Tugas Besar II IF3170 Inteligensi Buatan

Implementasi Algoritma

Disusun Oleh:

1.	Akmal Mahardika N P	(13521070)
2.	Ahmad Ghulam Ilham	(13521118)
3.	Sulthan Dzaky Alfaro	(13521159)
4.	Muhammad Habibi Husni	(13521169)

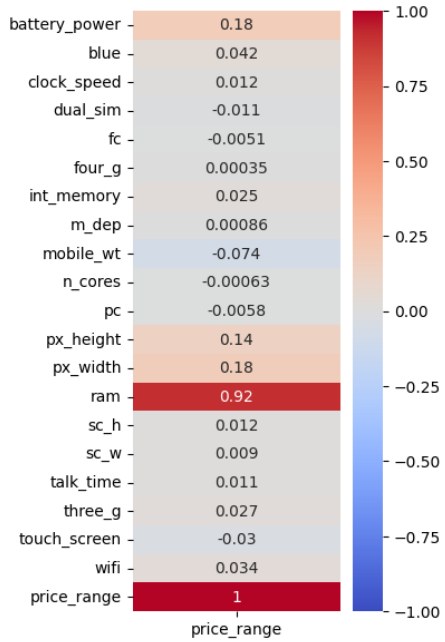
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
Tahun Ajaran 2023/2024

Implementasi Algoritma KNN

Implementasi algoritma KNN diterapkan pada kelas KNNclassifier. Algoritma ini dapat dibagi menjadi dua tahap yaitu tahap persiapan data dan tahap prediksi data. Untuk tahap persiapan data, kami melakukan normalisasi data terhadap data training dan juga data yang akan diuji agar setiap atribut berada di batas domain yang sama sehingga hasil perhitungan KNN tidak menjadi bias terhadap atribut tertentu. Metode normalisasi yang kami gunakan adalah *min-max normalization* dengan nilai min dan max diambil dari keseluruhan data training dan data test.

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

Pada tahap prediksi, kami menggunakan beberapa metode tambahan untuk memperbagus kinerja algoritma KNN yaitu *feature selection*, *feature weighting*, dan *weighted* KNN. *Feature selection* merupakan metode pemilihan beberapa feature atau atribut tertentu dalam memprediksi data. Tujuan dari *feature selection* ini adalah untuk menghindari *noise* dari atribut yang tidak terlalu signifikan terhadap klasifikasi data. Kami memilih *feature* berdasarkan nilai korelasi tiap atribut terhadap atribut 'price_range' sebagai berikut:



Berdasarkan data tersebut, kami memutuskan untuk memilih nilai atribut yang memiliki nilai korelasi $|r| > 0.1$ yaitu 'ram', 'px_width', 'battery_power', dan 'px_height'. Berikutnya, kami menggunakan metode *feature weighting* untuk memberikan bias berdasarkan nilai

korelasinya. Pada kasus ini, kami memilih weighting [0.7, 0.1, 0.1, 0.1] berurut berdasarkan atribut diatas.

Setelah normalisasi data dan memilih *feature,* berikutnya adalah melakukan prediksi dengan menggunakan algoritma KNN. Secara umum, prediksi suatu sampel dilakukan dengan tahap-tahap berikut:

1. Menghitung jarak sampel terhadap keseluruhan poin data pada data training. Fungsi jarak yang digunakan adalah *weighted euclidean distance.*

$$d(A, B) = \sqrt{\sum_i w_i (A_i - B_i)^2},$$

2. Mengurutkan data training berdasarkan nilai jarak secara menaik
3. Memilih *k neighbor* yang akan digunakan untuk melakukan *voting* nilai 'price_range'
4. Untuk penentuan nilai voting, kami menggunakan konsep *weighted* KNN, yaitu tiap poin data memiliki nilai bobot yang berbeda dalam penentuan nilai voting. Salah satu cara untuk mengimplementasikan hal tersebut adalah dengan menggunakan invers dari jarak poin data. Dengan cara ini, poin data yang lebih dekat memiliki nilai voting yang lebih besar daripada poin data yang lebih jauh.

```
score = np.zeros(4)
for i in range(self.k):
    if result['distance'].iloc[i] == 0:
        return result[self.y_label].iloc[i]
    score[result[self.y_label].iloc[i]] += 1 / (result['distance'].iloc[i])
return np.argmax(score)
```

5. Nilai prediksi 'price_range' dari sampel tersebut adalah nilai 'price_range' yang memiliki nilai voting terbanyak.

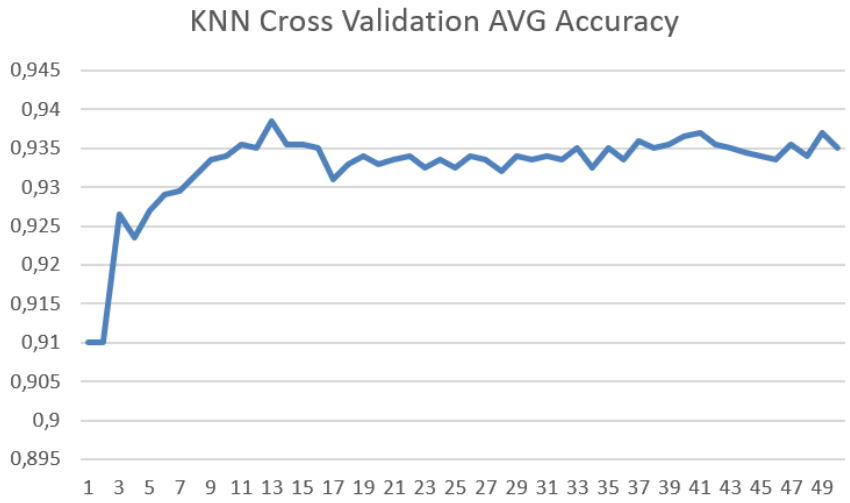
Untuk pemilihan nilai *k*, kami melakukan pengujian algoritma terhadap beberapa nilai *k* dan menilai *performance* dengan menggunakan *k-fold cross validation test*. Tes ini digunakan untuk mencegah pemilihan nilai *k* yang yang *overfitting* terhadap data yang sudah diketahui. Pemilihan nilai *k* dengan cara ini akan membantu untuk memperkirakan algoritma yang memiliki *performance* yang baik terhadap *unknown data*. Langkah yang dilakukan untuk melakukan tes adalah sebagai berikut:

1. Menggabungkan data training dan data tes menjadi data gabungan
2. Membagi data gabungan menjadi 10 grup
3. Untuk setiap grup:

a. Jadikan grup tersebut sebagai data test
b. Sisa grup digunakan sebagai data training
c. Lakukan prediksi dan dapatkan nilai akurasinya

4. Hitung nilai rata-rata akurasi sebagai nilai *performance*.

Berikut adalah hasil kinerja algoritma KNN terhadap nilai k = 1 hingga 50.



Berdasarkan hasil pengujian di atas, kami memilih nilai k = 13 yang memiliki nilai rata-rata akurasi terbaik.

Implementasi Algoritma Naive Bayes

Pada algoritma Naive Bayes, kami menggunakan Naive Bayes Classifier Tutorial: with Python Scikit-learn | DataCamp sebagai referensi untuk membuat kelas NaiveBayesClassifier yang memiliki atribut berikut.

1. Class_priors (bertipe dict): Menyimpan prior probability dari kelas-kelas pada kolom target
2. Num_feature_means (bertipe dict): Menyimpan rata-rata nilai fitur kolom numerik untuk setiap kelas
3. Num_feature_stds (bertipe dict): Menyimpan standar deviasi nilai fitur kolom numerik untuk setiap kelas
4. Cat_feature_probs (bertipe dict): Menyimpan probabilitas nilai fitur kolom non numerik untuk setiap kelas
5. Classes (bertipe np.ndarray): Menyimpan daftar kelas yang muncul pada kolom target

Nilai statistik dan probabilitas yang didapat dari data latih disimpan sebagai model dari algoritma Naive Bayes yang diimplementasikan. Model digunakan ketika melakukan prediksi terhadap data validasi dan data uji. Untuk membangun model berdasarkan data latih dan melakukan prediksi terhadap data validasi dan data uji, kami mengimplementasikan metode berikut.

1. fit(self, X, y, numerical_columns, categorical_columns): Menerima input kolom fitur dataset, kolom target dataset, daftar kolom numerik, daftar kolom non numerik. Metode fit akan menentukan daftar kelas yang muncul pada kolom target beserta prior probability dari masing-masing kelas. Kemudian, metode fit akan menentukan rata-rata dan standar deviasi nilai fitur kolom numerik. Setelah itu, metode fit akan menentukan probabilitas nilai fitur kolom non numerik.

```
def fit(self, X, y, numerical_columns, categorical_columns):
    self.class_priors = {c: np.sum(y == c) / len(y) for c in np.unique(y)}
    self.classes = np.unique(y)
    self.num_feature_means = {}
    self.num_feature_stds = {}
    for c in self.classes:
        c_mask = (y == c)
        class_features = X.loc[c_mask, numerical_columns]
        self.num_feature_means[c] = class_features.mean()
        self.num_feature_stds[c] = class_features.std()
    self.cat_feature_probs = {}
    for c in self.classes:
        c_mask = (y == c)
        class_features = X.loc[c_mask, categorical_columns]
        cat_probs = (class_features.apply(lambda x: x.value_counts(normalize=True)) + 1) / (len(class_features) + len(categorical_columns))
        self.cat_feature_probs[c] = cat_probs
```

2. save_model(self, filename): Menerima input nama file model. Metode save_model akan dipanggil setelah metode fit selesai. Metode save_model akan menyimpan atribut NaiveBayesClassifier sebagai sebuah model dalam bentuk dictionary.

```python
def save_model(self, filename):
    model_dict = {
        "class_priors": self.class_priors,
        "num_feature_means": self.num_feature_means,
        "num_feature_stds": self.num_feature_stds,
        "cat_feature_probs": self.cat_feature_probs,
        "classes": self.classes.tolist()
    }
    filename = "NaiveBayes/" + filename
    np.savez(filename, **model_dict, allow_pickle=True)
```

3. load_model(self, filename): Menerima input nama file model. Metode load_model akan dipanggil ketika prediksi data validasi dan data uji dilakukan. Metode load_model akan membuka file model dan mengubah atribut NaiveBayesClassifier menjadi value dari dictionary yang tersimpan pada model.

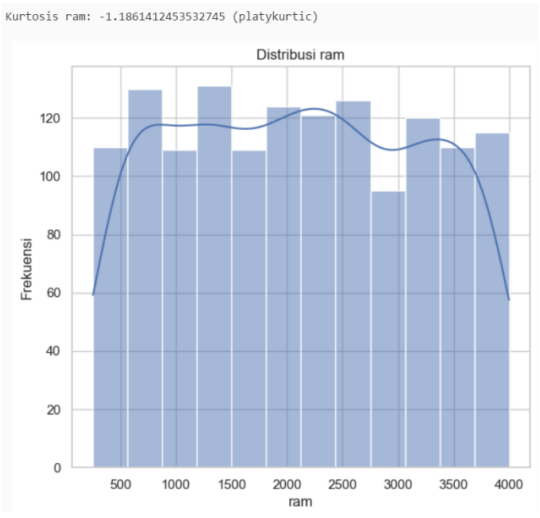
```python
def load_model(self, filename):
    filename = "NaiveBayes/" + filename
    model_data = np.load(filename, allow_pickle=True)
    self.class_priors = model_data["class_priors"].item()
    self.num_feature_means = model_data["num_feature_means"].item()
    self.num_feature_stds = model_data["num_feature_stds"].item()
    self.cat_feature_probs = model_data["cat_feature_probs"].item()
    self.classes = np.array(model_data["classes"])
```

4. predict(self, X): Menerima input kolom fitur dataset. Metode ini akan dipanggil setelah metode load_model selesai. Metode predict akan menginisialisasi sebuah array kosong bernama predictions. Kemudian, untuk setiap item pada dataset akan dilakukan kalkulasi probabilitas terhadap masing-masing kelas kolom target. Untuk kolom fitur numerik akan digunakan metode gaussian_pdf untuk menghitung probability density function nilai item, sedangkan untuk kolom fitur non numerik digunakan probabilitas nilai item. Kemudian, ditentukan class_probs sebagai probabililitas kelas merupakan hasil prediksi terhadap item. Setelah probabilitas dari masing-masing kelas didapatkan, akan dipilih predicted_class adalah kelas dengan probabilitas tertinggi untuk item tersebut. Dengan begitu, predicted_class akan ditambahkan pada array predictions yang diinisialisasi di awal. Metode predict akan mengembalikan array predictions berisi hasil prediksi terhadap label kolom target dari seluruh item pada data validasi dan data uji.

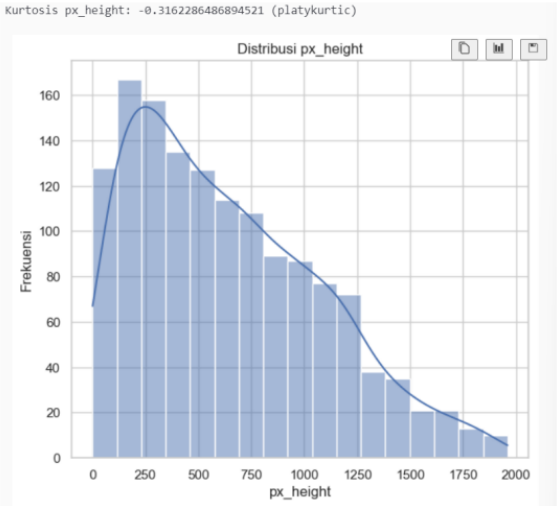
```python
def predict(self, X):
    predictions = []
    for _, instance in X.iterrows():
        class_probs = []
        for c in self.classes:
            num_probs = np.sum(np.log(self.gaussian_pdf(instance, self.num_feature_means[c], self.num_feature_stds[c])))
            cat_probs = np.sum(instance[categorical_columns].apply(lambda x: np.log(self.cat_feature_probs[c].get(x, 1))))
            class_prob = np.log(self.class_priors[c]) + num_probs + cat_probs
            class_probs.append(class_prob)
        predicted_class = self.classes[np.argmax(class_probs)]
        predictions.append(predicted_class)
    return predictions
```

5. gaussian_pdf(self, x, mean, std): Menerima input nilai fitur kolom numerik item beserta rata-rata dan standar deviasi dari seluruh kolom fitur numerik untuk kelas yang sedang dihitung. Metode gaussian_pdf digunakan untuk menghitung probability density function dari nilai fitur kolom numerik item. Justifikasi penggunaan metode gaussian_pdf pada implementasi algoritma Naive Bayes karena pada dasarnya Naive Bayes lebih optimal diterapkan pada probabilitas kondisional, sedangkan pada dataset terdapat kolom fitur numerik sehingga kami membuat implementasi algoritma yang sesuai dengan asumsi Naive Bayes, yaitu:
    a. Seluruh kolom fitur numerik memiliki distribusi normal
    b. Distribusi normal memudahkan perhitungan dan pengolahan probabilitas kolom fitur numerik yang bersifat kontinu dengan menggunakan probability density function
    c. Pendekatan Naive Bayes yang "naive" karena menganggap fitur-fitur saling independen sehingga dapat dilakukan pemisahan perhitungan antara kolom fitur numerik dan non numerik

   Namun, karena implementasi algoritma Naive Bayes yang kami buat menggunakan asumsi di atas, akurasi hasil prediksi yang didapat tidak akan optimal jika dibandingkan dengan implementasi algoritma KNN. Hal tersebut karena EDA yang dilakukan pada tucil memberikan insight bahwa mayoritas kolom fitur numerik tidak sepenuhnya memiliki distribusi normal, sebagai contoh pada kolom 'ram' dan 'px_height' berikut.



Gambar 2.5.3 Distribusi plot atribut ram



Gambar 2.5.2 Distribusi plot px_height

```python
def gaussian_pdf(self, x, mean, std):
    exponent = np.exp(-((x - mean) ** 2) / (2 * std ** 2))
    return (1 / (np.sqrt(2 * np.pi) * std)) * exponent
```

Perbandingan Hasil Prediksi Algoritma dengan Library

Perbandingan hasil prediksi dari algoritma yang telah dibuat dengan library pada python yaitu Scikit memiliki perbedaan pada nilai prediksi yang dihasilkan. Perbedaan yang didapat dari kedua algoritma cukup mendekati. Untuk algoritma Naive Bayes, memiliki perbedaan persentase akurasi yang cukup mendekati dengan library scikit. Dari algoritma Naive Bayes yang telah kami buat didapat persentase akurasi yang didapat adalah 79.5%, sedangkan persentase akurasi yang didapat dari library scikit yaitu 78.5%.

```
PS D:\AI\AI_Tubes2_AIngmaung> py calc_accuracy.py NaiveBayes/hasil.csv Data/data_validation.csv
accuracy:  0.795
```

```
PS D:\AI\AI_Tubes2_AIngmaung> python -u "d:\AI\AI_Tubes2_AIngmaung\Scikit-learn\NaiveBayes.py"
Scikit Naive Bayes: 0.785
```

Untuk Algoritma KNN, didapat perbedaan yang cukup mendekati dalam segi persentase akurasi yang didapat. Persentase akurasi yang didapat dari algoritma KNN yang telah kami buat adalah 94.8% dan persentase akurasi dari library scikit didapat 91.2%.

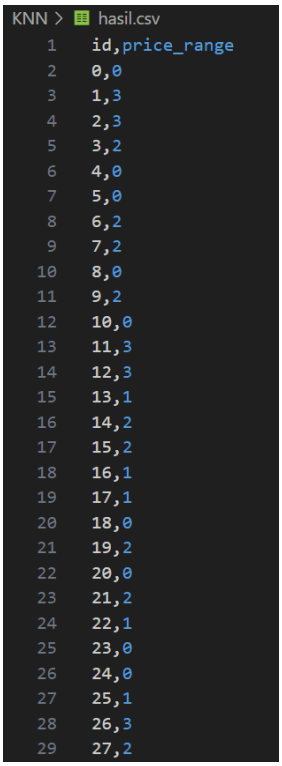
```
PS D:\AI\AI_Tubes2_AIngmaung> py calc_accuracy.py KNN/hasil.csv Data/data_validation.csv
accuracy:  0.9483333333333334
```

```
PS D:\AI\AI_Tubes2_AIngmaung> python -u "d:\AI\AI_Tubes2_AIngmaung\Scikit-learn\KNN.py"
Scikit KNN: 0.9116666666666666
```

Dapat disimpulkan bahwa terdapat perbedaan dalam nilai prediksi antara algoritma yang dibuat secara manual dan implementasi menggunakan library Scikit dalam bahasa pemrograman Python. Pada algoritma Naive Bayes, perbedaan persentase akurasi relatif kecil, dengan algoritma manual mencapai 79.5% dan library Scikit sebesar 78.5%. Namun, pada algoritma KNN, perbedaan persentase akurasi yang juga cukup kecil, yaitu 94.8% untuk algoritma manual dan 91.2% untuk library Scikit. Perbedaan ini menunjukkan pentingnya validasi dan pemahaman yang mendalam terhadap model yang dibuat, serta perluasan penelitian untuk memahami penyebab perbedaan signifikan pada algoritma KNN dan mungkin melakukan penyesuaian parameter atau perbaikan implementasi jika diperlukan.

Pemrosesan Submisi Kaggle

Untuk pemrosesan submisi pada kaggle dilakukan dengan cara membuat tim terlebih dahulu. Setelah membuat tim, jika ingin melakukan submisi sesuai dengan tim miliknya, harus terlebih dahulu mengikuti tim yang sesuai dengan kelompoknya. Setelah itu, upload hasil dari prediksi dari data kaggle dengan menggunakan algoritma yang telah dibuat lalu simpan hasilnya dalam sebuah file csv. Isi dari file csv ini berisi id dan prediksi dari price range.



Setelah didapat hasil prediksi dari data kaggle, selanjutnya input file tersebut pada submisi kaggle dan beri deskripsi. Setelah itu submit dan tunggu hasil prediksi dari kaggle.

Kontribusi Kelompok

Tugas	Kontributor
Algoritma KNN	• Muhammad Habibi Husni (13521169) • Sulthan Dzaky Alfaro (13521159)
Algoritma Naive Bayes	• Ahmad Ghulam Ilham (13521118) • Akmal Mahardika N P (13521070)
Library Python (Scikit)	• Muhammad Habibi Husni (13521169) • Sulthan Dzaky Alfaro (13521159)
Laporan	Semua

Github

https://github.com/SulthanDA28/AI_Tubes2_AIngmaung