

Nama Dosen : Teguh Iman Hermanto, M.Kom
Mata Kuliah : Machine Learning 2
Pembahasan : Machine Learning Project 1
Pokok Pemb : Membangun aplikasi Machine Learning berbasis WEB (Streamlit)

1. BUSINESS UNDERSTANDING

Project Domain

Domain: Agriculture & Precision Farming

Sub-Domain: Smart Farming & Decision Support Systems

Sistem rekomendasi tanaman (Crop Recommendation System) termasuk dalam bidang pertanian presisi (precision agriculture) yang memanfaatkan data science dan machine learning untuk membantu petani memilih tanaman yang paling sesuai berdasarkan kondisi tanah dan iklim.

Problem Statements

Masalah yang Dihadapi:

1. Ketidaktepatan Pemilihan Tanaman

- Petani sering memilih tanaman berdasarkan pengalaman tradisional tanpa mempertimbangkan data ilmiah tentang kondisi tanah dan iklim.
- Hal ini dapat menyebabkan hasil panen tidak optimal atau bahkan gagal.

2. Ketergantungan pada Ahli Pertanian

- Konsultasi dengan ahli agronomi membutuhkan biaya dan waktu yang tidak selalu terjangkau oleh petani kecil.

3. Perubahan Iklim yang Tidak Terduga

- Perubahan cuaca ekstrem memengaruhi produktivitas tanaman, sehingga diperlukan prediksi berbasis data.

4. Pemanfaatan Lahan yang Tidak Optimal

- Tanah dengan kandungan nutrisi berbeda membutuhkan tanaman yang berbeda, tetapi petani sering menanam satu jenis tanaman secara terus-menerus, menyebabkan degradasi tanah.

Goals

Membangun sistem berbasis AI (Artificial Neural Network) yang dapat merekomendasikan tanaman terbaik berdasarkan parameter:

1. Kondisi Tanah (Nitrogen, Phosphorus, Potassium, pH)
2. Kondisi Iklim (Suhu, Kelembaban, Curah Hujan)

Solution Statements

- Machine Learning Model

Menggunakan Artificial Neural Network (ANN) untuk klasifikasi tanaman berdasarkan dataset parameter tanah dan iklim.

Model dilatih untuk memprediksi tanaman dengan akurasi tinggi (>90%).

- Aplikasi Web (Streamlit)

Membuat antarmuka yang mudah digunakan bagi petani untuk memasukkan parameter tanah dan mendapatkan rekomendasi instan.

- Optimasi dengan TensorFlow Lite

Model dikonversi ke format TFLite agar ringan dan bisa dijalankan di perangkat dengan sumber daya terbatas.

2. DATA UNDERSTANDING

- Import Data Kaggle

```
1 from google.colab import files
2 files.upload()
```

```
1 !mkdir -p ~/.kaggle
2 !cp kaggle.json ~/.kaggle
3 !chmod 600 ~/.kaggle/kaggle.json
4 !ls ~/.kaggle
```

```
1 !kaggle datasets download -d atharvaingle/crop-recommendation-dataset
```

```
1 !mkdir crop-recommendation-dataset
2 !unzip crop-recommendation-dataset.zip -d crop-recommendation-dataset
3 !ls crop-recommendation-dataset
```

- Import Library

```
1 import pandas as pd
2 import numpy as np
3 import seaborn as sns
4 import matplotlib.pyplot as plt
5 import tensorflow as tf
6 from tensorflow import keras
7 from tensorflow.keras import layers
8 from sklearn.model_selection import train_test_split
9 from sklearn.preprocessing import LabelEncoder, MinMaxScaler
10 from sklearn.metrics import confusion_matrix, classification_report
```

- **Exploratory Data Analysis**



```
1 df = pd.read_csv('/content/crop-recommendation-dataset.zip')
```

```
df.head()
```

Python

	N	P	K	temperature	humidity	ph	rainfall	label
0	90	42	43	20.879744	82.002744	6.502985	202.935536	rice
1	85	58	41	21.770462	80.319644	7.038096	226.655537	rice
2	60	55	44	23.004459	82.320763	7.840207	263.964248	rice
3	74	35	40	26.491096	80.158363	6.980401	242.864034	rice
4	78	42	42	20.130175	81.604873	7.628473	262.717340	rice

3. DATA PREPARATION


```
1 # pisahkan fitur dan target
2 X = df.drop(columns=["label"])
3 y = df["label"]
```



```
1 # Encode label target
2 le = LabelEncoder()
3 y_encoded = le.fit_transform(y)
```



```
1 # Normalisasi fitur
2 scaler = MinMaxScaler()
3 X_scaled = scaler.fit_transform(X)
```

[illegible]

4. MODELING


```
1 model = keras.Sequential([
2     keras.Input(shape=(X_train.shape[1],)),
3     layers.Dense(128, activation="relu"),
4     layers.Dense(64, activation="relu"),
5     layers.Dense(32, activation="relu"),
6     layers.Dense(len(le.classes_), activation="softmax")
7 ])
```



```
1 model.compile(optimizer="adam",
2               loss="sparse_categorical_crossentropy",
3               metrics=["accuracy"])
```



```
1 model.summary()
```



```
1 from tensorflow.keras.utils import plot_model
2 plot_model(model, show_shapes = True)
```



```
1 history = model.fit(X_train, y_train,
2                     epochs=50,
3                     batch_size=16,
4                     validation_data=(X_test, y_test))
```

5. EVALUATION



```
1 test_loss, test_accuracy = model.evaluate(X_test, y_test, verbose=1)
2 print(f"Akurasi Model: {test_accuracy:.4f}")
3 print(f"Loss Model: {test_loss:.4f}")
```



```
1 plt.figure(figsize=(12, 5))
```



```
1 plt.subplot(2, 2, 4)
2 plt.plot(history.history["accuracy"], label='Training Accuracy')
3 plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
4 plt.xlabel('Epochs')
5 plt.ylabel('Accuracy')
6 plt.legend()
7 plt.title('Accuracy')
8
9 plt.show()
```



```
1 plt.subplot(1, 2, 2)
2 plt.plot(history.history["loss"], label='Training Loss')
3 plt.plot(history.history['val_loss'], label='Validation Loss')
4 plt.xlabel('Epochs')
5 plt.ylabel('Loss')
6 plt.legend()
```



```
1 y_pred = model.predict(X_test)
2 y_pred_classes = np.argmax(y_pred, axis=1)
3
4 cm = confusion_matrix(y_test, y_pred_classes)
5 plt.figure(figsize=(10, 7))
6 sns.heatmap(cm, annot=True, fmt="d", cmap="Blues")
7 plt.xlabel("Predicted Label")
8 plt.ylabel("True Label")
9 plt.title("Confusion Matrix")
10 plt.show()
11
12 print("Classification Report:")
13 print(classification_report(y_test, y_pred_classes))
```

6. DEPLOYMENT

- Model Simulation



```
1 sample_input = np.array([[78, 42, 42, 20.13, 81.60, 7.62, 262.71]])
2 sample_input_df = pd.DataFrame(sample_input)
```



```
1 sample_input_scaled = scaler.transform(sample_input_df)
```



```
1 predicted_class = np.argmax(model.predict(sample_input_scaled))
2 predicted_crop = le.inverse_transform([predicted_class])
3
4 print(f"Tanaman yang direkomendasikan: {predicted_crop[0]}")
```

- Save Model




```
1 # Konversi model ke TFLite
2 converter = tf.lite.TFLiteConverter.from_keras_model(model)
3 tflite_model = converter.convert()
4
5 # Simpan model
6 with open('crop_recommendation.tflite', 'wb') as f:
7     f.write(tflite_model)
```



```
1 # Simpan label encoder dan scaler
2 import joblib
3 joblib.dump(le, 'label_encoder.pkl')
4 joblib.dump(scaler, 'scaler.pkl')
```

APLIKASI STREAMLIT

1. IMPORT LIBRARY




```
1 import streamlit as st
2 import tensorflow as tf
3 import numpy as np
4 import joblib
```

2. LOAD MODEL




```
1 # Load scaler dan label encoder
2 scaler = joblib.load('scaler.pkl')
3 label_encoder = joblib.load('label_encoder.pkl')
4
5 # Load model TFLite
6 interpreter = tf.lite.Interpreter(model_path="crop_recommendation.tflite")
7 interpreter.allocate_tensors()
8
9 input_details = interpreter.get_input_details()
10 output_details = interpreter.get_output_details()
```

3. MEMBUAT INTERFACE



```
1 # Judul Aplikasi
2 st.title("Crop Recommendation System")
3 st.write("Masukkan informasi lingkungan untuk mendapatkan rekomendasi tanaman yang cocok.")
4
5 # Form input pengguna
6 N = st.number_input("Kandungan Nitrogen (N)", min_value=0.0, max_value=200.0, value=50.0)
7 P = st.number_input("Kandungan Phosphorus (P)", min_value=0.0, max_value=200.0, value=50.0)
8 K = st.number_input("Kandungan Potassium (K)", min_value=0.0, max_value=200.0, value=50.0)
9 temperature = st.number_input("Suhu (°C)", min_value=0.0, max_value=50.0, value=25.0)
10 humidity = st.number_input("Kelembapan (%)", min_value=0.0, max_value=100.0, value=60.0)
11 ph = st.number_input("pH Tanah", min_value=0.0, max_value=14.0, value=6.5)
12 rainfall = st.number_input("Curah Hujan (mm)", min_value=0.0, max_value=300.0, value=100.0)
```

4. EKSEKUSI MODEL



```
1 if st.button("Rekomendasikan Tanaman"):
2     # Preprocessing input
3     input_data = np.array([[N, P, K, temperature, humidity, ph, rainfall]])
4     input_scaled = scaler.transform(input_data).astype(np.float32)
5
6     interpreter.set_tensor(input_details[0]['index'], input_scaled)
7     interpreter.invoke()
8     prediction = interpreter.get_tensor(output_details[0]['index'])
9
10    predicted_label = np.argmax(prediction)
11    crop_name = label_encoder.inverse_transform([predicted_label])[0]
12
13    st.success(f"Rekomendasi tanaman: **{crop_name.upper()}**")
```