Tugas01-10 Praktikum Mandiri

SYAHRI GHIFARI MAULIDI 0110222217

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

1.1 Persiapan dan Setup Environment

```
import pandas as pd
import numpy as np
import pickle
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score
import matplotlib.pyplot as plt

from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
```

Proses dimulai dengan mengimpor semua library yang diperlukan termasuk pandas untuk manipulasi data, numpy untuk operasi matematika, scikit-learn untuk algoritma machine learning dan evaluasi model, serta matplotlib untuk visualisasi hasil. Library pickle juga disiapkan untuk penyimpanan model yang telah dilatih, sementara koneksi ke Google Drive dilakukan untuk mengakses dataset yang tersimpan secara cloud-based. Setup environment ini memastikan semua tools yang diperlukan tersedia untuk seluruh pipeline analisis data, dari loading dataset hingga evaluasi model akhir.

1.2 Load dan Eksplorasi Data

```
path = "/content/drive/MyDrive/Praktikum/Praktikum10/Data/weather_classification_data.csv"
```

```
df = pd.read_csv(path)
print("Loaded dataset shape:", df.shape)
print("Columns:", list(df.columns))
print(df.head())
```

```
Show hidden output
```

Dataset dimuat dari file CSV yang berisi 13,200 sampel dengan 11 fitur mencakup parameter meteorologi seperti temperature, humidity, wind speed, precipitation percentage, cloud cover,

atmospheric pressure, UV index, season, visibility, location, dan target variable weather type. Eksplorasi awal menunjukkan distribusi data yang seimbang dengan variabel numerik dan kategorikal, memberikan gambaran komprehensif tentang karakteristik dataset sebelum dilakukan preprocessing lebih lanjut. Data ini merepresentasikan berbagai kondisi cuaca yang akan diklasifikasikan menjadi beberapa kategori utama.

1.3 Preprocessing Data

```
label_encoders = {}
df_proc = df.copy()
```

```
cat_cols = df_proc.select_dtypes(include=['object', 'category']).columns.tolist()
# Ensure target included
if 'Weather Type' not in cat_cols and df_proc['Weather Type'].dtype != 'int64':
    cat_cols.append('Weather Type')

print("Categorical columns detected:", cat_cols)

for col in cat_cols:
    le = LabelEncoder()
    df_proc[col] = le.fit_transform(df_proc[col].astype(str))
    label_encoders[col] = le
```

```
Categorical columns detected: ['Cloud Cover', 'Season', 'Location', 'Weather Type']
```

```
X = df_proc.drop('Weather Type', axis=1)
y = df_proc['Weather Type']
```

```
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.20, random_state=42, stratify=y)
print("Train / Test sizes:", X_train.shape, X_test.shape)
```

```
Train / Test sizes: (10560, 10) (2640, 10)
```

```
numeric_cols = X.select_dtypes(include=[np.number]).columns.tolist()
scaler = StandardScaler()
X_train_scaled = X_train.copy()
X_test_scaled = X_test.copy()
X_train_scaled[numeric_cols] = scaler.fit_transform(X_train[numeric_cols])
X_test_scaled[numeric_cols] = scaler.transform(X_test[numeric_cols])
```

Tahap preprocessing meliputi transformasi variabel kategorikal menjadi numerik menggunakan LabelEncoder untuk fitur-fitur seperti cloud cover, season, location, dan weather

type, kemudian data dipisah menjadi feature set (X) dan target variable (y). Dataset dibagi menjadi training set (80%) dan test set (20%) dengan stratifikasi untuk mempertahankan distribusi kelas, sementara feature scaling dilakukan menggunakan StandardScaler untuk menormalisasi variabel numerik agar compatible dengan algoritma KNN yang sensitive terhadap perbedaan skala. Preprocessing yang komprehensif ini memastikan data siap untuk proses training model dengan kualitas optimal.

1.4 Pemodelan KNN

```
param_grid = {'n_neighbors': [3,5,7,9,11]}
knn = KNeighborsClassifier()
grid = GridSearchCV(knn, param_grid, cv=5, scoring='accuracy', n_jobs=-1)
grid.fit(X_train_scaled, y_train)
best_k = grid.best_params_['n_neighbors']
print("GridSearch best k:", best_k)
print("Best CV score:", grid.best_score_)
```

```
GridSearch best k: 7
Best CV score: 0.9005681818181819
```

```
final_knn = KNeighborsClassifier(n_neighbors=best_k)
final_knn.fit(X_train_scaled, y_train)
```

Proses pemodelan menggunakan algoritma K-Nearest Neighbors dengan hyperparameter tuning melalui GridSearchCV untuk menemukan nilai k optimal dari range [3,5,7,9,11] dengan cross-validation 5-fold. Hasil tuning menunjukkan k=7 memberikan performance terbaik dengan accuracy 90.06% pada validasi, kemudian model final dilatih menggunakan parameter optimal tersebut pada data training yang telah di-scaling. Pemilihan k=7 ini menyeimbangkan antara bias dan variance model, menghindari overfitting sambil tetap menangkap pola kompleks dalam data cuaca.

1.5 Evaluasi Model

```
y_pred = final_knn.predict(X_test_scaled)
acc = accuracy_score(y_test, y_pred)
print(f"\nTest Accuracy: {acc:.4f}\n")
print("Classification Report:\n")
print(classification_report(y_test, y_pred, digits=4))
cm = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:\n", cm)
```

Show hidden output

```
target_le = label_encoders['Weather Type']
class_names = target_le.inverse_transform(np.arange(len(target_le.classes_)))
```

```
fig, ax = plt.subplots(figsize=(7,6))
im = ax.imshow(cm, interpolation='nearest')
ax.set_title('Confusion Matrix (test set)')
ax.set_xlabel('Predicted label')
ax.set_ylabel('True label')
ax.set_xticks(np.arange(len(class_names)))
ax.set_yticks(np.arange(len(class_names)))
ax.set_xticklabels(class_names, rotation=45, ha='right')
ax.set_yticklabels(class_names)
thresh = cm.max() / 2.
for i in range(cm.shape[0]):
    for j in range(cm.shape[1]):
        ax.text(j, i, format(cm[i, j], 'd'),
                ha="center", va="center",
                color="white" if cm[i, j] > thresh else "black")
fig.tight_layout()
plt.show()
```

Evaluasi model dilakukan pada test set dengan hasil accuracy 89.55%, menunjukkan konsistensi yang baik dengan cross-validation score. Classification report mengungkapkan performance detail per kelas dengan precision antara 84.57%-93.90%, recall 88.33%-89.70%, dan f1-score 87.06%-92.38%, sementara confusion matrix visualisasi mengidentifikasi pola kesalahan klasifikasi antar kelas yang serupa. Model menunjukkan kemampuan terbaik dalam memprediksi kelas 2 dengan precision tertinggi, sedangkan kelas 0 memiliki precision terendah namun masih dalam range acceptable untuk aplikasi praktis.