Machine Learning Internship Manual # 01

Submitted by:

Nabiha Fatima

Phone number:

03211156270

Intern ID:

ARCH-2506-1434

PROJECT INTRODUCTION:

This project marks the beginning of my journey into Machine Learning (ML), designed for beginners with little or no prior coding experience. It consists of two foundational tasks aimed at building a solid understanding of Python programming and Machine Learning concepts through guided learning and a hands-on project.

Task 01: Learning Python and Tools

To start, I focused on learning the basics of Python—like loops, variables, and functions—through beginner-friendly YouTube tutorials (Apna College and CodeWithHarry). I practiced coding using **Programiz** and later moved to **Google Colab**, an online platform made for ML projects. I also used **W3Schools** to explore topics in more detail.

Task 02: Understanding Machine Learning Concepts

Next, I read **Chapter 1** of *Hands-On Machine Learning* by *Aurélien Géron.* It gave me a strong base in key ML ideas like supervised/unsupervised learning, model types, challenges in data, and hyper parameter tuning. I also solved all 19 end-of-chapter exercises to test my understanding.

Mini Project: Crop Prediction App

To apply what I learned, I built a simple **Crop Prediction web app** using a tutorial by **AlgoChat**. It predicts the best crop to grow based on input conditions. This hands-on project helped me understand the full ML workflow—training a model and deploying it as a web app using **Flask**.

Visuals

Code with Explanations

1. Google Colab:

```
# Import necessary libraries import pickle  # Used for
saving the trained model to a file import pandas as pd  #
For data manipulation and analysis
from sklearn.model_selection import train_test_split  # For splitting data into training and
testing sets
from sklearn.ensemble import RandomForestClassifier  # Random Forest algorithm for
classification

# Load dataset (CSV must be in same directory or full path should be given)
data = pd.read_csv("Crop_recommendation.csv")

# Display first 5 rows of the dataset
print(data.head())

# Display shape of the dataset (rows, columns)
print("Shape:", data.shape)

# Check for any missing (null) values in the dataset
print("Missing values:\n", data.isnull().sum())
# Separate dataset into features (X) and labels (y) X
= data.iloc[:, :-1]  # All columns except the last y =
data.iloc[:, -1]   # Only the last column (crop name)
```

```
# Split the dataset into training and testing sets (80% train, 20% test)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Create an instance of the Random Forest Classifier
model = RandomForestClassifier()

# Train the model on training data
model.fit(X_train, y_train)

# Save the trained model to a .pkl file for future use
pickle.dump(model, open("model.pkl", "wb"))

# Use the trained model to predict labels for the test set
predictions = model.predict(X_test)

# Evaluate the model's accuracy on the test data
accuracy = model.score(X_test, y_test)
print("Accuracy:", accuracy)
```

2. Flask App

```
# Import necessary libraries
import numpy as np  # For array/matrix operations
from flask import Flask, request, render_template  # Flask functions
import pickle  # For loading the saved ML model

# Initialize Flask app
flask_app = Flask(__name__)

# Load trained model from the saved pickle file
model = pickle.load(open("model.pkl", "rb"))

# Route for homepage (GET method)
@flask_app.route("/") def Home():     return render_template("index.html")
# Renders the HTML file with input form

# Route for prediction functionality (POST
method) @flask_app.route("/predict",
methods=["POST"]) def predict():

        # Extract form values from HTML, convert to float
```

```
    float_features = [float(x) for x in request.form.values()]

    # Convert the list of features into a NumPy array (2D for model input)
features = [np.array(float_features)]

    # Get prediction from the loaded model
prediction = model.predict(features)

    # Extract prediction result
    predicted_crop = prediction[0]

    # Send prediction result back to the same HTML page     return
render_template("index.html", prediction_text=f"The Predicted Crop is:
{predicted_crop}")

# Run the app in debug mode (helpful during
development) if __name__ == "__main__":
flask_app.run(debug=True)
```

3. CSS

```
/* Set background image for the webpage */ body {   background-
image: url("farm.jpeg");       /* Image of farm as background */
background-size: cover;                  /* Image stretches to fill the screen
*/   background-repeat: no-repeat;            /* Do not repeat image */
margin-top: 100px;                     /* Move content down */   margin-
left: 500px;                    /* Push content to the right */ }

/* Style the main heading (title) */
h1 {
  margin-left: 100px;                    /* Add space from the left for alignment */
}

/* Style input fields */ input {   width: 70%;                              /*
Input width as 70% of parent */   padding: 12px 20px;
/* Padding inside inputs */   margin: 8px 0;                           /*
Space above and below inputs */
  box-sizing: border-box;                 /* Ensures border/padding don’t affect width */
border-bottom: 4px solid black;          /* Bottom border to style input */
}
/* Style the 'Predict' button *
```

```
button {   background-color: #4CAF50;              /* Green
background */   margin: auto;                      /* Center it
horizontally */   color: white;                    /* White text
*/   padding: 15px 100px;                 /* Internal spacing */
text-align: center;                   /* Center text inside */
font-size: 16px;                      /* Font size */
  margin-left: 170px;                  /* Adjust alignment
*/ }

/* Style the prediction result */
#predi {

  margin-left: 60px;                   /* Slight indentation */

 color: white;                        /* White text for visibility */ }
```

4. HTML

```
<!DOCTYPE html>

<html>

<head>

  <meta charset="UTF-8">  <!-- Set character encoding -->

  <title>ML API</title>  <!-- Title shown on browser tab -->



  <!-- Google Fonts for styling -->

  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet'>

  <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet'>

  <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet'>   <link
href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300'
rel='stylesheet'>
```

```
  <!-- Link to custom CSS file -->
  <link rel="stylesheet" href="{{ url_for('static', filename='style.css') }}"> </head>

<body>
  <!-- Main container for the form -->
  <div class="login">
    <h1>Crop Prediction Model</h1> <!-- Page title -->

    <!-- Form to collect input from user -->
    <form action="{{ url_for('predict') }}" method="post"> <!-- Sends data to /predict -->
<!-- Input fields for crop features -->
      <input type="text" name="Nitrogen" placeholder="Nitrogen" required />
      <input type="text" name="Phosphorus" placeholder="Phosphorus" required />
      <input type="text" name="Potassium" placeholder="Potassium" required />
      <input type="text" name="temperature" placeholder="Temperature" required />
      <input type="text" name="humidity" placeholder="Humidity" required />
      <input type="text" name="pH" placeholder="pH" required />
      <input type="text" name="rainfall" placeholder="Rainfall" required />

      <!-- Submit button -->
      <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
```

```
    <br><br> <!-- Add space -->


    <!-- Section to show prediction result -->

    <h1 id="predi">{{ prediction_text }}</h1> <!-- Filled dynamically by Flask -->
</div>

</body>

</html>
```

Theory Tasks

Chapter 1: The Machine Learning Landscape

What Is Machine Learning?

Machine Learning (ML) is a field where computers learn from data instead of being explicitly programmed. It helps systems improve automatically through experience.

Where Is It Used?

- Spam filtering in emails
- Product recommendations (like Netflix or Amazon)
- Fraud detection in banking
- Voice assistants like Siri or Google Assistant

Types of ML Systems

1. Supervised Learning:
   Learns from labeled data (e.g., predicting house prices)
2. Unsupervised Learning:

Finds patterns in unlabeled data (e.g., customer segmentation).

3. Semi-supervised Learning:

Mix of both—uses a few labels and a lot of unlabeled data.

4. Reinforcement Learning:

Learns by interacting with an environment (e.g., game playing AI).

Key Concepts

- Model: What the algorithm builds from data.
- Training: Feeding data to the algorithm.
- Generalization: How well the model works on new, unseen data.

Challenges in ML

- Not enough data
- errors or missing values
- Overfitting: Too well fitted to training data, fails on new data.
- Underfitting: Model too simple, fails to learn patterns.

Chapter 1 Exercises – Solutions

1. How would you define Machine Learning?

Machine Learning is the science of getting computers to learn from data and improve with experience—without being directly programmed for every task.

2. Can you name four types of problems where it shines?

- Spam detection
- Product recommendations
- Image recognition
- Forecasting stock prices

3. What is a labeled training set?
It’s a dataset that includes both input data and the correct output (label) for each example, used to train supervised learning models.

4. What are the two most common supervised tasks?
- Classification (e.g., spam or not spam)
- Regression (e.g., predicting house prices)

5. Can you name four common unsupervised tasks?
- Clustering
- Visualization
- Dimensionality reduction
- Association rule learning

6. What type of Machine Learning algorithm would you use to allow a robot to walk in various unknown terrains?
Reinforcement Learning – the robot learns by trying actions and seeing the results.

7. What type of algorithm would you use to segment your customers into multiple groups?
Unsupervised learning, especially clustering algorithms like K-Means.

8. Would you frame the problem of spam detection as a supervised learning problem or an unsupervised learning problem?
Supervised learning, because we have many labeled examples of spam and non-spam emails.

9. What is an online learning system?
A system that can learn continuously from new data as it comes in, rather than all at once.

10. What is out-of-core learning?
A way to train models on data too large to fit in memory by using small data chunks.

11. What type of learning algorithm relies on a similarity measure to make predictions?
Instance-based learning (e.g., K-Nearest Neighbors).

12. What is the difference between a model parameter and a learning algorithm's hyperparameter?
- Parameters: learned from data (e.g., weights in a linear model).
- Hyperparameters: set before training (e.g., learning rate, number of neighbors).

13. What do model-based learning algorithms search for? What is the most common strategy they use to succeed? How do they make predictions?
They search for the best model parameters using a cost function (like minimizing error). Once trained, they use these parameters to make predictions on new data.

14. Can you name four of the main challenges in Machine Learning?
- Not enough data
- Poor data quality
- Overfitting
- Underfitting

15. If your model performs great on the training data but generalizes poorly to new instances, what is happening? Can you name three possible solutions?
It's overfitting.
Solutions:
- Use simpler model

- Use regularization
- Get more training data

16. What is a test set and why would you want to use it?
A set of data never used during training, used to evaluate how well your model performs on unseen data.

17. What is the purpose of a validation set?
Used to tune the model's hyperparameters without touching the test set.

18. What can go wrong if you tune hyperparameters using the test set?
You'll unknowingly overfit to the test set, making the final evaluation biased and less trustworthy.

19. What is repeated cross-validation and why would you prefer it to using a single validation set?
It splits the data into many training/validation sets, averaging results for more reliable performance metrics and less variance.

Video Demonstration:

https://youtu.be/WNEHjXarr34?si=Q3x6oDZEHSNgl_ny

GitHub Repository :

https://github.com/Nabihaa2/ML_Project