Nama : Satya Athaya Daniswara

NIM : 1103213152

TUGAS PERBAIKAN BAB 6

1. Pengantar Klasifikasi

- Klasifikasi adalah salah satu tugas utama dalam machine learning, di mana tujuan utamanya adalah untuk memprediksi kategori atau label dari data baru berdasarkan data pelatihan.

- Penulis menjelaskan perbedaan antara klasifikasi biner (dua kelas) dan klasifikasi multikelas (lebih dari dua kelas).

2. Algoritma Klasifikasi

- K-Nearest Neighbors (KNN):

- Algoritma yang sederhana dan intuitif yang mengklasifikasikan data baru berdasarkan kedekatannya dengan data pelatihan.

- Penulis menjelaskan cara menghitung jarak (misalnya, jarak Euclidean) dan memilih nilai K yang optimal.

- Logistic Regression:

- Meskipun namanya mengandung "regresi", ini adalah algoritma klasifikasi yang digunakan untuk memprediksi probabilitas kelas.

- Penulis menjelaskan fungsi sigmoid dan bagaimana model ini dapat digunakan untuk klasifikasi biner.

- Support Vector Machines (SVM):

- Algoritma yang mencari hyperplane optimal untuk memisahkan kelas-kelas dalam data.

- Penulis membahas konsep margin dan kernel trick untuk menangani data yang tidak terpisah secara linier.

- Decision Trees:

- Model yang membagi data menjadi subset berdasarkan fitur-fitur tertentu, membentuk struktur pohon.

- Penulis menjelaskan cara kerja algoritma ini, termasuk penggunaan kriteria seperti Gini impurity dan entropy untuk menentukan pembagian.

- Random Forest:

- Ensemble method yang menggabungkan beberapa decision trees untuk meningkatkan akurasi dan mengurangi overfitting.

- Penulis menjelaskan cara kerja Random Forest dan keuntungannya dibandingkan dengan decision tree tunggal.

- Gradient Boosting:

- Metode ensemble yang membangun model secara bertahap, di mana setiap model baru berusaha memperbaiki kesalahan model sebelumnya.

- Penulis membahas algoritma seperti XGBoost yang populer dalam kompetisi data science.

3. Implementasi Klasifikasi dengan Scikit-learn

- Penulis memberikan contoh kode untuk menerapkan berbagai algoritma klasifikasi menggunakan pustaka Scikit-learn.

- Contoh mencakup:

- Memuat dataset (misalnya, dataset Iris atau dataset lain yang relevan).

- Memisahkan data menjadi set pelatihan dan pengujian.

- Melatih model klasifikasi.

- Membuat prediksi dan mengevaluasi kinerja model menggunakan metrik yang telah dibahas sebelumnya (akurasi, precision, recall, F1 score).

4. Evaluasi Model Klasifikasi

- Penulis menekankan pentingnya evaluasi model klasifikasi dengan menggunakan metrik yang sesuai.

- Diskusi tentang penggunaan confusion matrix untuk memahami kinerja model secara lebih mendetail, termasuk true positives, false positives, true negatives, dan false negatives.

5. Tuning Model

- Penulis menjelaskan pentingnya tuning hyperparameter untuk meningkatkan kinerja model klasifikasi.

- Contoh penggunaan Grid Search dan Randomized Search untuk menemukan kombinasi hyperparameter terbaik.

6. Studi Kasus

- Di akhir bab, penulis sering menyertakan studi kasus atau contoh praktis yang menunjukkan penerapan algoritma klasifikasi pada dataset nyata.

- Ini memberikan konteks dan pemahaman yang lebih baik tentang bagaimana algoritma bekerja dalam situasi dunia nyata.

7. Kesimpulan

- Bab ini memberikan pemahaman yang komprehensif tentang berbagai algoritma klasifikasi, cara menerapkannya, dan pentingnya evaluasi serta tuning model.

- Pembaca diharapkan dapat memilih algoritma yang tepat berdasarkan karakteristik data dan tujuan analisis.

Contoh Kode Klasifikasi

Berikut adalah contoh kode sederhana yang menunjukkan penerapan K-Nearest Neighbors (KNN) untuk klasifikasi menggunakan dataset Iris:

```python
Importing necessary libraries
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

Load the Iris dataset
iris = load_iris()
X = iris.data
y = iris.target

Split the dataset 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)

Create KNN classifier
knn = KNeighborsClassifier(n_neighbors=3)
```

```
 Fit the classifier to the training data
knn.fit(X_train, y_train)

 Make predictions on the test data
y_pred = knn.predict(X_test)

 Evaluate the model
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred))
print("Classification Report:\n", classification_report(y_test, y_pred))
```

Jika Anda memiliki pertanyaan lebih lanjut atau ingin mendalami bagian tertentu dari bab ini, silakan beri tahu!