**System Requirements Specification Index**

**For Machine learning Algorithm No 6**

1.0

**Machine Learning usecase: Decision Trees and Random Forests**

**Heart Disease and Chicken Disease Prediction**

**Overview**

**This assessment focuses on implementing two machine learning models for disease prediction:**

1. Heart Disease Prediction using a Random Forest Classifier

2. Chicken Disease Prediction using a Decision Tree Classifier

You will implement code that provides functionality for data loading, preprocessing, model training, and prediction. The goal is to understand the core concepts of data preprocessing, model training, and evaluation for classification tasks.

**Datasets information**

1. Heart Disease Dataset (`heart.csv`)

This dataset contains medical information for 303 patients, with a target variable indicating the presence of heart disease.

Features:

- `age`: Age in years

- `sex`: Sex (1 = male, 0 = female)

- `cp`: Chest pain type (0-3)

- `trestbps`: Resting blood pressure (in mm Hg)

- `chol`: Serum cholesterol in mg/dl

- `fbs`: Fasting blood sugar > 120 mg/dl (1 = true, 0 = false)

- `restecg`: Resting electrocardiographic results (0-2)

- `thalach`: Maximum heart rate achieved

- `exang`: Exercise induced angina (1 = yes, 0 = no)

- `oldpeak`: ST depression induced by exercise relative to rest

- `slope`: Slope of the peak exercise ST segment

- `ca`: Number of major vessels colored by fluoroscopy (0-3)

- `thal`: Thalassemia (3 = normal, 6 = fixed defect, 7 = reversible defect)

Target:

- `target`: Presence of heart disease (1 = yes, 0 = no)

2. Chicken Disease Dataset (`chicken\_disease\_data.csv`)

This dataset contains information about 1000 chickens, with features related to their health and a target variable indicating disease status.

Features:

- `Age`: Age of the chicken in years

- `Weight`: Weight of the chicken in kg

- `Height`: Height of the chicken in cm

- `Egg Production`: Number of eggs produced per week

- `Feed Consumption`: Amount of feed consumed in kg per week

- `Temperature`: Body temperature in Celsius

- `Disease Predicted`: Disease status (Healthy or Diseased)

**Tasks**

**Task 1: Heart Disease Prediction**

You will implement the following functions in `heart.py`:

1. `load\_heart\_disease\_data()`: Load the heart disease dataset

* Load the CSV file "heart.csv"
* Print confirmation message with the number of records loaded
* Return the loaded DataFrame

2. `preprocess\_heart\_data(df)`: Preprocess the heart disease data

* Separate features (X) and target (y)
* X should contain all columns except "target"
* y should contain only the "target" column
* Print confirmation message
* Return X and y

3. `split\_heart\_data(X, y, test\_size=0.2)`: Split the data into training and testing sets

* Use train\_test\_split with test\_size=0.2 and random\_state=42
* Print confirmation message with the size of training and testing sets
* Return X\_train, X\_test, y\_train, y\_test

4. `create\_model(n\_estimators=100, max\_depth=None)`: Create a Random Forest model

* Create a RandomForestClassifier with specified parameters and random\_state=42
* Print confirmation message
* Return the model

5. `train\_model(model, X\_train, y\_train)`: Train the model

* Fit the model to the training data
* Print confirmation message
* Return the trained model

6. `save\_model(model, filename="random\_forest\_heart\_model.pkl")`: Save the model

* Save the model to the specified filename using joblib.dump
* Print confirmation message

7. `load\_model(filename="random\_forest\_heart\_model.pkl")`: Load the model

* Load the model from the specified filename using joblib.load
* Print confirmation message
* Return the loaded model

8. `**check\_new\_data\_from\_json(model, json\_file="heart\_data.json")**

* Read a new patient's data from the given JSON file (default: "heart\_data.json").
* Convert the patient's data into a pandas DataFrame.
* Use the trained model to predict whether the patient has heart disease.
* Calculate the prediction confidence if the model supports predict\_proba().
* Identify high-risk factors manually based on patient's input:
  + Age > 60 years
  + Chest pain type 3 or 4
  + Cholesterol > 240
  + ST depression (oldpeak) > 2.0
  + More than one blocked vessel (ca >= 2)
  + Abnormal thalassemia (thal >= 2)
* If **3 or more risk factors** are present or **significant high-risk** indicators are detected, manually set prediction to **heart disease** even if model predicts no disease.
* **Print clear information in the following order:**
  + JSON file name is being checked
  + Model prediction output
  + Risk factors (if any) identified
  + Any adjustments made to the prediction
  + Final diagnosis for the patient: "Heart Disease" or "No Heart Disease"
* If manual adjustment is applied, print a warning that model prediction was overridden.
* Print the patient's input features in a readable tabular format.
* Handle any exceptions and print a clear error message if the file can't be loaded or processed

**Expected Output Format:**

Loading dataset...

Loaded xxx records.

Preprocessing data...

Features and target separated.

Splitting data...

Train: x42, Test: x1

Creating Random Forest model...

Training model...

Training complete.

Saving model as 'random\_forest\_heart\_model.pkl'...

Model saved.

You, as the test taker have to take heart\_data.json from the dataset to check for heart disease

**Sample Input:🡪This is only for your reference**

age  sex  cp  trestbps  chol  fbs  restecg  thalach  exang  oldpeak  slope  ca  thal

 63    1   3       145   233    1        0      150      0      2.3      0   0     1

Model Confidence Level: 0.X1

Prediction: x --> Heart Disease or Not 🡪 data checked for heart\_data.json

FINAL HEART DISEASE PREDICTION RESULT:

Patient has heart disease: YES/NO

Risk factors identified:

- Severe chest pain (type 3-4)

- Significant ST depression (>2.0)

- Multiple blocked vessels

- Abnormal thalassemia

Diagnosis: The patient is predicted to have heart disease based on the provided features.

**Task 2: Chicken Disease Prediction**

You will implement the following functions in `chicken.py`:

1. `load\_chicken\_disease\_data()`: Load the chicken disease dataset

   - Load the CSV file "chicken\_disease\_data.csv"

   - Limit to 1000 rows

   - Print confirmation message with the number of records loaded

   - Return the loaded DataFrame

2. `perform\_eda\_on\_age(df)`: Perform exploratory data analysis on the Age column

   - Count chickens with age > 2

   - Print the count

   - Return nothing

3. `preprocess\_chicken\_data(df)`: Preprocess the chicken disease data

   - Convert categorical features to dummy variables using pd.get\_dummies with drop\_first=True

   - Verify that "Disease Predicted\_Healthy" column exists after encoding

   - Separate features (X), target (y), and keep the encoded DataFrame

   - X should contain all columns except "Disease Predicted\_Healthy"

   - y should contain only the "Disease Predicted\_Healthy" column

   - Print confirmation message

   - Return X, y, and the encoded DataFrame

4. `split\_chicken\_data(X, y, test\_size=0.2)`: Split the data into training and testing sets

   - Use train\_test\_split with test\_size=0.2 and random\_state=42

   - Print confirmation message with the size of training and testing sets

   - Return X\_train, X\_test, y\_train, y\_test

5. `create\_and\_train\_model(X\_train, y\_train)`: Create and train a Decision Tree model

   - Create a DecisionTreeClassifier with random\_state=42

   - Fit the model to the training data

   - Print confirmation message

   - Return the trained model

6. `calculate\_entropy(y)`: Calculate the entropy of the target column

* + Calculate entropy using the formula: -sum(p \* log2(p))
  + Print the entropy value
  + Return nothing

7 check\_new\_data\_from\_json(model, df\_encoded, json\_file="chicken\_data.json"):

* Read new chicken data from the specified JSON file (default: "chicken\_data.json").
* Extract the chicken's information and process it into a DataFrame matching the training features.
* Load the original chicken disease dataset to ensure encoding consistency.
* Append the new chicken’s data to the dataset and re-encode all data using pd.get\_dummies(drop\_first=True).
* Extract the new chicken's feature row correctly aligned with the training format.
* Use the trained model to predict if the chicken is healthy or diseased.

Print the following:

* Chicken input features clearly.
* Initial model prediction ("Healthy" or "Diseased").

Determine likely disease type using basic rules:

* If high fever, coughing, and bloody feces → Predict **Avian Influenza**.
* If decreased eating and high temperature → Predict **Coccidiosis**.
* If breed is Plymouth Rock and high temperature → Predict **Marek's Disease**.

Handle any exceptions (file not found, missing keys, etc.) and print a friendly error message

**Sample output screen**

Model Prediction: False --> Diseased'

Likely Disease Type: Avian Influenza

FINAL CHICKEN DISEASE PREDICTION RESULT:

Chicken is healthy: NO

Likely Disease Type: Avian Influenza

Diagnosis: The chicken is predicted to have a disease based on the provided features.

**Running the Tests**

To run the tests, use the following command:

Python3 -m unittest

**Submission Guidelines**

1. Complete all the required functions in `heart.py` and `chicken.py`

2. Ensure all tests pass

3. Submit your code files

**Execution Steps to Follow:**

* + All actions like build, compile, running application, running test cases will be through Command Terminal.
  + To open the command terminal the test takers, need to go to Application menu (Three horizontal lines at left top) -> Terminal -> New Terminal
  + This editor Auto Saves the code
  + If you want to exit(logout) and continue the coding later anytime (using Save & Exit option on Assessment Landing Page) then you need to use **CTRL+Shift+B** -command compulsorily on code IDE. This will push or save the updated contents in the internal git/repository. Else the code will not be available in the next login.
  + These are time bound assessments the timer would stop if you logout and while logging in back using the same credentials the timer would resume from the same time it was stopped from the previous logout.
  + To setup environment:

You can run the application without importing any packages

* + To launch application:
  + **Python3 heart.py` and `python3 chicken.py`**
  + To run Test cases:

**python3 -m unittest**

* + Before Final Submission also, you need to use **CTRL+Shift+B** - command compulsorily on code IDE, before final submission as well. This will push or save the updated contents in the internal git/repository, and will be used to evaluate the code quality.

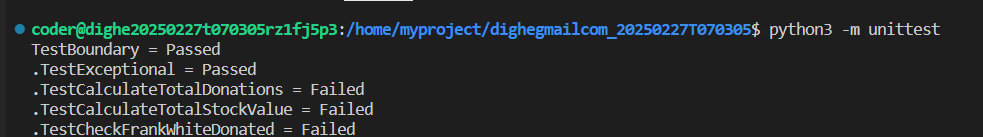
**Screen shot to run the program**

**To run the application**

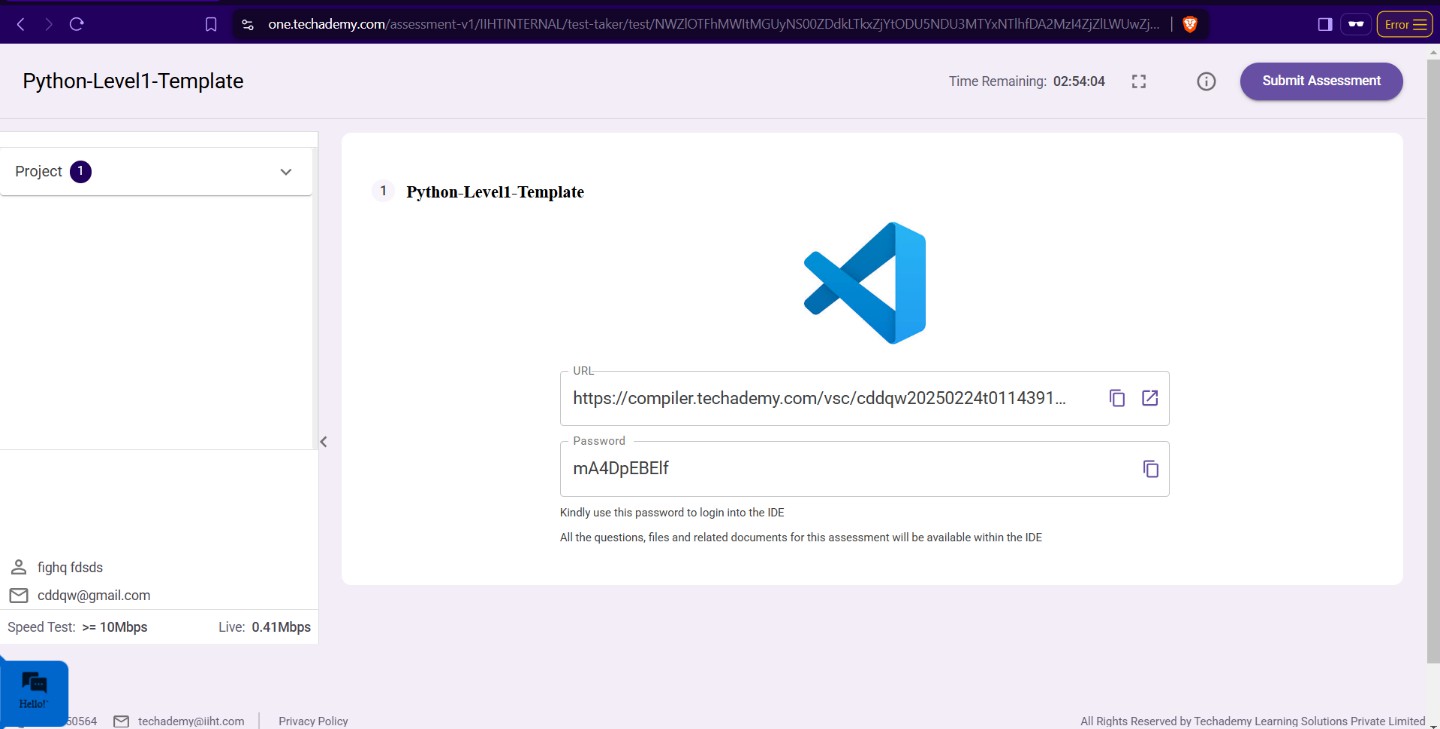


**Python3 heart.py**

**python3 chicken.py`**



**To run the testcase python3 -m unittest**



* + **Once you are done with development and ready with submission, you may navigate to the previous tab and submit the workspace. It is mandatory to click on “Submit Assessment” after you are done with code.**