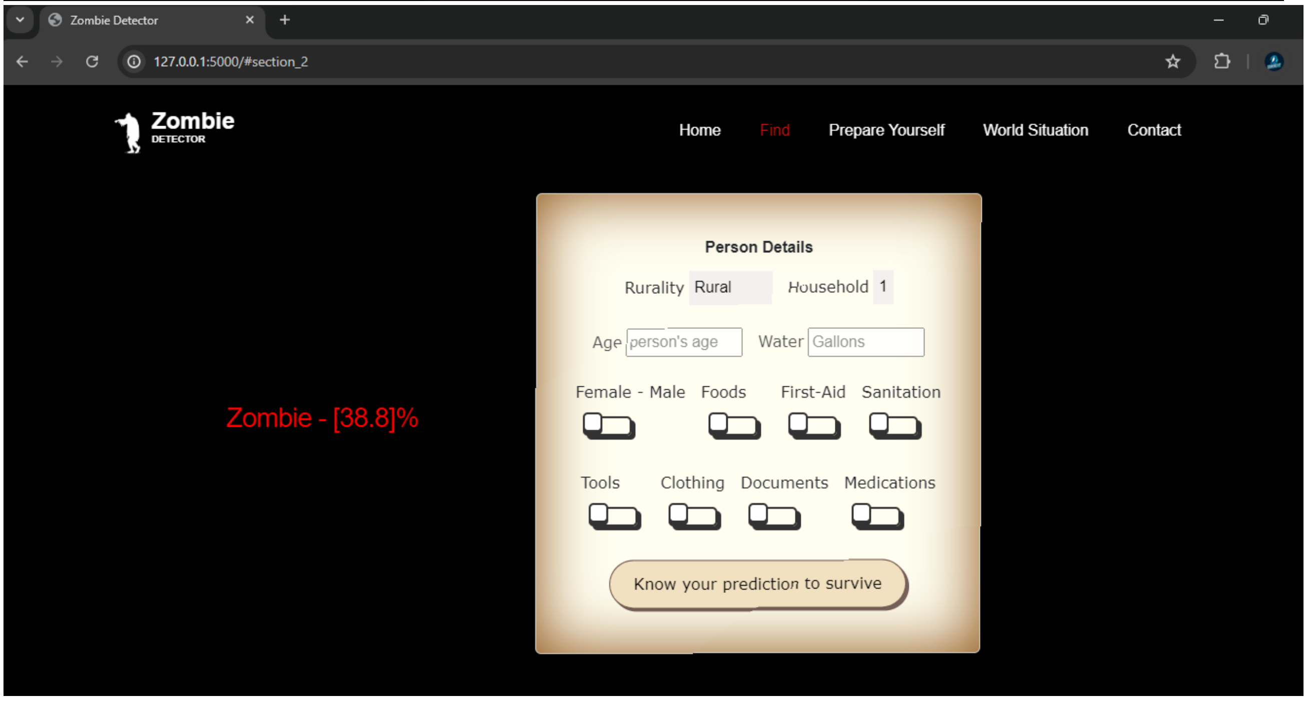


```
(base) seher@192 flask % python3 app.py
 * Serving Flask app 'app'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment.
 Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with watchdog (fsevents)
 * Debugger is active!
 * Debugger PIN: 956-989-235
127.0.0.1 - - [17/Apr/2023 17:34:26] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [17/Apr/2023 17:34:26] "GET /static/vendor/bootstrap/css/bootstrap
.min.css HTTP/1.1" 200 -
```

Now,Go the web browser and write the localhost url (http://127.0.0.1:5000) to get the below result




Zoombie Detector
127.0.0.1:5000
Zombie DETECTOR
Home Find Prepare Yourself World Situation Contact
Person Details
Rurality: Rural
Household: 1
Age: person's age
Water: Gallons
Female - Male
Foods
First-Aid
Sanitation
Tools
Clothing
Documents
Medications
Know your prediction to survive


Zombie Detector
127.0.0.1:5000/#section_2
Zombie DETECTOR
Home Find Prepare Yourself World Situation Contact
Zombie - [38.8]%
Person Details
Rurality: Rural
Household: 1
Age: person's age
Water: Gallons
Female - Male
Foods
First-Aid
Sanitation
Tools
Clothing
Documents
Medications
Know your prediction to survive

Project Demonstration & Documentation

Below mentioned deliverables to be submitted along with other deliverables

Activity 1:- Record explanation Video for project end to end solution

Activity 2:- Project Documentation-Step by step project development procedure

Create document as per the template provided