LAPORAN TUGAS BESAR 2
IF3170 INTELEGENSI BUATAN
IMPLEMENTASI ALGORITMA

Kelompok 32 (Wa bingung)

Daniel Egiant Sitanggang	13521056
Go Dillon Audris	13521062
Margaretha Olivia Haryono	13521071
Austin Gabriel Pardosi	13521084

PROGRAM STUDI SARJANA TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG
2023

A. Algoritma KNN (K-*Nearest Neighbors*)

1. Penjelasan Singkat Algoritma

K-Nearest Neighbors (KNN) adalah salah satu algoritma dalam *supervised learning* yang digunakan untuk masalah klasifikasi dan regresi. Algoritma KNN merupakan *Instance-Based Classifier* sehingga akan menyimpan seluruh data *train* dan tidak memiliki hipotesis atau model yang dibentuk pada saat *training*. Sifat algoritma KNN yang seperti ini membuatnya disebut sebagai *lazy learner.* Algoritma ini memanfaatkan ketergantungan pada tetangga terdekat suatu data *point* dalam ruang fitur untuk membuat prediksi.

Berikut adalah langkah-langkah algoritma KNN dalam melakukan klasifikasi:

1. Menghitung jarak antara data yang akan diprediksi dengan semua data dalam *set*.
2. Mengidentifikasi K data terdekat (tetangga) berdasarkan jarak.
3. Menggunakan mayoritas dari label tetangga sebagai hasil prediksi untuk data.

KNN memiliki beberapa kelebihan sebagai berikut:

1. Sederhana dan mudah diimplementasikan.
2. Tidak memerlukan pembentukan model.
3. Efektif untuk *dataset* yang tidak linier.
4. Tidak sensitif terhadap *outliers.*

Adapun kelemahan KNN yaitu memiliki komputasi yang tinggi sehingga kinerja algoritma relatif lebih lambat dibandingkan dengan algoritma pembelajaran lainnya.

2. Implementasi Algoritma

Dalam memecahkan persoalan melakukan klasifikasi price_range pada *dataset test* yang disediakan berdasarkan *dataset train* dengan algoritma pembelajaran KNN, terdapat tiga tahapan yaitu *preprocessing* data sebelum dilatih ke model, *fitting* atau melatih data menjadi model KNN, dan *predicting* atau melakukan klasifikasi pada data baru berdasarkan model yang telah di *train*. Berikut tahapan-tahapan implementasi algoritma KNN yang kami implementasikan

2.1. *Preprocessing* data untuk algoritma KNN

a. Melakukan normalisasi data

Hal ini dilakukan karena algoritma KNN yang diimplementasikan memanfaatkan *euclidean distance* yang sensitif terhadap skala antar-fitur *dataset*. Normalisasi dilakukan dengan pendekatan *min-max scaling* dengan rumus berikut.

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

b. Melakukan *feature selection*

Hal ini dilakukan untuk mencegah *overfitting* pada model serta meningkatkan kinerja dari model KNN. *Feature selection* dilakukan dengan pendekatan *wrapper* sehingga diperoleh fitur-fitur yang di-*drop* yaitu blue, clock_speed, dual_sim, fc, four_g, int_memory, m_dep, mobile_wt, n_cores, pc, sc_h, sc_w, talk_time, three_g, touch_screen, dan wifi.

2.2. *Fitting* atau *learning*

Pada tahap ini sebenarnya model tidak benar-benar di train untuk menghasilkan hipotesis, melainkan melakukan *restructuring* pada data *train* menjadi *point-point* yang dipartisi berdasarkan klasifikasi pada target yaitu price_range. Hal ini dilakukan untuk memudahkan proses prediksi ke depannya.

2.3. *Predicting*

Pada tahap ini, model akan melakukan klasifikasi berdasarkan data atau *point-point* baru yang dimasukkan berdasarkan *point-point* pada *data train* sebelumnya. Klasifikasi dari suatu point ditentukan berdasarkan kelas terbanyak dari K tetangga terdekat *(K-Nearest-Neighbor)* yang dihitung berdasarkan *euclidean distance.* Pada implementasi ini, dipilih nilai K sebesar 33 berdasarkan *trial and error.*

3. Analisis Perbandingan Hasil Algoritma dengan Hasil Pustaka

Pustaka yang digunakan sebagai pembanding dari model KNN hasil implementasi kami yaitu sklearn.neighbors. Metrik yang kami gunakan dalam menentukan kemiripan model hasil implementasi kami dengan model yang tersedia pada pustaka yaitu *accuracy* karena *metric* tersebut kami rasa paling mendeskripsikan suatu model pada *multiclass classification.* Dilakukan *preprocessing* data yang hendak dimasukkan ke model pustaka

sesuai dengan *preprocessing* yang dilakukan ke model hasil implementasi. Kami juga melakukan *tuning* terhadap model KNN pustaka sklearn agar semirip mungkin dengan model kami yaitu dengan menggunakan nilai K sebesar 33, menghitung jarak dengan *euclidean distance*, serta KNN yang dihitung dengan *bruteforce*. Berikut adalah kode program *python* untuk menentukan akurasi model dari pustaka *sklearn.*

comparison.py

```
from sklearn.neighbors import KNeighborsClassifier

def preprocess(X):
      X = X.drop(["blue", "clock_speed", "dual_sim", "fc", "four_g",
                  "int_memory", "m_dep", "mobile_wt", "n_cores", "pc",
                  "sc_h", "sc_w", "talk_time", "three_g",
                  "touch_screen", "wifi"], axis=1)

      column_names = X.columns.tolist()
      for column in column_names:
          max = X[column].max()
          min = X[column].min()
          X[column] = (X[column] - min) / (max - min)

      return X

knn_model = KNeighborsClassifier(n_neighbors=33, algorithm='brute',
            p=2)
knn_model.fit(preprocess(X_train), y_train)
knn_predictions = knn_model.predict(preprocess(X_validation))
knn_accuracy = accuracy_score(y_validation, knn_predictions)
print(f"Akurasi KNN: {knn_accuracy}")
```

Dari model tersebut, diperoleh nilai akurasi sebesar 90.5%. Nilai ini sedikit berbeda dengan nilai model hasil implementasi yaitu sebesar 90.34%. Hal ini mungkin terjadi karena adanya perbedaan struktur dan tipe data yang digunakan pada algoritma implementasi kami dengan algoritma pustaka (misal kami menggunakan *built-in array python*, *sklearn* menggunakan np.array). Perbedaan struktur dan tipe data tersebut berpengaruh pada akurasi hasil komputasi matematis sehingga dapat menghasilkan hasil akhir yang sedikit berbeda, yang sering disebut sebagai *floating-point error.* Namun, hasil akurasi tersebut sudah cukup mendekati sehingga algoritma KNN yang kami implementasikan sudah cukup baik.

B. Algoritma Naive Bayes

1. Penjelasan Singkat Algoritma

Naive Bayes adalah sebuah algoritma klasifikasi berdasarkan pada Teorema Bayes dengan asumsi independensi yang kuat antara fitur-fitur dalam *dataset*. Algoritma ini dikenal sebagai *naive* (naif) karena menganggap setiap pasangan fitur dalam data sebagai independen satu sama lain, meskipun sebenarnya fitur-fitur tersebut mungkin saling terkait atau bergantung.

Berikut adalah langkah-langkah algoritma Naive Bayes dalam melakukan klasifikasi:

1. Menghitung frekuensi setiap nilai atribut untuk kelas tertentu, dan frekuensi setiap kelas.
2. Menghitung P(Ai | Vi) yaitu probabilitas kondisional dari suatu nilai fitur (Ai) terhadap suatu kelas tertentu (Vi). Dengan kata lain, probabilitas kemungkinan munculnya suatu nilai fitur jika diketahui (*given)* kelas tertentu.
3. Menghitung probabilitas kemunculan suatu kelas P(Vi).
4. Membangun hipotesis atau model berdasarkan nilai-nilai probabilitas tersebut.

Naive Bayes memiliki beberapa kelebihan sebagai berikut:

1. Komputasi yang cepat.
2. Kinerja baik pada *dataset* yang besar.

Adapun kelemahan Naive Bayes yaitu pada asumsi algoritma yang beranggapan bahwa seluruh fitur saling independen meski pada kenyataannya seringkali fitur-fitur yang disertakan saling bergantung satu dengan yang lain.

2. Implementasi Algoritma

Pada program ini, terdapat sebuah kelas NaiveBayes yang digunakan untuk mengimplementasikan algoritma NaiveBayes dan menyimpan informasi yang diperlukan. Terdapat beberapa atribut dalam kelas ini, yaitu sebagai berikut:

1. columns_name:

   List yang berisi nama-nama fitur.
2. nominal_columns:

   List yang berisi nama-nama fitur yang bersifat nominal.

3. data_count:

   Jumlah total data yang digunakan untuk pelatihan.
4. label_count:

   Dictionary yang berisi pasangan nilai antara label dan jumlah kemunculan label pada data pelatihan.
5. feature_model:

   Dictionary bertingkat yang berisi informasi untuk setiap label dan fitur. Untuk fitur numerik, berisi mean dan std, sedangkan untuk fitur non-numerik, berisi kemunculan setiap nilai fitur.
6. trained:

   Menyatakan apakah model Naive Bayes sudah dilatih atau belum.

Pada kelas ini, terdapat metode learn_from_data() untuk melatih model Naive Bayes dari data pelatihan. Pertama, metode reset dipanggil untuk mengatur ulang atribut data_count, label_count, dan feature_model. Hal ini memastikan bahwa setiap kali metode ini dipanggil, kita memiliki state yang bersih untuk proses pelatihan baru. Selanjutnya, nilai trained diubah menjadi True yang menandakan model telah dilatih. Kemudian, jumlah total data disimpan dalam atribut data_count. Selanjutnya, dilakukan iterasi terhadap setiap label pada data latih. Untuk setiap label pada data latih, dilakukan perhitungan jumlah kemunculan label dan hasilnya disimpan dalam label_count.

Kemudian, dilakukan iterasi pada setiap fitur. Fitur dibedakan menjadi numerik dan nominal dengan perhitungan sebagai berikut:

1. Jika fitur adalah fitur numerik, dilakukan perhitungan mean dan std dari data latih dan hasilnya disimpan dalam feature_model untuk setiap label.
2. Jika fitur bukan fitur numerik (fitur nominal), dilakukan iterasi pada setiap label dan dilakukan perhitungan kemunculan setiap nilai fitur. Hasilnya disimpan dalam feature_model untuk setiap label.

Setelah selesai melakukan iterasi pada setiap label dan setiap fitur, informasi pelatihan berhasil disimpan dalam atribut feature_model. Variabel feature_model menjadi *nested dictionary* yang mencakup informasi untuk setiap label dan setiap fitur.

Untuk melakukan klasifikasi, terdapat metode calculate_label_probability() dan classify() yang bertanggung jawab untuk menghitung probabilitas dan melakukan

klasifikasi berdasarkan model Naive Bayes yang telah dilatih. Kedua metode ini mengambil data uji sebagai input dan menggunakan model Naive Bayes yang telah dilatih untuk menghitung probabilitas dan melakukan klasifikasi. Perhitungan probabilitas dilakukan dengan mengalikan probabilitas awal (yaitu probabilitas munculnya suatu label) dengan probabilitas dari setiap fitur untuk label tersebut. Jika fitur numerik, dilakukan perhitungan probabilitas menggunakan fungsi distribusi normal (*Gaussian Probability*) dengan mean dan std yang telah dihitung pada tahap pelatihan. Jika fitur non-numerik, dilakukan perhitungan probabilitas menggunakan frekuensi kemunculan setiap nilai pada fitur. Nilai probabilitas akhir dari suatu label akan didapat dengan mengalikan semua probabilitas. Kemudian, metode classify() akan membandingkan nilai probabilitas akhir dari setiap label dan memilih label dengan probabilitas tertinggi sebagai hasil klasifikasi.

3. Analisis Perbandingan Hasil Algoritma dengan Hasil Pustaka

comparison.py

```
from sklearn.naive_bayes import GaussianNB

nb_model = GaussianNB()
nb_model.fit(X_train, y_train)
nb_predictions = nb_model.predict(X_validation)
nb_accuracy = accuracy_score(y_validation, nb_predictions)
print(f"Akurasi Naive Bayes: {nb_accuracy}")
```

Dari model tersebut, diperoleh nilai akurasi sebesar 78.5%. Nilai ini sama dengan nilai model hasil implementasi yaitu sebesar 78.5%. Hal ini menunjukkan bahwa implementasi algoritma Naive Bayes yang kami buat telah memiliki nilai akurasi yang sama dengan menggunakan pustaka *scikit-learn*. Hal ini dapat terjadi karena kemungkinan implementasi algoritma yang kami lakukan menggunakan metode pembelajaran (*learning*) yang sama dengan yang digunakan oleh pustaka *scikit-learn*. Parameter yang digunakan dalam pembelajaran pun bisa jadi sama dengan yang digunakan oleh pustaka. Dengan demikian, nilai akurasi yang didapatkan mendekati atau sama dengan penggunaan pustaka *scikit-learn*.

C. Submisi Kaggle

Untuk melakukan submisi pada Kaggle, maka terdapat beberapa pemrosesan yang harus dilakukan. Pemrosesan dilakukan khususnya terhadap *file* test.csv yang telah disediakan pada Kaggle. Setiap sampel data pada *file* akan dicoba untuk diprediksi kelas atau labelnya dengan memanfaatkan algoritma yang telah diimplementasikan. Berikut merupakan tahap pemrosesan yang dilakukan:

1. *File* test.csv yang diberikan memuat suatu kolom id yang nantinya digunakan untuk mengidentifikasi setiap sampel data. *File* yang akan dikumpulkan pada Kaggle nantinya memuat kolom id, serta kolom price_range. Dalam hal ini, kolom price_range berisi label hasil prediksi algoritma terhadap sampel data.
2. Setiap kolom id pada *file* test.csv akan diambil dan disimpan pada suatu *dataframe* baru di dalam program.
3. Selanjutnya, setiap sampel data kemudian akan diprediksi labelnya dengan memanfaatkan algoritma KNN yang telah dilatih. Algoritma KNN dipilih untuk melakukan prediksi karena memiliki tingkat akurasi yang lebih baik dibandingkan algoritma Naive Bayes (khususnya dalam memproses data numerik). Hasil prediksi akan mengeluarkan label price_range yang dimasukkan ke dalam suatu *list*.
4. *List* berisi setiap prediksi label data selanjutnya akan dimasukkan ke dalam dataframe. Dengan demikian, dataframe terdiri atas dua kolom, yaitu kolom id dan kolom price_range.
5. *Dataframe* kemudian ditulis kembali menjadi *file* csv dengan memanfaatkan *library* yang tersedia.
6. *File* csv hasil prediksi selanjutnya dikumpulkan pada Kaggle untuk diperiksa.

Untuk menambah kemudahan dan efektivitas dalam implementasi, langkah pemrosesan di atas dijadikan suatu fitur pada program yang telah dibuat, dimana program dapat menerima *file* semacam test.csv dan mengeluarkan *file* hasil prediksi.

D. Pembagian Tugas

NIM	Kontribusi
13521056 Daniel Egiant Sitanggang	Mengimplementasikan algoritma KNN memanfaatkan pustaka yang ada, mengisi laporan bagian A.
13521062 Go Dillon Audris	Membuat *main program*, mengimplementasikan algoritma Naive Bayes *from scratch*, mengisi laporan Bagian C.
13521071 Margaretha Olivia Haryono	Mengimplementasikan algoritma Naive Bayes memanfaatkan pustaka yang ada, mengisi laporan bagian B.
13521084 Austin Gabriel Pardosi	Mengimplementasikan algoritma KNN *from scratch,* membuat README untuk *repository*.

E. Lampiran

Link Repository: AustinPardosi/Tubes_2_AI: KNN dan Naive-Bayes Algorithms (github.com)