Machine Learning Internship
Manual 01

Prepared by:

ARCH TECHNOLOGIES

INTERN DETAILS
NAME: Humna Usman
PHONE NO: 03275880558
INTERN ID:ARCH-2506-0399

Project Overview

This task was created for beginners who are new to Python programming and Machine Learning. The main goal is to help learners build a strong foundation in Python and understand the basic concepts of Machine Learning through simple, guided steps.

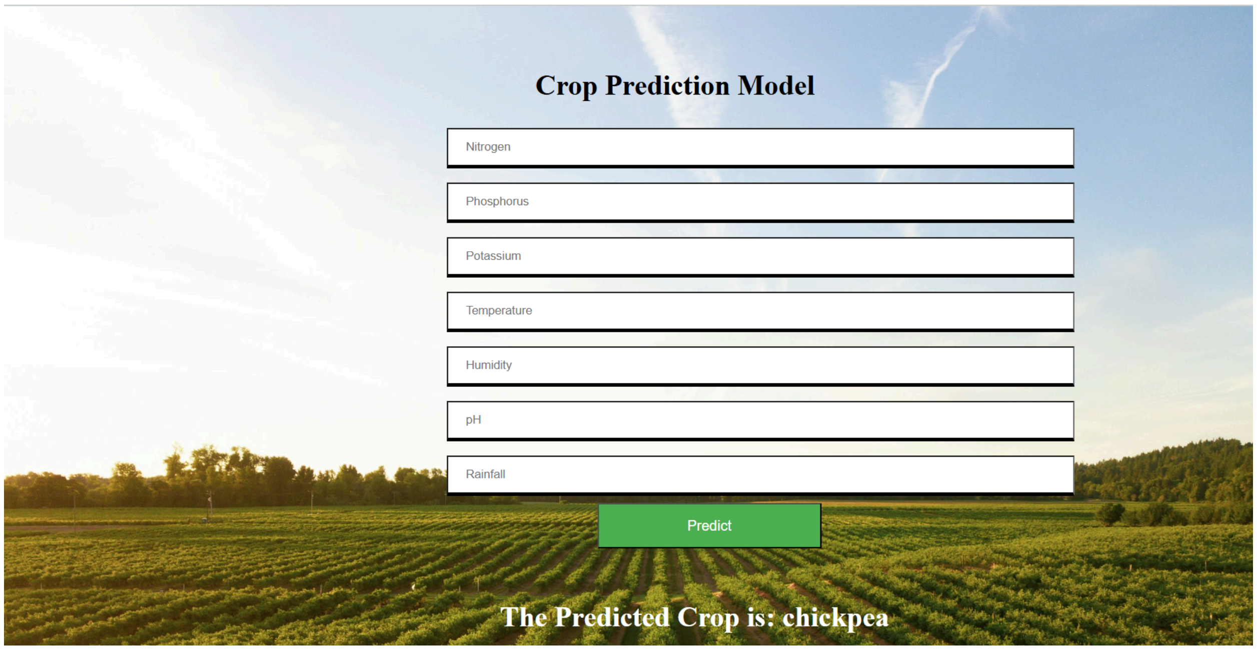
Objectives

- Learn Python programming from scratch using beginner-friendly videos and online resources.
- Understand basic coding concepts like variables, loops, functions, and operators.
- Get familiar with Google Colab, an online tool used to write and run Python code, especially for ML tasks.
- Read Chapter 1 of *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow* to learn about types of Machine Learning, key terms, challenges, and model selection.
- Complete the exercises from the chapter to test your understanding.
- Build your first ML project – a crop prediction web application using Python and Flask.

Approach

1. Start learning Python by watching beginner videos and practicing the code using an online compiler or Google Colab.
2. Use W3Schools and other resources to explore Python topics in more detail.
3. Read and take notes from Chapter 1 of the recommended book, and complete the exercises for better understanding.
4. Follow a guided tutorial to build and deploy a basic Machine Learning model in a web app.

PICTURES:



Code with Explanations

1.google collab

```
# 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

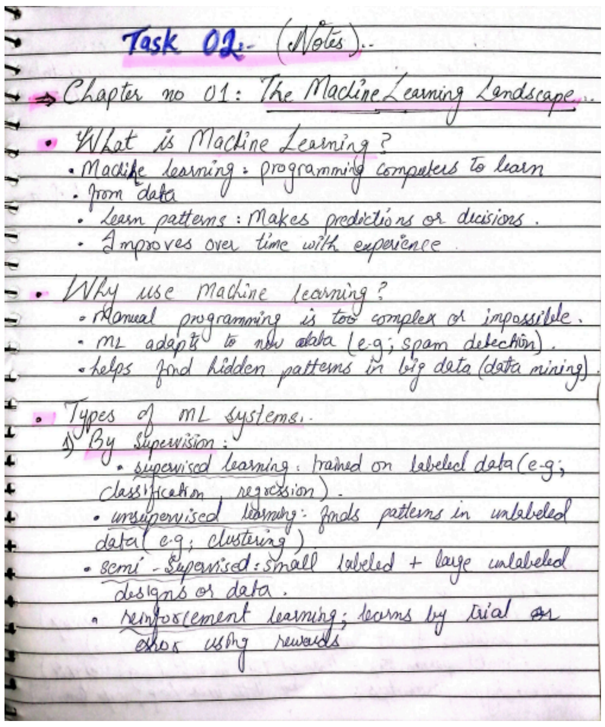
```
/* Set background image for the webpage */
body {
  background-image: url("farm.jpeg");       /* Image of farm as background */
  background-size: cover;                   /* Image stretches to fill the screen */
  background-repeat: no-repeat;             /* Do not repeat image */
  margin-top: 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:



Task 02:- (Notes)..

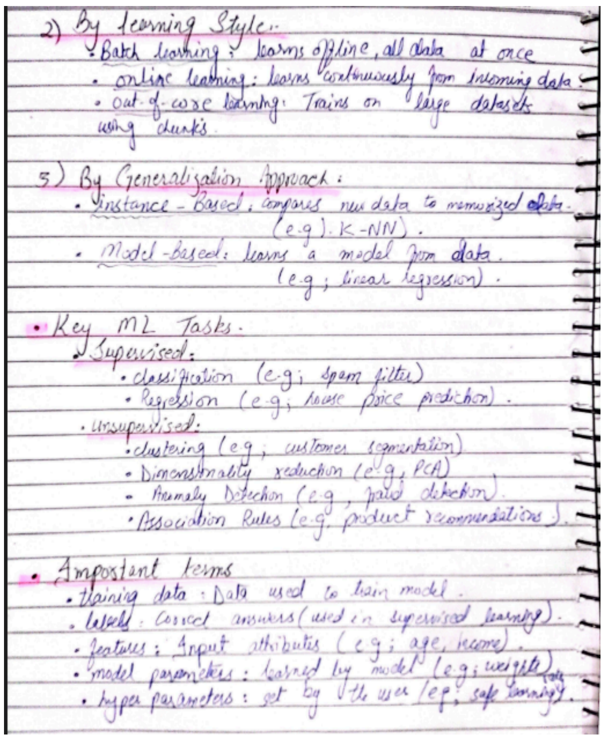
⇒ Chapter no 01: The Machine Learning Landscape..

• What is Machine Learning?
- Machine learning: programming computers to learn
- from data
- Learn patterns: Makes predictions or decisions.
- Improves over time with experience

• Why use Machine learning?
- Manual programming is too complex or impossible.
- ML adapts to new data (e.g; spam detection).
- helps find hidden patterns in big data (data mining).

• Types of ML systems..

1) By Supervision:
- supervised learning: trained on labeled data (e.g; classification, regression).
- unsupervised learning: finds patterns in unlabeled data (e.g; clustering)
- semi-Supervised: small labeled + large unlabeled designs or data.
- reinforcement learning; learns by trial ~~or~~ error using rewards

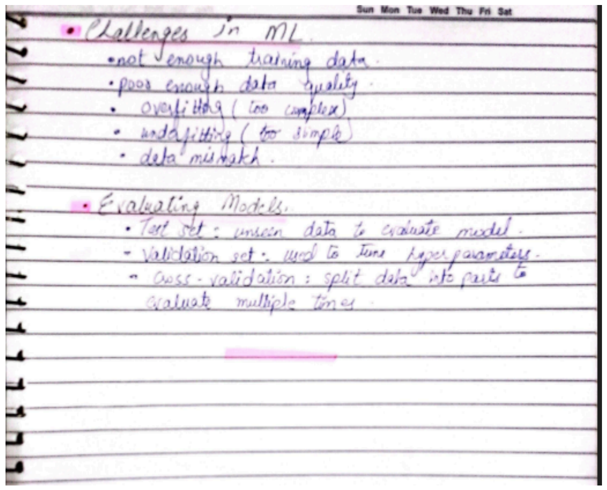


2) By learning Style:
- Batch learning: learns offline, all data at once
- online learning: learns continuously from incoming data.
- out-of-core learning: Trains on large datasets using chunks.

3) By Generalization Approach:
- instance-Based: compares new data to memorized data. (e.g). K-NN).
- Model-based: learns a model from data. (e.g; linear regression).

• Key ML Tasks.
- Supervised:
  - classification (e.g; spam filter)
  - Regression (e.g; house price prediction).
- unsupervised:
  - clustering (e.g; customer segmentation)
  - Dimensionality reduction (e.g, PCA)
  - Anomaly Detection (e.g, fraud detection).
  - Association Rules (e.g. product recommendations).

• Important terms
- training data: Data used to train model.
- labels: Correct answers (used in supervised learning).
- features: Input attributes (e.g; age, income).
- model parameters: learned by model (e.g; weights).
- hyper parameters: set by the user (e.g; [illegible] rate).

 Sun Mon Tue Wed Thu Fri Sat • Challenges in ML. • not enough training data. • poor enough data quality. • overfitting (too complex) • underfitting (too simple) • data mismatch. • Evaluating Models. • Test set : unseen data to evaluate model. • Validation set : used to tune hyperparameters. • Cross-validation : split data into parts to evaluate multiple times.	

Exercise questions from Chapter 1:

1. How would you define Machine Learning?

   Programming computers to learn from data without explicit programming.

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

   Spam detection, recommendations, speech recognition, trend forecasting.

3. What is a labeled training set?

   A dataset with input-output pairs used for training.

4. What are the two most common supervised tasks?

   Classification and regression.

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.

7. What type would you use to segment your customers into multiple groups?

   Clustering (unsupervised learning).

8. Would you frame the problem of spam detection as a supervised or unsupervised learning problem?

   Supervised learning.

9. What is an online learning system?

A system that learns incrementally from data streams.

10. What is out-of-core learning?

Training on data too large for memory using small batches.

11. What type of learning algorithm relies on a similarity measure to make predictions?

Instance-based learning.

12. What is the difference between a model parameter and a hyperparameter?

Parameters are learned; hyperparameters are set before training.

13. What do model-based learning algorithms search for? What is the goal of this search?

They find the best model to minimize prediction error.

14. Can you name four of the main challenges in Machine Learning?

Lack of 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?

Overfitting.

16. What is a test set, and why would you want to use it?

To evaluate final model performance.

17. What is the purpose of a validation set?

To tune model hyperparameters.

18. What is cross-validation?

A method to evaluate models by splitting the data into multiple parts.

19. What is the difference between supervised, unsupervised, and Reinforcement Learning?

Supervised uses labeled data; unsupervised uses unlabeled data; reinforcement learns via rewards.

Video Demonstration

LINK: https://youtu.be/ZTNh9phtnP0?si=vguUFf3rJn0qVEIR

GitHub Repository:

LINK: https://github.com/HumnaUsman/ML-project