ZOMBIE DETECTOR:

Using ML to save lives during a Zombie Apocalypse

Project Hand-out – Seher Khan

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.

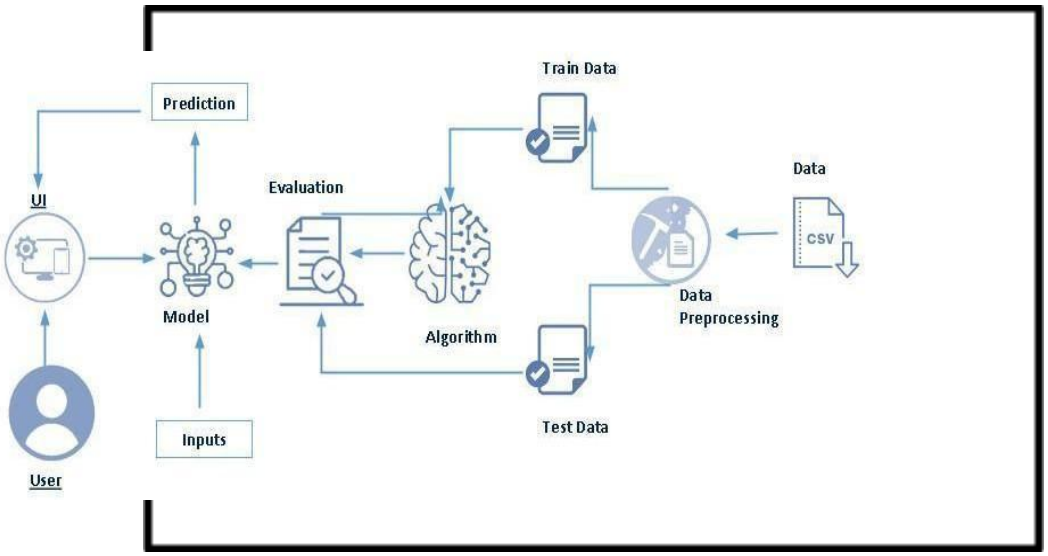
This project can be used for:

Emergency Preparedness: The project can help raise awareness about the importance of emergency preparedness, especially during disasters or crises that can be as unpredictable and catastrophic as a zombie outbreak. The model developed can be used to identify areas that are at high risk of infection, and the Flask app can help people understand their own risk levels and the supplies they need to stay safe.

Disaster Response: The model can be integrated into emergency response systems to make informed decisions about resource allocation and logistics. The predictions generated by the model can help emergency responders identify areas that need urgent assistance and distribute supplies accordingly.

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:



Project Flow:

- User interacts with the UI to enter the input.
- Entered input is analysed by the model which is integrated.
- Once model analyses the input the prediction is showcased on the

UI To accomplish this, we have to complete all the activities listed below,

- Define Problem / Problem Understanding
  - Specify the business problem
  - Business requirements
  - Literature Survey
  - Social or Business Impact.
- Data Collection & Preparation
  - Collect the dataset
  - Data Preparation
- Exploratory Data Analysis
  - Descriptive statistical
  - Visual Analysis
- Model Building
  - Training the model in multiple algorithms
  - Testing the model
- Performance Testing & Hyperparameter Tuning
  - Testing model with multiple evaluation metrics
  - Comparing model accuracy before & after applying hyperparameter tuning
- Model Deployment
  - Save the best model
  - Integrate with Web Framework
- Project Demonstration & Documentation
  - Record explanation Video for project end to end solution
  - Project Documentation-Step by step project development procedure

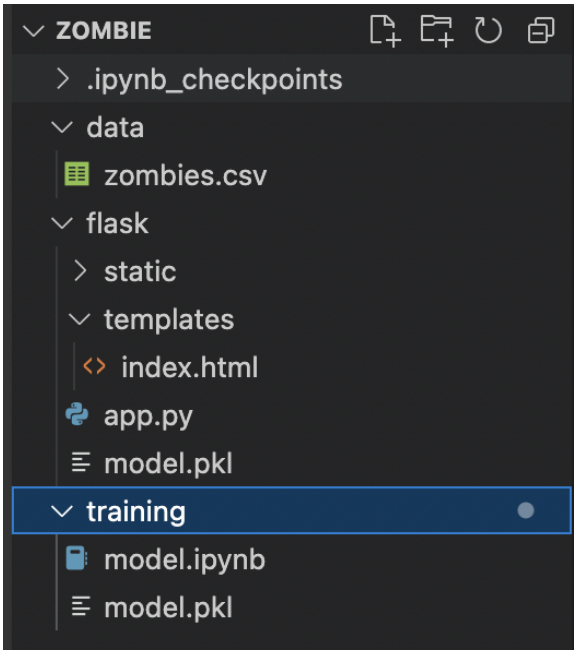
Prior Knowledge:

You must have prior knowledge of the following topics to complete this project.

- ML Concepts
    - Supervised learning: https://www.javatpoint.com/supervised-machine-learning
    - Unsupervised learning: https://www.javatpoint.com/unsupervised-machine-learning
    - Logistic Regression: https://www.geeksforgeeks.org/understanding-logistic-regression/
    - Support Vector Machine: https://www.javatpoint.com/machine-learning-support-vector-machine-algorithm
    - Decision tree: https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm
    - Random forest: https://www.javatpoint.com/machine-learning-random-forest-algorithm
    - Evaluation metrics: https://www.analyticsvidhya.com/blog/2019/08/11-important-model-evaluation-error-metrics/
- Flask Basics : https://www.youtube.com/watch?v=lj4I_CvBnt0

Project Structure:

Create the Project folder which contains files as shown below



- We are building a Flask application that needs HTML pages stored in the templates folder and a Python script app.py for scripting.
- model.pkl is our saved model. Further, we will use this model for flask integration.
- The training folder contains a model training file.

Milestone 1: Define Problem / Problem Understanding

Activity 1: Specify the business problem

Refer Project Description

Activity 2: Business requirements

A zombie detector project can have a variety of business requirements, depending on the specific goals and objectives of the project. Some potential requirements may include:

- Accuracy: The model must have a high degree of accuracy in predicting whether a person is a zombie or not. This is the most important requirement, as the accuracy of the model directly impacts its usefulness in emergency response systems and disaster recovery.
- Usability: The Flask app must be user-friendly and easy to navigate. This requirement is important to ensure that people can quickly and easily input their data and get a prediction.
- Security: The app must have robust security measures in place to protect user data and prevent unauthorized access. This requirement is important to protect user privacy and prevent malicious attacks.
- Scalability: The model must be scalable to handle large volumes of data and provide accurate predictions in real time. This requirement is important to ensure that the model can handle the demands of emergency response systems and businesses that deal with disaster recovery.
- Integration: The model must be easily integrable into existing systems and workflows. This requirement is important to ensure that the model can be used by businesses that already have established emergency management and disaster recovery systems in place.

Activity 3: Literature Survey (Student Will Write)

A literature survey for a zombie detection project would involve researching and reviewing existing studies, articles, and other publications on the topic of zombie apocalypse and survival. The survey would aim to gather information on current disaster management systems, their strengths and weaknesses, and any gaps in knowledge that the project could address. The literature survey would also look at the methods and techniques used in previous projects, and any relevant data or findings that could inform the design and implementation of the current project.

Activity 4: Social or Business Impact.

Social Impact:- The project has significant social impact potential as it focuses on identifying supplies that are associated with safety during a zombie outbreak, which can be extrapolated to other emergencies as well. This model can be used by disaster response organizations to make informed decisions on which supplies are needed in specific areas during a crisis and to allocate resources accordingly. The model can also be used by public health agencies to track disease outbreaks, especially in rural areas, and to make better decisions on disease control measures.

Business Model/Impact:- The prediction model developed in this project can be used by various businesses that deal with emergency management or disaster recovery. These businesses can integrate this model into their systems to make better decisions related to supply chain management, logistics, and resource allocation during a crisis such as a natural disaster, pandemic, or zombie outbreak. The model can also be used by insurance companies to identify high-risk areas and provide coverage accordingly.

Milestone 2: 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.

Activity 1: 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.

Activity 1.1: 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')
```

Activity 1.2: 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

Activity 2: 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
- Handling Outliers

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.

Activity 2.1: 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
```

Activity 2.2: Handling Categorical Values

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')
```

```
#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
```

```
# Add new level and recode NA to "No clothing"
zombies["clothing"] = pd.Categorical(zombies["clothing"], categories=pd.unique(zombies["clothing"].cat.categories.apper
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
```

Milestone 3: Exploratory Data Analysis

Activity 1.1: Descriptive statistical analysis - Numerical Features

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
```

Activity 1.2: 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

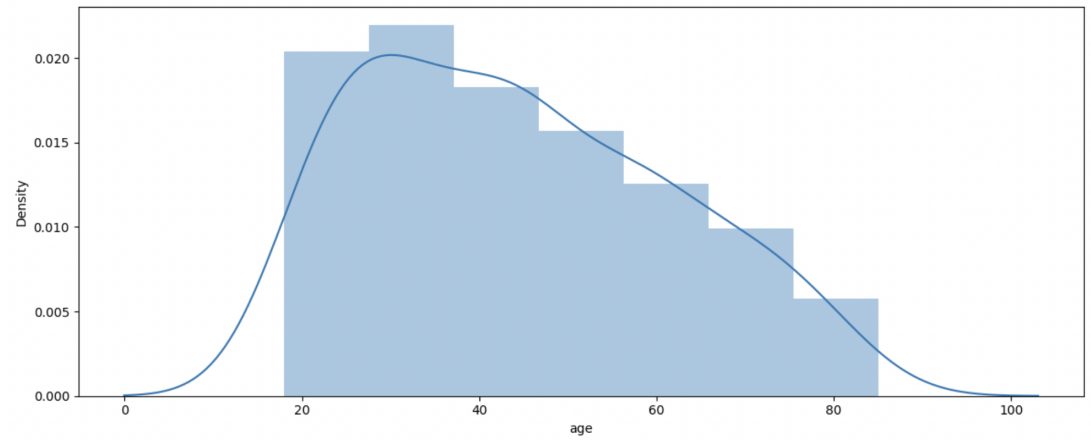
Activity 2: Visual analysis

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.

Activity 2.1: Univariate analysis

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

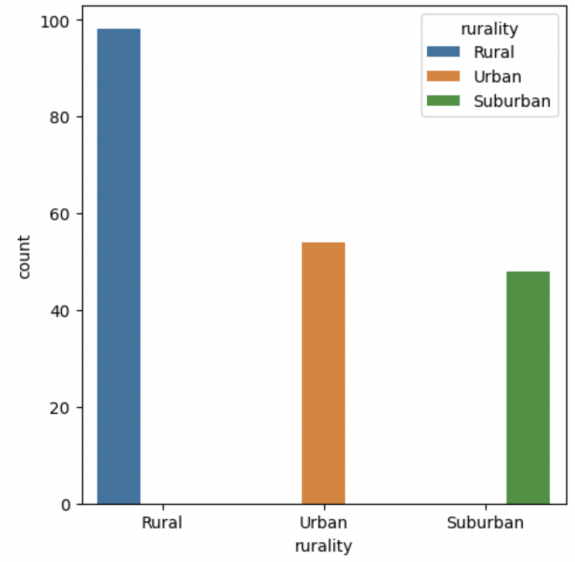
```
#Plotting Age
fig = plt.figure(figsize=(12,5))
sns.distplot(zombies['age'])
plt.show()
```



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.

Activity 2.2: Bivariate analysis: Compare humans and zombies - Part 2

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())
```

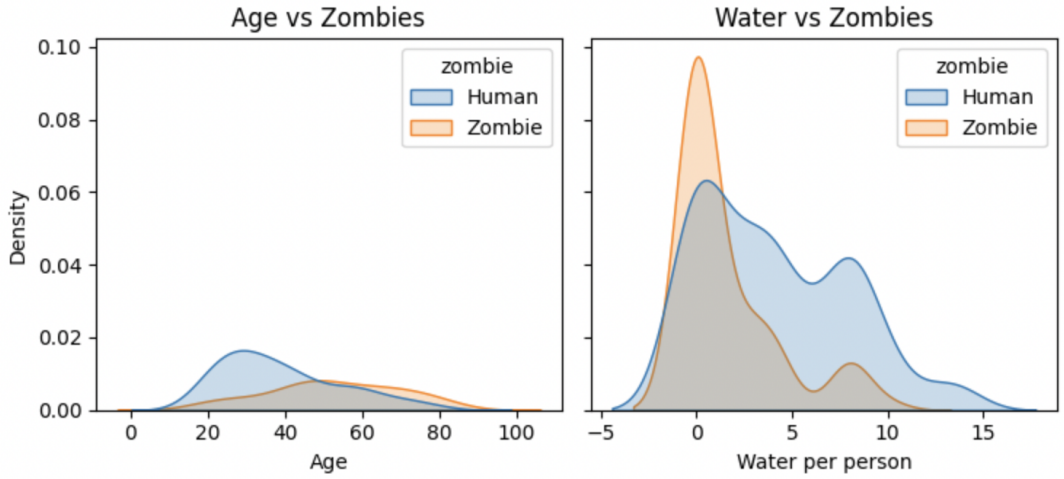
```
count   200.00
mean      3.09
std       3.63
min       0.00
25%       0.00
50%       2.00
75%       5.33
max      13.33
Name: water.person, dtype: float64
```

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.

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']
```

Label encoding categorical features.
Now We use the LabelEncoder function from the sklearn library to encode transform our categorical variables.

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

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

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)
```

Milestone 4: Model Building

Activity 1: 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.

Activity 1.1: 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

1. LogisticRegression()
2. DecisionTreeClassifier()
3. RandomForestClassifier() and
4. SVC().

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

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

Activity 1.2: 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)
```

Activity 2: 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']
```

Milestone 5: Performance Testing

Activity 1: Testing model with multiple evaluation metrics

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.

Activity 1.1: Evaluate and 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.

```
# 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

Milestone 6: Model Deployment

Activity 1: 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'))
```

Activity 2: 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

Activity 2.1: Building Html Pages:

For this project create 1 HTML file namely

- index.html

and save them in the templates folder. Refer this link for templates.

Activity 2.2: 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.

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

Render HTML page:

```
@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:

```
@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:

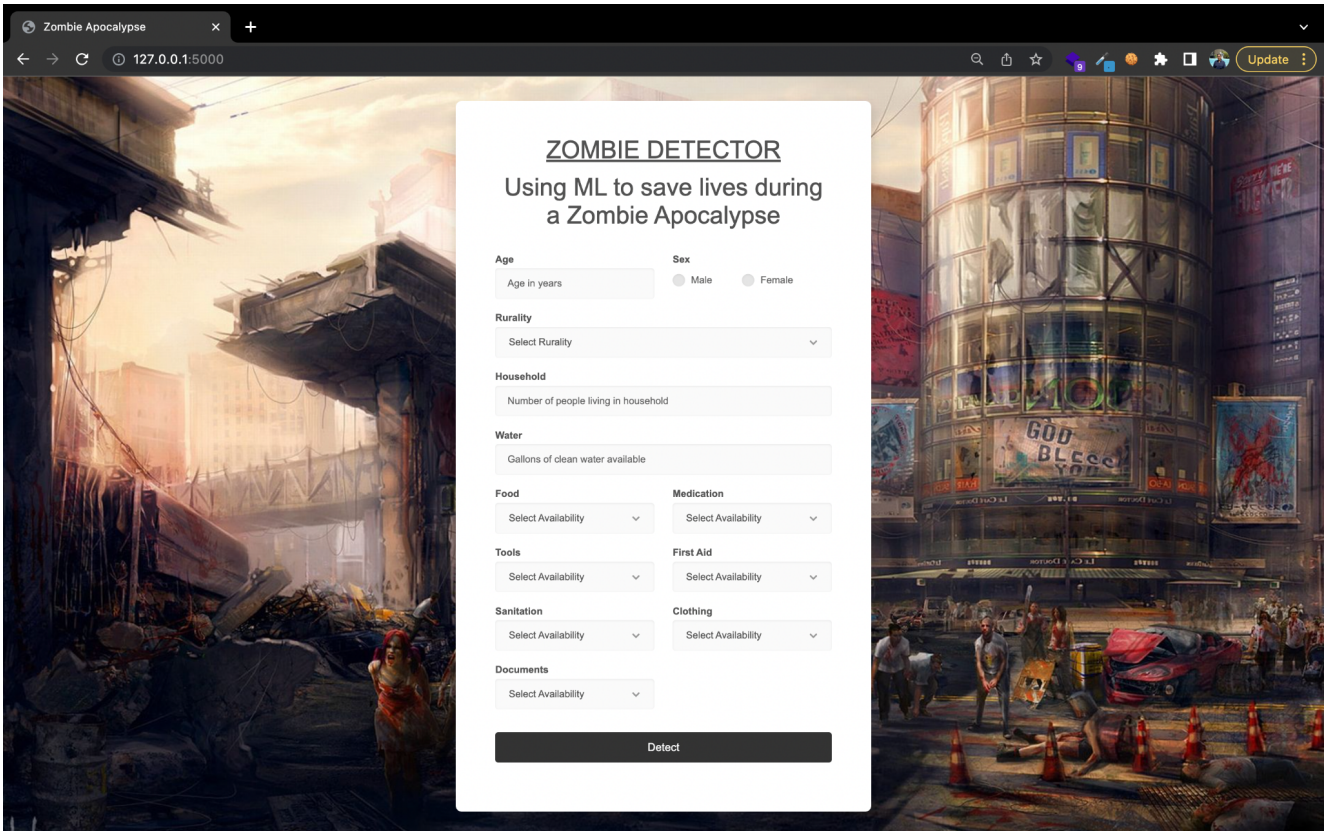
```
if __name__ == "__main__":
    flask_app.run(debug=True)
```

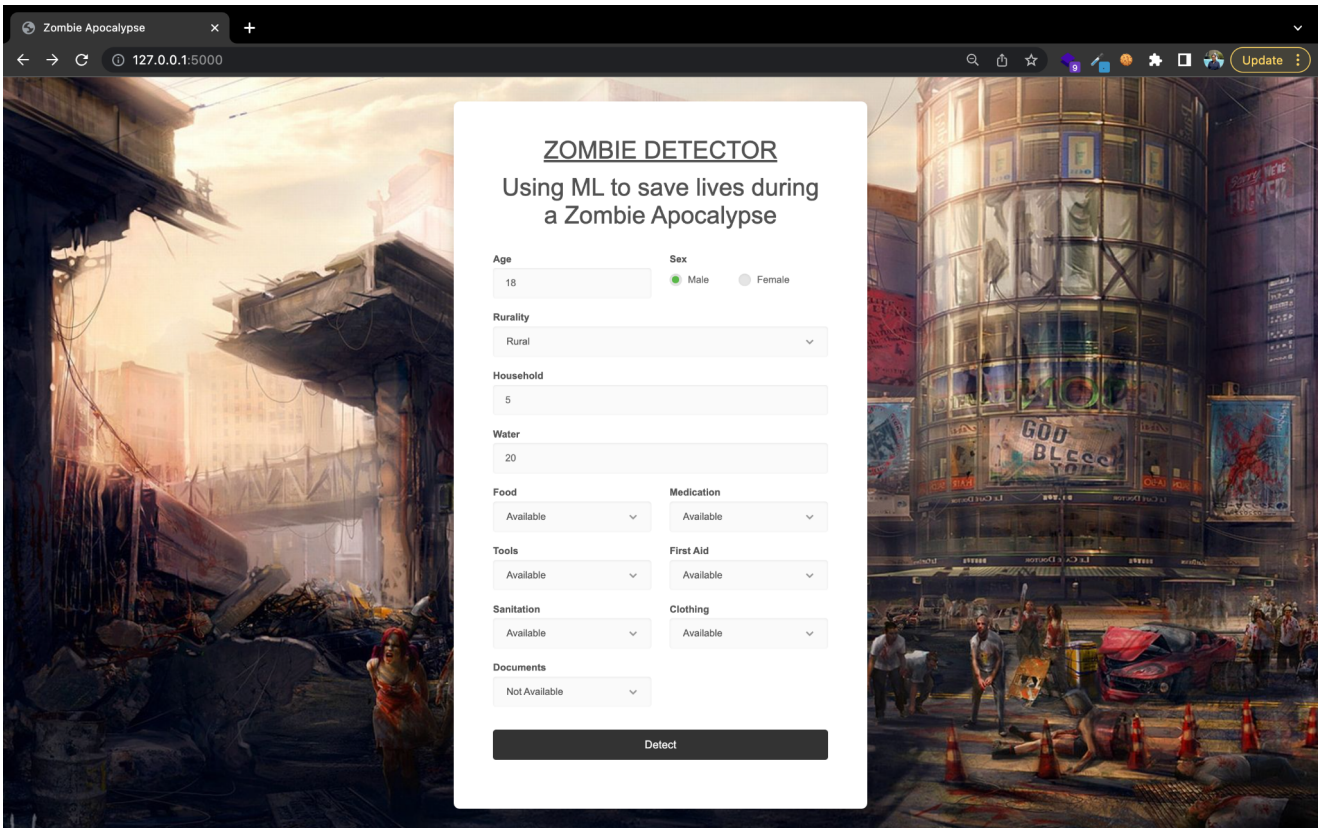
Activity 2.3: Run the web application

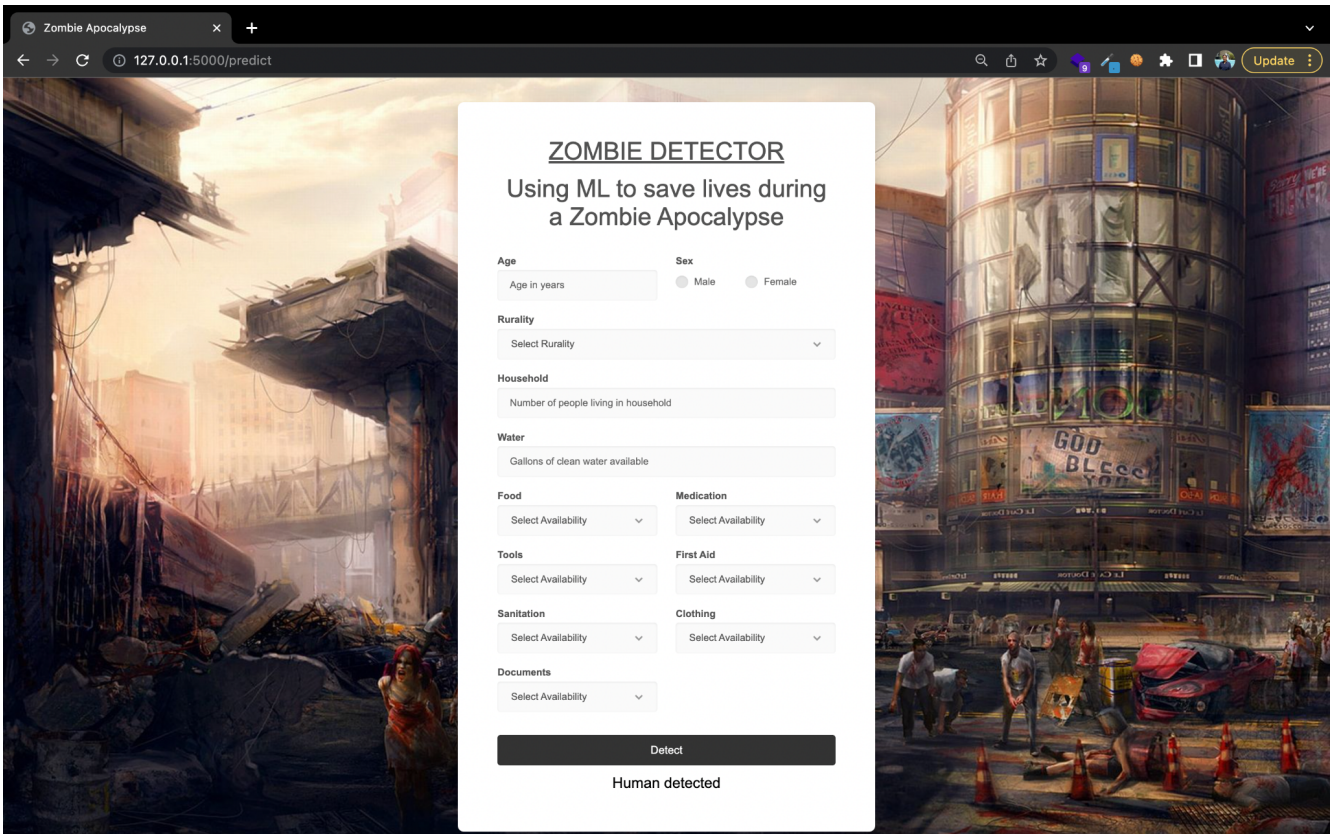
- Open anaconda prompt from the start menu
- Navigate to the folder where your python script is.
- Now type “python app.py” command
- Navigate to the localhost where you can view your web page.
- Click on the predict button from the top left corner, enter the inputs, click on the submit button, and see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


Zombie Apocalypse
127.0.0.1:5000
Update
ZOMBIE DETECTOR
Using ML to save lives during a Zombie Apocalypse
Age
Age in years
Sex
Male
Female
Rurality
Select Rurality
Household
Number of people living in household
Water
Gallons of clean water available
Food
Select Availability
Medication
Select Availability
Tools
Select Availability
First Aid
Select Availability
Sanitation
Select Availability
Clothing
Select Availability
Documents
Select Availability
Detect


Zombie Apocalypse
127.0.0.1:5000
Update
ZOMBIE DETECTOR
Using ML to save lives during a Zombie Apocalypse
Age
18
Sex
Male
Female
Rurality
Rural
Household
5
Water
20
Food
Available
Medication
Available
Tools
Available
First Aid
Available
Sanitation
Available
Clothing
Available
Documents
Not Available
Detect


Zombie Apocalypse
127.0.0.1:5000/predict
Update
ZOMBIE DETECTOR
Using ML to save lives during a Zombie Apocalypse
Age
Age in years
Sex
Male
Female
Rurality
Select Rurality
Household
Number of people living in household
Water
Gallons of clean water available
Food
Select Availability
Medication
Select Availability
Tools
Select Availability
First Aid
Select Availability
Sanitation
Select Availability
Clothing
Select Availability
Documents
Select Availability
Detect
Human detected

Milestone 7: 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