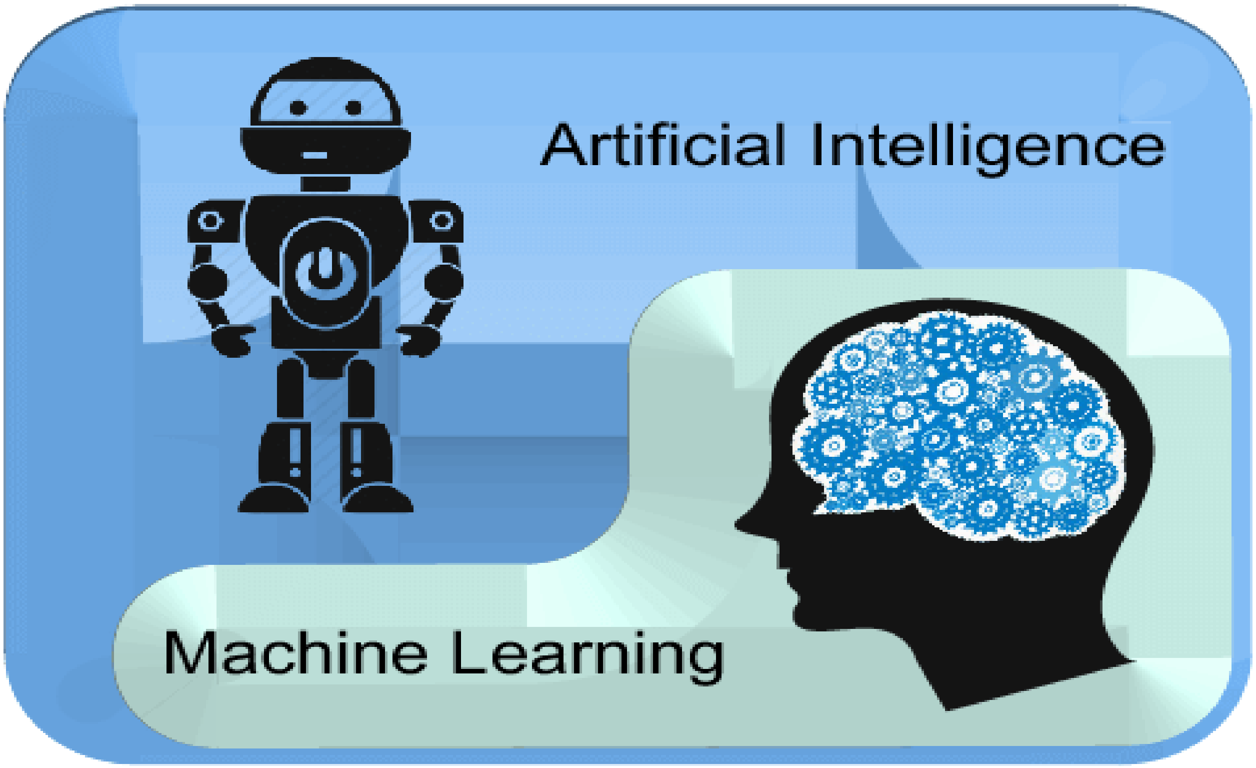
DAYANANDA SAGAR UNIVERSITY
SCHOOL OF ENGINEERING

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
Artificial Intelligence & Machine Learning

Artificial Intelligence-II Laboratory Manual
20AM3603



Dayananda Sagar University
Innovation City Campus, Hosur Main Road, Kudlu Gate, Bangalore, India,
Karnataka -560068

DAYANANDA SAGAR UNIVERSITY
SCHOOL OF ENGINEERING

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING
Artificial Intelligence & Machine Learning

Artificial Intelligence-II Laboratory Manual
20AM3603

NAME: ....

USN: ....

Semester: ....

Dayananda Sagar University
Innovation City Campus, Hosur Main Road, Kudlu Gate, Bangalore, India,
Karnataka -560068

Vision

- To Develop highly competent engineers in the field of AI & ML contributing globally to the benefit of industry and society.

Mission

- To develop state-of-the-art academic and infrastructural facilities with the latest tools and other learning resources supported by curriculum that can produce self-sustainable professionals.
- To emerge as a research centre that interacts with industry on a regular basis for imparting wholistic curriculum to the students.
- To impart emerging skill sets that the industry require, apart from ensuring that the soft skills of the learners are given adequate thrust.

Values

The values that drive DSU and support its vision:

The Pursuit of Excellence

- A commitment to strive continuously to improve ourselves and our systems with the aim of becoming the best in our field.

Fairness

- A commitment to objectivity and impartiality, to earn the trust and respect of society.

Leadership

- A commitment to lead responsively and creatively in educational and research processes.

Integrity and Transparency

- A commitment to be ethical, sincere and transparent in all activities and to treat all individuals with dignity and respect.

DAYANANDA SAGAR UNIVERSITY

Laboratory Certificate

This is to certify that, Mr./Ms. ______________________________ Bearing University Seat number (USN) __________________ has satisfactorily completed the experiments in above practical subject prescribed by the University for the __________semester B.tech program in the Artificial Intelligence -II Laboratory of this university during the year 2022-2023 (Even Sem).

Date: ___________________

MARKS	
Maximum	Obtained

____________________________ ____________________________

Signature of the Faculty incharge Signature of the Chairman

Instructions for LaboratoryExercises:

1. The programs with comments are listed for your reference. Write the programs in observation book.
2. Create your own subdirectory in the computer. Edit (type) the programs with program number and place them in your subdirectory.
3. Execute the programs as per the steps discussed earlier and note the results in your observation book.
4. Initially you will start with PYTHON & ANACONDS tools, execute the program.
5. You can also use Google clab, jyupter notebook for execution of the program.
6. Please include program output screen for every program.

List of Experiments

Exp No	Experiment Name	Date	Marks	Sign
1	Design & analyze the application of Artificial Intelligence for Graph Theory concept.			
2	For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent.			
3	Write a python program to remove punctuation's from the given string?			
4	Implement naive Bayes theorem to classify the English text.			
5	Implement the finite words classification system using back propagation algorithm.			
6	To implement the model to correctly identify the sentiments of the users by reviews which is an English paragraph and the result will be in positive or negative only. “NLP - Sentiment Analysis - Restaurant Reviews”.			
7	Write a Python program to implement Named Entity recognition.			
8	Handwritten recognition using MNSIT.			
9	Prediction of Humidity using DHT11 temperature sensor & Raspberry- pi interfacing With Python Programming			

Internals (40M)	**Lab Record & Attendance (20M)**	**Total Marks (60M)**

Signature of the Student **Signature of the Staff**

EXPERIMENT: 1

AIM: Design & analyze the application of Artificial Intelligence for Graph Theory concept.

PROGRAM:

```
class City:
    def _init_(self, name):
        self.name = name
        self.connection = {}

    def addConnection(self, city, distance):
        self.connection[city] = distance

class StateSpaceGraph:
    def _init_(self):
        self.cities = {}

    def addCities(self, name):
        city = City(name)
        self.cities[name] = city

    def addConnection(self, city1, city2, distance):
        self.cities[city1].addConnection(self.cities[city2], distance)
        self.cities[city2].addConnection(self.cities[city1], distance)

    def shortestPath(self, start, end):
        distances = {city: float('inf') for city in self.cities}
        distances[start] = 0
        visited = set()
        unvisited = set(self.cities.values())

        while unvisited:
            currentCity = min(unvisited, key=lambda city: distances[city.name])
            unvisited.remove(currentCity)
            visited.add(currentCity)

            for neighbor, distance in currentCity.connection.items():
                if neighbor in visited:
                    continue

            newDistance = distances[currentCity.name] + distance
            if newDistance < distances[neighbor.name]:
```

```
        distances[neighbor.name] = newDistance

    return distances[end]

graph = StateSpaceGraph()
graph.addCities('A')
graph.addCities('B')
graph.addCities('C')
graph.addCities('D')
graph.addConnection('A', 'B', 5)
graph.addConnection('A', 'C', 3)
graph.addConnection('B', 'C', 2)
graph.addConnection('B', 'D', 4)
graph.addConnection('C', 'D', 1)

firstCity = input((f"Enter the starting city: "))
secondCity = input((f"Enter the destination city: "))
print(f"The shortest distance between the cities {firstCity} and {secondCity} is {graph.shortestPath(firstCity, secondCity)}.")
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS

Staff Signature with date

EXPERIMENT: 2

AIM: For a given set of training data examples stored in a .CSV file, implement and demonstrate the Candidate-Elimination algorithm to output a description of the set of all hypotheses consistent.

PROGRAM:

```
import pandas as pd
file = "2_EnjoySports.csv" # Replace with 2_CarSales.csv
df = pd.read_csv(f"Datasets/{file}", header = 0)
dataset = df.values.tolist()

s = dataset[0][0:-1]
print(f"The initial value of specific hypothesis is {s}")
#initialize the general hypothesis
g = [['?' for _ in range(len(s))] for _ in range(len(s))]
print(f"The initial value of general hypothesis is {g}")
for index, row in enumerate(dataset):
    if row[-1] == "Yes":
        for j in range(len(s)):
            if row[j] != s[j]:
                s[j] = '?'
                g[j][j] = '?'
    elif row[-1] == "No":
        for j in range(len(s)):
            if row[j] != s[j]:
                g[j][j] = s[j]
            else:
                g[j][j] = "?"
    print(f"Instance: {index+1}")
    print(f"Specific boundary is: {s}")
    print(f"General boundary is: {g}")
print(f"Final specific hypothesis: {s}")
print(f"Final general hypothesis: {g}")
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS:

Staff signature with date

EXPERIMENT: 3

AIM: Write a python program to remove punctuations from the given string?

PROGRAM:

```
import string
with open("Datasets/3_Paragraph.txt", "r") as file:
    textList = file.readlines()
    text = ""
    for element in textList:
        text += element
    file.close()

print(f"The original contents of the file are:\n {text}")

punctuations = string.punctuation

countPunc, countSpaces, countReplace = 0, 0, 0

repeat = True

while repeat:
    choice = int(input("1. Remove Punctuation marks from the text\n2. Count
the number of spaces in the text\n3. Replace a punctuation mark with a specific
Character.\nEnter your choice: "))

    match choice:
        case 1:
            for element in text:
                if element in punctuations:
                    text = text.replace(element, "")
                    countPunc += 1
            print(f"\nThe contents of the file after filtering the
punctuations are:\n {text}")
            print(f"The number of punctuation marks in the text
are:\n\t{countPunc}")
        case 2:
            for element in text:
                if element == " ":
                    countSpaces += 1
            print(f"The number of spaces in the text
are:\n\t{countSpaces}")
        case 3:
            punc = input("\nEnter the punctuation character: ")
            replace = input("\nEnter the replacement character: ")

            if punc in punctuations:
                for element in text:
                    if element == punc:
                        text = text.replace(punc, replace)
                        countReplace += 1
                print(f"\nThe contents of the file after replacing \'{punc}\' with \'{replace}\' are:\n {text}")
                print(f"\nThe number of characters replaced are:\n\t{countReplace}")
            else:
                print("Please enter a valid punctuation mark.")
    repeat = int(input("\nWould you like to repeat? \t1. Yes 0. No\n"))
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS:

Staff signature with date

EXPERIMENT: 4

AIM: Implement naïve bayes theorem to classify the English text.

PROGRAM:

```
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB

X_train = ['I love playing sports', 'Politics is my passion', 'Sports are fun', 'Politics is boring']
y_train = ['sports', 'politics', 'sports', 'politics']
vectorizer = CountVectorizer(stop_words='english')
X_train_counts = vectorizer.fit_transform(X_train)

model = MultinomialNB()
model.fit(X_train_counts, y_train)

X_test = ['I enjoy watching sports on TV', 'Politics is important for our country', 'i love politics']
X_test_counts = vectorizer.transform(X_test)
y_pred = model.predict(X_test_counts)

output = []
for index, i in enumerate(range(len(X_test))):
    output.append([X_test[i], y_pred[i]])
    print(output[index])
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS:

Staff signature with date

EXPERIMENT: 5

AIM: Implement the finite words classification system using back propagation algorithm.

PROGRAM:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import accuracy_score, confusion_matrix, precision_score,
recall_score

data = pd.read_csv('Datasets/5_FiniteWords.csv', names=['Message', 'Label'])
print("The Total instances in the Dataset: ", data.shape[0])
print(data)
data['labelnum'] = data.Label.map({'pos': 1, 'neg': 0})
print(data)
X = data["Message"]
y = data.labelnum
X_train, X_test, y_train, y_test = train_test_split(X, y)
count_vect = CountVectorizer()
X_train_dims = count_vect.fit_transform(X_train)
X_test_dims = count_vect.transform(X_test)
model = MLPClassifier(solver='lbfgs', alpha=1e-5, hidden_layer_sizes=(5, 2),
random_state=1)
model.fit(X_train_dims, y_train)
prediction = model.predict(X_test_dims)
print('***Accuracy Metrics****')
print(f'Accuracy: {accuracy_score(y_test, prediction)}')
print(f'Recall: {recall_score(y_test, prediction)}')
print(f'Precision: {precision_score(y_test, prediction)}')
print(f'Confusion Matrix : \n{confusion_matrix(y_test, prediction)}')
print(10*"-")

test_stmt = [input("Enter any statement to predict: ")]
test_dims = count_vect.transform(test_stmt)
pred = model.predict(test_dims)
for stmt, lbl in zip(test_stmt, pred):
    if lbl == 1:
        print("Statement is Positive")
    else:
        print("Satement is Negative")
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS: Staff signature and date

EXPERIMENT: 6

AIM: To implement the model to correctly identify the sentiments of the users by reviews which is an English paragraph and the result will be in positive or negative only. “NLP - Sentiment Analysis - Restaurant Reviews”.

PROGRAM:

```
import numpy as np
import pandas as pd
# import nltk
# nltk.download('stopwords') # Uncomment these lines if you haven't downloaded
'stopwords'
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
import re
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import accuracy_score, precision_score, recall_score,
confusion_matrix
import matplotlib.pyplot as plt
import seaborn as sns

file_path = 'Datasets/6_RestaurantReviews.tsv'
df = pd.read_csv(file_path, sep='\t', header=0)

corpus = []
for i in range(0, 1000):
    review = re.sub(pattern='[^a-zA-Z]',repl=' ', string=df['Review'][i])
    review = review.lower()
    review_words = review.split()
    review_words = [word for word in review_words if not word in
set(stopwords.words('english'))]
    ps = PorterStemmer()
    review = [ps.stem(word) for word in review_words]
    review = ' '.join(review)
    corpus.append(review)

cv = CountVectorizer(max_features=1500)
X = cv.fit_transform(corpus).toarray()
y = df.iloc[:, 1].values
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20,
random_state=0)
classifier = MultinomialNB()
```

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

accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
print("---- Scores ----")
print(f"Accuracy score is: {round(accuracy*100, 2)}%")
print(f"Precision score is: {round(precision,2)}")
print(f"Recall score is: {round(recall,2)}.")

cm = confusion_matrix(y_test, y_pred)

plt.figure(figsize=(10,6))
sns.heatmap(cm, annot=True, cmap="YlGnBu", xticklabels=['Negative',
'Positive'], yticklabels=['Negative', 'Positive'])
plt.xlabel('Predicted values')
plt.ylabel('Actual values')
plt.show()

best_accuracy = 0.0
alpha_val = 0.0
for i in np.arange(0.1, 1.1, 0.1):
    temp_classifier = MultinomialNB(alpha=i)
    temp_classifier.fit(X_train, y_train)
    temp_y_pred = temp_classifier.predict(X_test)
    score = accuracy_score(y_test, temp_y_pred)
    print(f"Accuracy score for alpha={round(i, 1)} is: {round(score*100, 2)}%")
    if score > best_accuracy:
        best_accuracy = score
        alpha_val = i
print('------------------------------------------------')
print(f'The best accuracy is {round(best_accuracy*100, 2)}% with alpha value as
{round(alpha_val, 1)}')

classifier = MultinomialNB(alpha=0.2)
classifier.fit(X_train, y_train)

def predict_sentiment(sample_review):
    sample_review = re.sub(pattern='[^a-zA-Z]',repl=' ', string=sample_review)
    sample_review = sample_review.lower()
    sample_review_words = sample_review.split()
    sample_review_words = [word for word in sample_review_words if not word in
set(stopwords.words('english'))]
    ps = PorterStemmer()
    final_review = [ps.stem(word) for word in sample_review_words]
    final_review = ' '.join(final_review)
    temp = cv.transform([final_review]).toarray()
    return classifier.predict(temp)

sample_review = input('Enter a sample review: ')

if predict_sentiment(sample_review):
    print('This is a POSITIVE review!')
else:
    print('This is a NEGATIVE review!')
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS: Staff signature and date

EXPERIMENT: 7

AIM: Write a Python program to implement Named Entity recognition.

PROGRAM:

```
import nltk
# nltk.download('punkt')
# nltk.download('averaged_perceptron_tagger')
# nltk.download('maxent_ne_chunker')
# nltk.download('words') # Uncomment these lines if the specified files aren't downloaded

paragraph = """Scarface is a 1983 American crime drama film directed by Brian De Palma and written by Oliver Stone. \
   Loosely based on the 1929 novel of the same name and serving as a loose remake of the 1932 film,\
   it tells the story of Cuban refugee Tony Montana (Al Pacino), \
   who arrives penniless in Miami during the Mariel boatlift and becomes a powerful and extremely homicidal drug lord.\
   The film co-stars Steven Bauer, Michelle Pfeiffer, Mary Elizabeth Mastrantonio and Robert Loggia.\
   De Palma dedicated this version of Scarface to the memories of Howard Hawks and Ben Hecht, \
   the writers of the original film."""

for sentence in nltk.sent_tokenize(paragraph):
   for chunk in nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sentence))):
      if hasattr(chunk, 'label'):
         print(f"{' '.join(c[0] for c in chunk):<35} {chunk.label()}")
```

INPUT DATA:

OUTPUT DATA:

VIVA QUESTIONS:

Staff signature with date

EXPERIMENT: 8

AIM: Write a Python program to recognize handwritten numbers from the MNIST dataset using Tensorflow.

PROGRAM:

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.datasets import mnist
import matplotlib.pyplot as plt
import numpy as np
import random
(X_train, y_train), (X_test, y_test) = mnist.load_data()
tem = random.randint(1, 1000)
images = X_train[tem]
labels = y_train[tem]
X_train = X_train.reshape(X_train.shape[0], 28, 28, 1)
X_test = X_test.reshape(X_test.shape[0], 28, 28, 1)
X_train = X_train.astype('float32') / 255.
X_test = X_test.astype('float32') / 255.

model = Sequential([
    Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(28, 28, 1)),
    MaxPooling2D(pool_size=(2, 2)),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.5),
    Dense(10, activation='softmax')
])
model.compile(loss='sparse_categorical_crossentropy', optimizer='adam', metrics=['accuracy'])
model.fit(X_train, y_train, batch_size=128, epochs=5, verbose=1, validation_data=(X_test, y_test))
score = model.evaluate(X_test, y_test, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])
predictions = model.predict(X_test)

x = model.predict(images.reshape(1, 28, 28, 1))
plt.imshow(images)
plt.show()
np.argmax(x, axis=1)
number = np.where(x==1.)[1]
number = str(number).lstrip('[').rstrip(']')
print(f' The model has predicted that the image is of the number {number}.')
```

INPUT DATA:

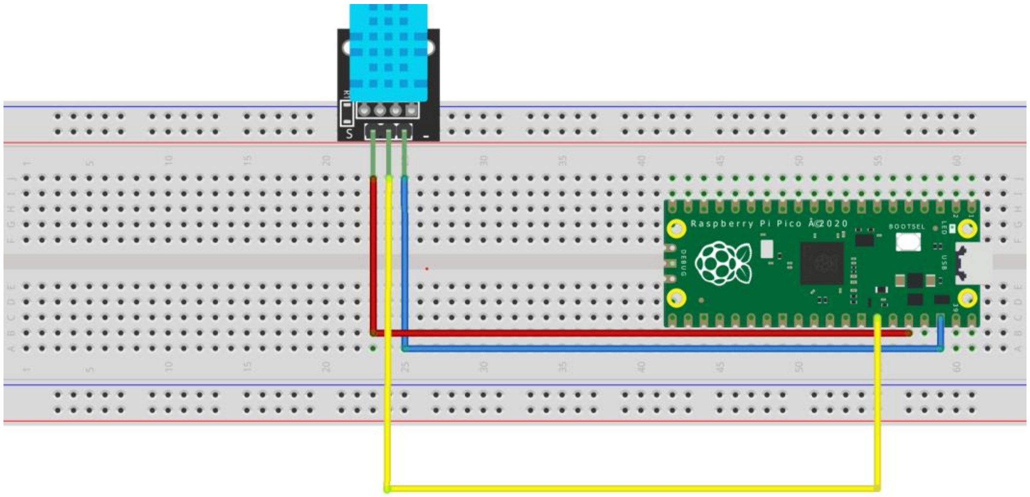
OUTPUT DATA:

VIVA QUESTIONS:

Staff signature with date

Experiment 9

Aim: Prediction of Humidity using DHT11 temperature sensor & Raspberry- pi interfacing With Python Programming



1. Connect the VCC pin of the DHT11 sensor to the VBUS pin on the Pico. The VBUS pin can be found on the top-left corner of the Pico board.
2. Connect the GND pin of the DHT11 sensor to any of the GND pins on the Pico.
3. Connect the data pin of the DHT11 sensor to any of the GP pins on the Pico. For example, you can connect it to GP0.

Algorithm:

Step 1: Connect the circuit as shown in the figure.

Step 2: Collect data using sensors and store it in a text file.

Step 3: Load the data from the text file using Pandas.

Step 4: Split the data into input (X) and output (y) variables. Input is temperature and output is humidity.

Step 5: Create a Linear Regression model using Scikit-Learn and fit it to the data.

Step 6: Save the trained model to a file in json format.

Step 7: On the Raspberry Pi Pico Specify the model coefficient and intercept of the trained model to predict the output variable based on the input variable and Display the predicted output value.

Training Dataset

```
import pandas as pd
from sklearn.linear_model import LinearRegression
import json
```

```
# Load the data from the CSV file
data = pd.read_csv('data.txt')

# Split the data into input (X) and output (y) variables
X = data[['Temperature']]
y = data['Humidity']

# Create a linear regression model and fit it to the data
model = LinearRegression()
model.fit(X, y)

# Extract the coefficients and intercept from the trained model
coefficients = model.coef_[0]
intercept = model.intercept_

# Create a dictionary to store the model data
model_data = {
    'coefficients': coefficients,
    'intercept': intercept
}

# Save the model data to a JSON file
with open('model.json', 'w') as f:
    json.dump(model_data, f)
```

Collect.Py

```
import machine
import dht
import time

# Create a DHT11 object and specify the GPIO pin
d = dht.DHT11(machine.Pin(0))

# Open the data file for writing
with open('data.txt', 'w') as f:

    # Write the header row
    f.write('Temperature,Humidity\n')

    i = 1000
    # Continuously read data from the DHT11 sensor and write it to the file
    while i >= 0:
        # Read the temperature and humidity from the DHT11 sensor
        d.measure()
        temperature = d.temperature()
        humidity = d.humidity()
        print(f"Temperature: {temperature} c")
        print(f"Humidity: {humidity}%")
        print()

        # Write the temperature and humidity to the file
        f.write(f'{temperature},{humidity}\n')

        # Wait for 1 second before taking the next reading
        time.sleep(1)
        i = i - 1
```

Predict.Py

```
import machine
import dht
import utime

# Specify the model coefficients and intercept
coefficients = -2.8212674826885054
intercept = 151.34381063636422

# Create a DHT11 object and specify the GPIO pin
d = dht.DHT11(machine.Pin(0))
# Continuously read data from the DHT11 sensor and predict the humidity using the loaded model
while True:
    # Read the temperature from the DHT11 sensor
    d.measure()
    temperature = d.temperature()
    true_humidity = d.humidity()
    # Predict the humidity using the loaded model
    start_time = utime.ticks_us()
    humidity = temperature * coefficients + intercept
    end_time = utime.ticks_us()
    execution_time_us = utime.ticks_diff(end_time, start_time)
    # Print the predicted humidity and the execution time
    print(f"Temperature: {temperature} C")
    print(f"Predicted Humidity: {humidity:.0f} %")
    print(f"True Humidity: {true_humidity:.0f} %")
    print(f"Execution Time: {execution_time_us} µs")
    print()
    # Wait for 1 second before taking the next reading
    utime.sleep(1)
```