Laporan Tugas Besar II IF3170 Inteligensi Buatan

Implementasi Algoritma KNN dan *Naive-Bayes*

Semester I tahun 2023/2024

Kelompok 1001tubes:

13521077 Husnia Munzayana

13521088 Puti Nabilla Aidira

13521111 Tabitha Permalla

13521130 Althaaf Khasyi Atisomya

Sekolah Teknik Elektro dan Informatika

Institut Teknologi Bandung

2023

Daftar Isi

I. Implementasi KNN

Algoritma *k-Nearest Neighbor* (KNN) diimplementasikan pada kelas KNN. Kelas ini mengimplementasikan algoritma pemodelan prediktif. Pendekatan pemodelan ini dapat memprediksi label target data baru dengan mencari k titik data terdekat (*k-nearest neighbors*) dalam ruang fitur berdasarkan jarak atau perbedaan nilai antara data *train* yang ada dengan data masukkan pengguna.

Proses *learning* pada pemodelan dengan KNN dilakukan dengan menyimpan seluruh data *training* dan tidak menghasilkan hipotesis apapun. Pada proses klasifikasi, KNN akan melakukan prediksi kelas target pada *unseen data* berdasarkan data *training* yang paling mirip atau memiliki jarak (*distance)* terkecil.

Atribut kelas KNN antara lain:

- k : jumlah tetangga terdekat yang akan dipertimbangkan saat melakukan prediksi.
- col_numeric: list nama kolom fitur dengan data *numerical*.
- col_non_numeric : list nama kolom fitur dengan data *non-numerical*.
- target : kolom yang menjadi label target yang akan diprediksi.

Method kelas MixedNaiveBayes antara lain:

- fit(self, train, test): mengimplementasikan proses *learning* dengan menyimpan *data training* yang digunakan dalam prediksi. Metode ini mempersiapkan data dengan memisahkan kolom fitur dan kolom target pada setiap data.
- predict(self, test, train): mengimplementasikan proses prediksi terhadap data uji (test). Dalam implementasinya, *method* ini akan memanggil *method* predict_unseen untuk setiap data uji dan mengembalikan hasil prediksi pada setiap data *test*.
- predict_unseen(self, unseen): mengimplementasikan proses prediksi terhadap sebuah data uji atau *unseen data* dengan algoritma KNN. Dalam implementasinya, *method* ini akan memanggil *method* get_distance untuk mendapatkan jarak *unseen data* dengan setiap kolom fitur terkait pada *data training* (x_train). Dari data *distance* tersebut, kemudian

diidentifikasi dilakukan *sorting* secara terurut menaik (*ascending*) untuk kemudian diambil k nilai tetangga terdekat. Dari k nilai *nearest neighbor* tersebut, diidentifikasi kelas mayoritas atau kelas yang paling banyak terdapat pada *k-nearest neighbor*. *Majority class* inilah yang menjadi hasil prediksi terhadap *unseen data.*

- get_distance(self, atrain, unseen): metode untuk menghitung jarak antara *unseen data* atau data yang ingin diprediksi dengan satu *data train*. Untuk setiap kolom fitur, dilakukan perhitungan jarak antara fitur pada *unseen data* dengan kolom fitur terkait pada *data training.* Perhitungan ini sendiri dibedakan menjadi perhitungan jarak pada kolom data *numeric* dan kolom data *non-numeric.*
- distance_non_numeric(self, atrain, unseen): metode untuk menghitung jarak pada suatu kolom data fitur *non-numeric* antara *unseen data* dan *data training*. Perhitungan jarak atau *distance* pada kolom *non-numeric* dilakukan dengan rumus sebagai berikut:

$$distance \begin{cases} 0, jika\ atrain \neq unseen\ data \\ 1, jika\ atrain = unseen\ data \end{cases}$$

- distance_numeric(self, atrain, unseen): metode untuk menghitung jarak pada suatu kolom data fitur *numeric* antara *unseen data* dan *data training*. Perhitungan jarak atau *distance* pada kolom *numeric* dilakukan berdasarkan *euclidean distance* dengan rumus sebagai berikut:

$$distance = \sqrt{(atrain\ -\ unseen)^2}$$

- accuracy(self, result, target): metode untuk menghitung akurasi dari prediksi yang dihasilkan dengan membandingkan hasil prediksi dengan target yang sebenarnya yang terdapat pada *data validation.* Fitur ini kemudian mengembalikan nilai akurasi.

Sebelum algoritma diterapkan pada data, dilakukan proses persiapan data pada *training* data. Proses persiapan data ini melibatkan beberapa langkah sebagai berikut:

- *Feature selectio*n yaitu menghapus atribut yang memiliki korelasi <=0.1 dengan atribut target. Nilai 0.1 didapat dari percobaan yang menghasilkan persentase akurasi tertinggi. Langkah ini juga diterapkan pada data yang akan di tes. Setelah seleksi, atribut yang dipakai adalah battery_power, ram, px_width, px_height.
- Penghapusan *outlier* dengan metode IQR (*Interquartile Range*). Data yang ekstrim dihapus dari data training.
- Penghapusan *missing value* yaitu dengan menghapus data yang tidak lengkap.
- Penghapusan *invalid data* yaitu menghapus data yang memiliki nilai 0. Contohnya, pada data training yang digunakan terdapat data dengan px_width bernilai 0 karena tidak mungkin terjadi.

- Normalization, dilakukan untuk memastikan setiap atribut yang digunakan dalam perhitungan jarak memiliki pengaruh yang seimbang terhadap hasil akhir.

  Rumus:

$$X = \frac{X - Xmin}{Xmax - Xmin}$$

Pada algoritma KNN ini, digunakan nilai k=9. Nilai k ini didapat dari percobaan prediksi *data validation* sehingga mendapatkan akurasi yang paling optimal.

Berikut adalah penerapan algoritma pada data_train dan data_validation.

```
# membaca data dari file csv
colnames = ['battery_power', 'blue' , 'clock_speed', 'dual_sim', 'fc',
'four_g', 'int_memory', 'm_dep', 'mobile_wt',
            'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h',
'sc_w', 'talk_time', 'three_g' , 'touch_screen',
            'wifi', 'price_range']
col_numeric =['battery_power', 'clock_speed', 'fc', 'int_memory',
'm_dep',
                  'mobile_wt', 'n_cores', 'pc', 'px_height',
```

```
'px_width',
                  'ram', 'sc_h', 'sc_w', 'talk_time']
col_non_numeric = ['blue', 'dual_sim', 'four_g', 'three_g',
'touch_screen', 'wifi']

df_train = pd.read_csv("../data/data_train.csv", header=None,
skiprows=1)
df_train.columns = colnames
df_validation = pd.read_csv("../data/data_validation.csv",
header=None, skiprows=1)
df_validation.columns = colnames

# DATA PREPARATION
# Feature Selection
def select_features(data, target_column, threshold,
feature_to_exclude=None):
    correlations =
data.corr()[target_column].abs().sort_values(ascending=False)
    correlations = correlations.drop(target_column, errors='ignore')
    if feature_to_exclude:
        correlations = correlations.drop(feature_to_exclude,
errors='ignore')
    if correlations.empty:
        selected_features =
[data.corr()[target_column].abs().idxmax()]
    else:
        selected_features = correlations[correlations >
threshold].index
    return selected_features

# Removing Outlier
def remove_outlier(data, columns):
    for col in columns:
        Q1 = data[col].quantile(0.25)
        Q3 = data[col].quantile(0.75)
        IQR = Q3 - Q1
        upper_bound = Q3 + 1.5 * IQR
        lower_bound = Q1 - 1.5 * IQR

        upper_array = np.where(data[col]>upper_bound)[0]
        lower_array = np.where(data[col]<lower_bound)[0]

        data.drop(index=upper_array, inplace=False)
        data.drop(index=lower_array, inplace=False)
    return data

# Remove Missing Value
def remove_missing_value(data, columns):
```

```
    for col in columns:
        data.dropna(subset=[col], inplace=False)
    return data

# Remove Duplicate Data
def remove_duplicate_data(data):
    data.drop_duplicates(subset=None, keep="first", inplace=False)
    return data

# Remove Invalid Data
def remove_invalid_data(data, columns):
    for col in columns:
        data = data[data[col] ≠ 0]
    return data

# Data Normalization
def normalize(data, columns):
    for col in columns:
        data[col] = (data[col] - data[col].min()) / (data[col].max() -
data[col].min())
    return data

# Data Preparation
selected_col = select_features(df_train, 'price_range',
0.1).append(pd.Index(['price_range']))
selected_col_numeric = [col for col in selected_col if col in
col_numeric]
selected_col_non_numeric = [col for col in selected_col if col in
col_non_numeric]
print("Hasil seleksi atribut", "\nNumeric: ", selected_col_numeric,
"\nNon-Numeric: ",selected_col_non_numeric)

# Data Train Preparation
df_train = df_train[selected_col]
df_train = remove_outlier(df_train, selected_col_numeric)
df_train = remove_missing_value(df_train, selected_col_numeric)
df_train = remove_duplicate_data(df_train)
df_train = remove_invalid_data(df_train, selected_col_numeric)
df_train = normalize(df_train, selected_col_numeric)

# Data Validation Preparation
df_validation = df_validation[selected_col]
df_validation = normalize(df_validation, selected_col_numeric)

# Mengaplikasikan Algoritma
knn = KNN(9, selected_col_numeric, selected_col_non_numeric,
'price_range')
knn.fit(df_train, df_validation)
```

```
predicted = knn.predict(df_validation, df_train)

print(Fore.YELLOW + '\033[1m' + "\nPredicted labels: " + '\033[0m')
predicted_table = pd.DataFrame(predicted, columns=['price_range'])
display(predicted_table)
```

II. Implementasi *Naive-Bayes*

Algoritma Naive-Bayes diimplementasikan pada kelas MixedNaiveBayes. Kelas ini mengimplementasikan kombinasi dari *Gaussian Naive-Bayes* untuk fitur *numerical* dan *Categorical Naive-Bayes* untuk fitur *categorical*.

Atribut kelas MixedNaiveBayes antara lain:

- class_probabilities: dict probabilitas prior tiap kelas/label; P(Y = c).
- feature_summaries: dict of dict *summary* tiap fitur pada tiap kelas/label.
- col_numeric: list nama kolom dengan data *numerical*.

Method kelas MixedNaiveBayes antara lain:

- fit(self, X_train, Y_train): mengimplementasikan proses *learning* dengan membangun atribut feature_summaries dengan memanggil fungsi summarize_numerical atau summarize_categorical (tergantung jenis fitur) untuk tiap fitur pada tiap kelas/label; X_train merupakan kolom-kolom fitur pada training data, Y_train merupakan kolom target pada training data.
- summarize_categorical(self, data): menghasilkan representasi probabilitas distribusi kondisional P(X | Y = c) untuk setiap nilai unik X dalam fitur *categorical* data, dengan c sebagai kelas atau label yang sesuai. Keluarannya berupa Pandas Series, dengan indeks mewakili nilai unik dalam fitur data dan nilai-nilai dalam Series mewakili probabilitas yang sesuai.

  Rumus:

$$P(X \mid Y = c) = \frac{count(X \; AND \; Y = c)}{count(Y = c)}$$

- summarize_numerical(self, data): menghitung mean dan standar deviasi dari fitur *numerical* data.
- gaussian_prob(self, x, mean, std): Menghitung probabilitas Gaussian dari x. Variabel x adalah nilai dari suatu fitur dalam baris data yang sedang dievaluasi.

  Rumus:

  $$\frac{1}{\sqrt{2\pi\sigma^2 + 10^{-9}}} \times e^{\frac{1}{2\sigma^2}(x-\varphi)^2}$$, dengan σ = std, φ = mean

- calculate_class_prob(self, class_value, row): melakukan iterasi pada tiap fitur dalam baris data, menghitung probabilitas setiap fitur berdasarkan tipe fitur (*numeric* atau *categorical*) kemudian menyimpannya dalam feature_probs. Probabilitas kelas kemudian dihitung dengan mengalikan probabilitas prior kelas dengan probabilitas setiap fitur.

  Rumus:

  $$P(Y = c \mid X) = P(Y = c) \times \prod_i P(X_i \mid Y = c)$$

  , dengan:

  P(Y = c | X) = probabilitas kelas/label c *given* nilai fitur X,

  P(Y = c) = probabilitas prior kelas/label c,

  P(Xi | Y = c) = probabilitas fitur X_i dalam kelas c.

- predict(self, X_test): memprediksi kelas/label untuk setiap baris pada *unseen* data X_test dengan memilih kelas/label yang memiliki nilai probabilitas paling tinggi.

Sebelum algoritma diterapkan, dilakukan *data preparation* pada t*raining data* dengan melakukan *feature selection*. Nilai threshold 0.035 didapat dari percobaan yang menghasilkan persentase akurasi tertinggi. Dengan nilai threshold ini, fitur yang dipilih antara lain, 'ram', 'battery_power', 'px_width', 'px_height', 'mobile_wt', dan 'blue']. Selain itu, berdasarkan percobaan, bentuk-bentuk *data preparation* lainnya tidak memberikan persentase akurasi yang lebih tinggi.

Berikut adalah penerapan algoritma pada data_train dan data_validation.

```
# Membaca Data dari File CSV
colnames = ['battery_power', 'blue' , 'clock_speed', 'dual_sim', 'fc',
'four_g', 'int_memory', 'm_dep', 'mobile_wt',
            'n_cores', 'pc', 'px_height', 'px_width', 'ram', 'sc_h',
'sc_w', 'talk_time', 'three_g' , 'touch_screen',
            'wifi', 'price_range']
col_numeric =['battery_power', 'clock_speed', 'fc', 'int_memory',
'm_dep',
                  'mobile_wt', 'n_cores', 'pc', 'px_height',
'px_width',
                  'ram', 'sc_h', 'sc_w', 'talk_time']

df_train = pd.read_csv("../data/data_train.csv", header=None,
skiprows=1)
df_train.columns = colnames
df_validation = pd.read_csv("../data/data_validation.csv",
header=None, skiprows=1)
df_validation.columns = colnames


# DATA PREPARATION
# Feature Selection
def select_features(data, target_column, threshold,
feature_to_exclude=None):
    correlations =
data.corr()[target_column].abs().sort_values(ascending=False)
    correlations = correlations.drop(target_column, errors='ignore')
    if feature_to_exclude:
        correlations = correlations.drop(feature_to_exclude,
errors='ignore')
    if correlations.empty:
        selected_features =
[data.corr()[target_column].abs().idxmax()]
    else:
        selected_features = correlations[correlations >
threshold].index
    return selected_features

selected_col = select_features(df_train, 'price_range', 0.035)
df_train = df_train[selected_col.append(pd.Index(['price_range']))]

# Mengaplikasikan Algoritma
target_column = 'price_range'
```

```
X_train = df_train.drop(columns=[target_column])
Y_train = df_train[target_column]
X_validation = df_validation.drop(columns=[target_column])

naive_bayes_model = MixedNaiveBayes(col_numeric)
naive_bayes_model.fit(X_train, Y_train)

predicted_labels = naive_bayes_model.predict(X_validation)
```

III. Perbandingan Hasil Prediksi Algoritma Implementasi dan Pustaka

Hasil prediksi dari algoritma yang diimplementasikan dibandingkan dengan hasil prediksi dari fungsi yang terdapat pada pustaka *scikit-learn*. Perbandingan ini dilakukan dengan menghitung nilai akurasi dan juga *F1-score* dari tiap-tiap model.

Model perbandingan dilatih dengan memanggil fungsi KNN dan *Naive-Bayes* pada pustaka. Model dilatih menggunakan data latih yang sudah di-*prepare*. Kemudian model melakukan prediksi terhadap data validasi. Lalu dilakukan perhitungan akurasi dan *F1-score* terhadap hasil prediksi tersebut.

Algoritma untuk membangun model menggunakan pustaka adalah sebagai berikut.

```
# KNN Scikit Model
# Splitting Data
X_train = df_train.drop(columns=['price_range'])
y_train = df_train['price_range']

x_test = df_validation.drop(columns=['price_range'])
y_test = df_validation['price_range']

# Scale data
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
x_test = scaler.transform(x_test)

# Search best k value
k_range = range(1, 100)
scores = []
y_pred = []
max_score = -1
```

```
for k in k_range:
    knn_sci = KNeighborsClassifier(n_neighbors=k)
    knn_sci.fit(X_train, y_train)
    y_temp = knn_sci.predict(x_test)
    scores.append(accuracy_score(y_test, y_temp))
    if scores[-1] > max_score:
        max_score = scores[-1]
        y_pred = y_temp

print(Fore.YELLOW + '\033[1m' + "\nBest k value: " + '\033[0m')
print(f"{k_range[scores.index(max(scores))]}")

# plot k accuracy
import matplotlib.pyplot as plt
plt.plot(k_range, scores)
plt.xlabel('K Value')
plt.ylabel('Accuracy')
plt.show()

# Result
print(Fore.YELLOW + '\033[1m' + "\nPredicted labels: " + '\033[0m')
predicted_table = pd.DataFrame(y_pred, columns=['price_range'])
display(predicted_table)

# Naive-Bayes Scikit Model
# Data preparation sama dengan Naive-Bayes hasil implementasi
scikit_nb = GaussianNB()
scikit_nb.fit(X_train, Y_train)

predicted_labels_scikit_nb =
scikit_nb.predict(X_validation[selected_col])
print(Fore.YELLOW + '\033[1m' + "\nScikit Naive Bayes Predicted
labels: " + '\033[0m')
print(predicted_labels_scikit_nb)
```

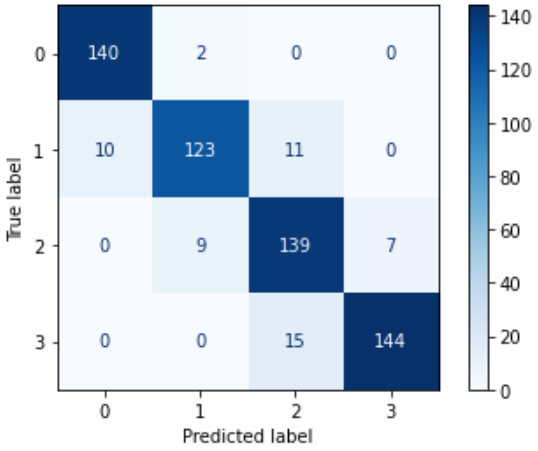
Hasil perhitungan akurasi, *precision, recall, F1-score,* dan *Confusion Matrix* tiap-tiap model adalah sebagai berikut.

Tabel 3.1 Perbandingan Akurasi dan F1 Score Hasil Implementasi dengan Scikit

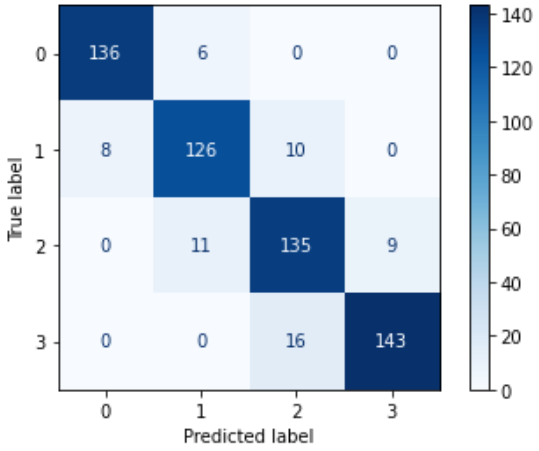
Model	KNN		*Naive-Bayes*	
	Implementasi	Scikit	Implementasi	Scikit
Akurasi	91.00%	90.00%	79.50%	79.50%

Precision	91.18%	90.12%	79.56%	79.56%
Recall	91.06%	90.08%	79.39%	79.39%
F1-*Score*	91.04%	90.08%	79.45%	79.45%

KNN Hasil Implementasi

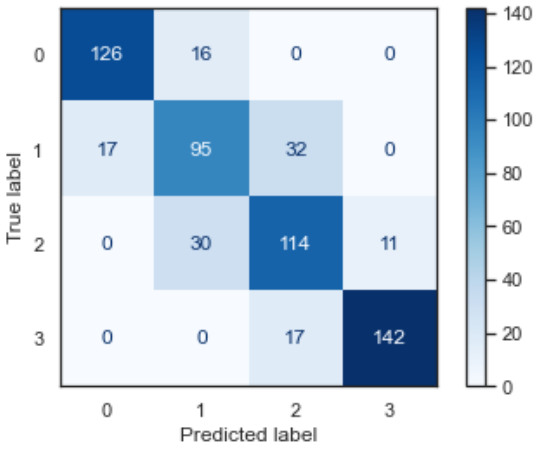


KNN Hasil Scikit

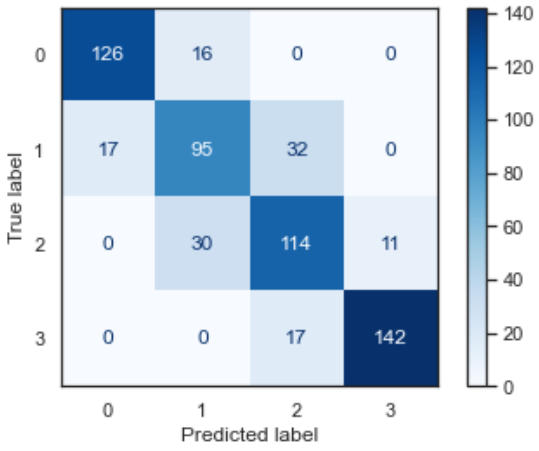


Gambar 1. Perbandingan *Confusion Matrix* KNN Hasil Implementasi dengan Scikit

Naive-Bayes Hasil Implementasi



Naive-Bayes Hasil Scikit



Gambar 2. Perbandingan *Confusion Matrix* Naive-Bayes Hasil Implementasi dengan Scikit

Dari hasil perhitungan akurasi, *precision*, *recall*, dan F1-*score* di atas, dapat disimpulkan bahwa model KNN hasil implementasi cenderung lebih akurat dibanding model KNN menggunakan pustaka. Sedangkan model Naive-Bayes hasil implementasi memiliki performa yang sama dengan model Naive-Bayes menggunakan pustaka.

IV. Kontribusi Anggota Kelompok

Tabel 4.1 Kontribusi Anggota Kelompok

Nama	NIM	Source Code	Laporan
Husnia Munzayana	13521077	knn-algo.ipynb	Bab I
Puti Nabilla Aidira	13521088	naive-bayes-algo.ipynb	Bab II
Tabitha Permalla	13521111	naive-bayes-algo.ipynb	Bab III
Althaaf Khasyi Atisomya	13521130	knn-algo.ipynb	Bab I

Referensi

Analytics Vidhya. (2021). Naive Bayes for Mixed Typed Data in Scikit-Learn. Medium. https://medium.com/analytics-vidhya/naive-bayes-for-mixed-typed-data-in-scikit-learn-fb6843e241f0

Brownlee, J. (2019). Naive Bayes Classifier From Scratch in Python. Machine Learning Mastery. https://machinelearningmastery.com/naive-bayes-classifier-scratch-python/

Brownlee, J. (2019). Naive Bayes Tutorial for Machine Learning. Machine Learning Mastery. https://machinelearningmastery.com/naive-bayes-tutorial-for-machine-learning/

Informatics Research Group School of Electrical Engineering and Informatics Institut Teknologi Bandung. (n.d.). Data Science.

Informatics Research Group School of Electrical Engineering and Informatics Institut Teknologi Bandung. (n.d.). Exploratory Data Analytics & Data Preparation.

Laraswati, B. D. (2022, September 22). *Mengenal K-nearest Neighbor dan Pengaplikasiannya*. Algoritma Data Science School. https://blog.algorit.ma/k-nearest-neighbor/

Maulidevi, Nur Ulfa. (n.d.). Modul : Supervised Learning k-Nearest Neighbor

Maulidevi, Nur Ulfa. (n.d.). Modul : Supervised Learning Naive Bayes

Link *Source Code*

https://github.com/Bitha17/Tubes2_1001tubes