🚀 Day 6 – Logistic Regression with CSV & Intro to K-Nearest Neighbors (KNN)

📅 **Date:** 1 July 2025

🔁 Logistic Regression – Revisited (Now with CSV!)

Today, we continued exploring **Logistic Regression**, but this time instead of using preloaded datasets, we imported the **Iris dataset from a CSV file** using pandas. This gave a much more realistic feeling — just like working on a real-world project!

We went through the **entire ML workflow**:

- Loading data
- Splitting into training and test sets
- Training the logistic regression model
- Evaluating it using **accuracy** and **classification report**
- Taking **custom user input** to predict species
- Saving the model using **joblib** so it can be reused anytime

Cool part? We typed in actual sepal and petal measurements, and the model correctly predicted the flower's species! Felt like magic 🌼🔮

🧠 Understanding K-Nearest Neighbors (KNN)

We also got introduced to a new algorithm today: **KNN (K-Nearest Neighbors)**.

It's very intuitive — it classifies new data based on how similar it is to the existing data. Imagine asking your neighbors for advice — the majority opinion wins! 😄

🖊️ Key Points:

- KNN checks the 'k' closest points to make a decision
- It doesn't build a complex model in advance, just compares distances
- Works great when you have a small dataset and well-separated classes

We'll explore its implementation more in upcoming sessions.

FINAL FULL CODE :::
```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
```

```
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report
import joblib
# Load the Iris dataset
df = pd.read_csv('IRIS.csv')
print("First few rows of the dataset:")
print(df.head())
# Features and target
X = df[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].values
y = df['species'].values
# Split data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
# Train logistic regression model
model = LogisticRegression(multi_class='ovr', random_state=42)
model.fit(X_train, y_train)
# Predict on test set
y_pred = model.predict(X_test)
# Evaluate model
accuracy = accuracy_score(y_test, y_pred)
print("\nModel Evaluation:")
print(f"Accuracy: {accuracy:.2f}")
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
# Save model
joblib.dump(model, 'iris_model.pkl')
print("\nModel saved as 'iris_model.pkl'")
# Function to predict species from user input
def predict_iris_species(model, feature_names):
print("\nEnter values (in cm):")
user_input = []
for feature in feature_names:
while True:
try:
value = float(input(f"{feature}: "))
user_input.append(value)
break
except ValueError:
print("Enter a valid number.")
# Convert input to array and predict
user_input = np.array(user_input).reshape(1, -1)
predicted_species = model.predict(user_input)[0]
print(f"\nPredicted Species: {predicted_species}")
# Test with user input
feature_names = ['Sepal Length', 'Sepal Width', 'Petal Length', 'Petal Width']
print("\nTesting with user input:")
predict_iris_species(model, feature_names)
)
```

📊 **Final Output and Reflections**

- The model worked great — very high accuracy!
- Classification report showed excellent precision and recall
- Saved model using joblib so we can load it later anytime
- Took real-time input and predicted flower species — very interactive!

💬 **What I Learned Today:**

- How to use **real CSV data** instead of built-in datasets
- How to **evaluate classification models**
- How to **take dynamic input** from the user
- What **KNN** is and how it works in simple terms