Nama: Dhilan Djalani Kusuma Putra

NIM: 231011401056

Kelas: 04TPLP025

Mata kuliah: Data Mining

Summary Data Mining

1. Pandas: Fondasi Manipulasi dan Analisis Data di Python

-Bayangkan Anda memiliki data dalam bentuk tabel, seperti spreadsheet Excel atau database. Pandas adalah toolkit super canggih di Python yang dirancang khusus untuk bekerja dengan data seperti itu secara efisien dan fleksibel. Intinya, Pandas memperkenalkan dua struktur data utama:

-Series: Ini seperti array satu dimensi (atau kolom tunggal dalam spreadsheet) yang dapat menyimpan berbagai jenis data (angka, teks, tanggal, dll.). Yang membuatnya istimewa adalah ia memiliki index, yaitu label untuk setiap

elemen. Index ini bisa berupa angka, tanggal, atau bahkan teks, memungkinkan Anda mengakses data dengan lebih intuitif daripada hanya menggunakan posisi angka.

-DataFrame: Ini adalah struktur data inti Pandas. Anda bisa membayangkannya sebagai tabel dua dimensi (seperti spreadsheet) yang terdiri dari beberapa Series yang berbagi index yang sama. Setiap kolom dalam DataFrame adalah sebuah Series. DataFrame adalah tempat sebagian besar pekerjaan analisis data Anda akan terjadi.

Pandas:Library Python yang menyediakan struktur data (DataFrame, Series) yang kuat dan fleksibel untuk manipulasi dan analisis data tabular. Memudahkan pembacaan data dari berbagai format (CSV, Excel, SQL), pembersihan data yang hilang, transformasi data, dan analisis statistik deskriptif.

2. Matplotlib: Visualisasi Data yang Kuat di Python

Setelah Anda membersihkan dan memproses data Anda dengan Pandas, langkah selanjutnya yang seringkali penting adalah memvisualisasikannya. Di sinilah Matplotlib berperan. Matplotlib adalah library fundamental untuk membuat berbagai jenis grafik dan plot di Python.

Apa yang Bisa Dilakukan Matplotlib?

Berbagai Jenis Plot: Matplotlib memungkinkan Anda membuat berbagai visualisasi, termasuk:

Line Plot: Untuk menunjukkan tren data dari waktu ke waktu atau hubungan antara dua variabel kontinu.

Scatter Plot: Untuk melihat hubungan dan distribusi antara dua variabel.

Bar Chart: Untuk membandingkan nilai antar kategori.

Histogram: Untuk menunjukkan distribusi frekuensi dari satu variabel.

Pie Chart: Untuk menunjukkan proporsi kategori dalam keseluruhan.

Box Plot: Untuk merangkum distribusi data dan mengidentifikasi outlier.

Dan banyak lagi (area plot, contour plot, dll.).

Matplotlib: Library Python untuk visualisasi data statis, interaktif, dan animasi. Memungkinkan pembuatan berbagai jenis plot (histogram, scatter plot, bar chart, line plot) untuk memahami pola dan tren dalam data

3. Naive Bayes: Klasifikasi Probabilistik yang Sederhana Namun Efektif

Setelah Anda memiliki data yang bersih dan mungkin telah Anda visualisasikan, Anda mungkin ingin menggunakan machine learning untuk membuat prediksi atau mengklasifikasikan data baru. Naive Bayes adalah

salah satu algoritma klasifikasi yang sederhana namun seringkali efektif, terutama dalam kasus di mana jumlah fitur sangat banyak (misalnya, dalam pemrosesan teks).

Bagaimana Cara Kerja Naive Bayes?

Naive Bayes didasarkan pada Teorema Bayes, sebuah konsep fundamental dalam probabilitas:

$P(A|B) =$

$P(B)$

$P(B|A) \cdot P(A)$

Di mana:

$P(A|B)$ adalah probabilitas kejadian A terjadi mengingat kejadian B telah terjadi (disebut posterior probability).

$P(B|A)$ adalah probabilitas kejadian B terjadi mengingat kejadian A telah terjadi (disebut likelihood).

$P(A)$ adalah probabilitas kejadian A terjadi tanpa kondisi (disebut prior probability).

$P(B)$ adalah probabilitas kejadian B terjadi tanpa kondisi (disebut evidence atau normalizing constant).

Dalam konteks klasifikasi, kita ingin memprediksi kelas (label) y untuk sebuah instance data x (yang terdiri dari beberapa fitur). Jadi, kita ingin menghitung $P(y|x)$.

Algoritma Naive Bayes:

-Konsep Dasar: Algoritma klasifikasi probabilistik yang didasarkan pada teorema Bayes dengan asumsi "naive" (independensi antar fitur).

-Jenis-jenis Naive Bayes: Gaussian Naive Bayes (untuk fitur kontinu berdistribusi normal), Multinomial Naive Bayes (untuk fitur diskrit seperti frekuensi kata), dan Bernoulli Naive Bayes (untuk fitur biner).

-Penerapan: Digunakan dalam berbagai tugas klasifikasi seperti klasifikasi teks (spam filtering, sentiment analysis), diagnosis medis sederhana, dan sistem rekomendasi.

-Implementasi di Scikit-learn: Menggunakan kelas seperti GaussianNB, MultinomialNB, dan BernoulliNB dari sklearn.naive_bayes. Melibatkan tahap inisialisasi model, pelatihan model menggunakan data latih (.fit()), dan prediksi pada data uji (.predict()).

4. Decision Tree: Klasifikasi dan Regresi Berbasis Aturan

Decision Tree (Pohon Keputusan) adalah algoritma machine learning yang bekerja dengan membuat serangkaian keputusan (seperti pertanyaan ya/tidak) untuk mengklasifikasikan atau memprediksi nilai dari suatu data. Struktur algoritma ini menyerupai pohon dengan node-node yang merepresentasikan keputusan atau pengujian pada fitur, cabang-cabang yang merepresentasikan hasil dari pengujian tersebut, dan daun-daun yang merepresentasikan label kelas (untuk klasifikasi) atau nilai prediksi (untuk regresi).

Bagaimana Cara Kerja Decision Tree?

Proses pembangunan Decision Tree melibatkan pemilihan fitur yang paling informatif untuk membagi data pada setiap node. Tujuannya adalah untuk menciptakan pemisahan yang menghasilkan subset data yang lebih "murni" dalam hal kelas target (untuk klasifikasi) atau variasi nilai target (untuk regresi).

Algoritma Decision Tree:

Konsep Dasar: Algoritma klasifikasi dan regresi non-parametrik yang membangun struktur pohon seperti alur flowchart, di mana setiap node internal merepresentasikan pengujian pada suatu fitur, setiap cabang merepresentasikan hasil pengujian, dan setiap node daun merepresentasikan label kelas atau nilai prediksi.

Pembentukan Pohon: Pohon dibangun secara rekursif dengan memilih fitur yang paling informatif untuk membagi data pada setiap node (berdasarkan impurity measures seperti Gini impurity atau entropy untuk klasifikasi, dan variance reduction untuk regresi).

Kelebihan: Mudah diinterpretasikan dan divisualisasikan, dapat menangani data numerik dan kategorikal, dan tidak memerlukan asumsi tentang distribusi data.

Kekurangan: Cenderung overfitting (terlalu cocok dengan data latih), tidak stabil (perubahan kecil pada data latih dapat menghasilkan pohon yang berbeda).

Implementasi di Scikit-learn: Menggunakan kelas DecisionTreeClassifier untuk klasifikasi dan DecisionTreeRegressor untuk regresi dari sklearn.tree. Melibatkan tahap inisialisasi model, pelatihan model (.fit()), prediksi (.predict()), dan visualisasi pohon menggunakan export_graphviz dari sklearn.tree.