IF3170 Inteligensi Buatan

IMPLEMENTASI ALGORITMA KKN DAN NAIVE-BAYES

Laporan Tugas Besar 2

Disusun untuk memenuhi tugas mata kuliah IF3170 Inteligensi Buatan

Tahun Akademik 2023/2024

Disusun oleh:

Muhamad Salman Hakim Alfarisi	13521010
Syarifa Dwi Purnamasari	13521018
Agsha Athalla Nurkareem	13521027
Jauza Lathifah Annassalafi	13521030

PROGRAM STUDI TEKNIK INFORMATIKA

SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA

INSTITUT TEKNOLOGI BANDUNG

BANDUNG 2023

1. Algoritma KNN

K-Nearest Neighbors (KNN) adalah sebuah metode klasifikasi terhadap sekumpulan data berdasarkan pembelajaran data yang sudah terklasifikasikan sebelumya. Termasuk dalam supervised learning, dimana hasil query instance yang baru diklasifikasikan berdasarkan mayoritas kedekatan jarak dari kategori yang ada dalam K-NN. Tujuan dari algoritma ini adalah untuk mengklasifikasikan objek baru berdasarkan atribut dan sampel-sampel dari training data.

Implementasi algoritma K-Nearest Neighbors (KNN) terdiri dari dua bagian utama: pertama, pembuatan fungsi jarak Euclidean (euclidean_distance) untuk menghitung jarak antara instance data dalam implementasi algoritma KNN, dan kedua, kelas KNN yang berfungsi sebagai model KNN. Kelas KNN ini memiliki tiga metode utama. Metode fit digunakan untuk menyimpan data pelatihan dan labelnya. Metode predict bertanggung jawab untuk memprediksi label dari data uji dengan menghitung jarak Euclidean terhadap setiap instance data pelatihan, kemudian menentukan k tetangga terdekat dan mengurutkan indeks jarak terkecil untuk mendapatkan k tetangga terdekat, lalu menggunakan label dari tetangga terdekat untuk memprediksi label instance data uji. Metode score digunakan untuk mengukur akurasi model dengan membandingkan prediksi yang dihasilkan dengan label sebenarnya. Dalam penggunaan, objek KNN dibuat dengan menentukan jumlah tetangga (k), diikuti dengan pemanggilan metode fit untuk menyimpan data pelatihan. Selanjutnya, metode predict dapat digunakan untuk melakukan prediksi pada data uji, dan metode score memberikan informasi mengenai akurasi model.

```
def euclidean_distance(x1, x2):
        distance = np.sqrt(np.sum((x1-x2)**2))
        return distance

class KNN:
    def __init__(self, k):
        self.k = k

    def fit(self, X_train, y_train):
        self.X_train = X_train
        self.y_train = y_train

    def predict(self, X_test):
        y_pred = []
        for x in X_test.values:
            dist = []
            for x_train in self.X_train.values:
                dist.append(euclidean_distance(x, x_train))
            dist = np.array(dist)
            dist_sorted = np.argsort(dist)[:self.k]
y_pred.append(np.argmax(np.bincount(self.y_train.iloc[dist_sor
ted])))
        return np.array(y_pred)

    def score(self, X_test, y_test):
```

```
        y_pred = self.predict(X_test)
        return np.sum(y_pred == y_test) / len(y_test)
```

2. Algoritma Naive-Bayes

Naive Bayes merupakan teknik klasifikasi yang didasarkan pada Teorema Bayes dengan asumsi bahwa semua fitur yang memprediksi nilai target tidak bergantung satu sama lain. Ini menghitung probabilitas setiap kelas dan kemudian memilih salah satu yang memiliki probabilitas tertinggi. Teorema Bayes menggambarkan probabilitas suatu peristiwa, berdasarkan pengetahuan sebelumnya tentang kondisi yang mungkin terkait dengan peristiwa tersebut.

Algoritma Naive-Bayes yang telah diimplementasikan memiliki empat fungsi utama yang merepresentasikan langkah-langkah Teorema Bayes. Pertama, fungsi `categorize_data` digunakan untuk mengkategorikan data pada lima kolom: `battery_power`, `mobile_wt`, `px_height`, `px_width`, dan `ram`, berdasarkan kuartil. Langkah ini membantu dalam membagi nilai-nilai numerik ke dalam interval kategorikal. Selanjutnya, fungsi `calculate_p_vj` menghitung probabilitas prior P(Vj), yaitu probabilitas munculnya setiap nilai dalam variabel target `price_range`. Kemudian, fungsi `calculate_p_ai_vj` menghitung probabilitas likelihood P(Ai|Vj) untuk setiap fitur dalam dataset. Proses ini melibatkan perhitungan probabilitas munculnya setiap nilai dalam fitur, dengan memperhatikan kelas tertentu. Terakhir, fungsi `calculate_posterior` menghitung probabilitas posterior P(Vj|A), yang merepresentasikan probabilitas munculnya setiap kelas kondisional pada nilai-nilai fitur tertentu. Proses ini melibatkan perkalian antara probabilitas prior dan likelihood. Model ini kemudian dapat digunakan untuk melakukan prediksi pada data validasi menggunakan fungsi `predict`, dan akurasi prediksi dapat diukur dengan fungsi `accuracy`. Dengan langkah-langkah ini, algoritma Naive-Bayes dapat memberikan prediksi yang informatif dan dapat diandalkan pada data dengan struktur yang sesuai.

```
class NaiveBayes:
    def __init__(self):
        self.data_train =
pd.read_csv("../../data/data_train.csv")
        #Memilih kolom yang paling berpengaruh pada penilaian
        self.data_train = self.data_train[["battery_power",
"mobile_wt", "px_height", "px_width", "ram", "price_range"]]
        #Mengubah kolom dari nilai numerik menjadi nilai
kategorikal
        self.categorical_data_train =
self.categorical(self.data_train)

    def categorical(self, dataset):
        batas = [[864.75, 1219.9, 1655.75], [111.0, 143.0,
172.0], [307.0, 576.5, 928.5], [874.0, 1249.0, 1639.75],
[1220.75, 2286.0, 3114.5]]
        nama = ["battery_power", "mobile_wt", "px_height",
"px_width", "ram"]
        for i in range(len(batas)):
```

```
            dataset.loc[dataset[nama[i]] <= batas[i][0],
nama[i]] = 0
            dataset.loc[(dataset[nama[i]] > batas[i][0]) &
(dataset[nama[i]] <= batas[i][1]), nama[i]] = 1
            dataset.loc[(dataset[nama[i]] > batas[i][1]) &
(dataset[nama[i]] <= batas[i][2]), nama[i]] = 2
            dataset.loc[dataset[nama[i]] > batas[i][2],
nama[i]] = 3
        return dataset

    def p_vj(self):
        list = []
        for i in np.unique(self.data_train["price_range"]):
            count = sum(self.data_train["price_range"] == i)
            list.append(count/len(self.data_train))
        return list

    def p_ai_vj(self):
        nama = ["battery_power", "mobile_wt", "px_height",
"px_width", "ram"]
        list = []
        for i in range(len(nama)):
            kolom = []
            for j in
np.unique(self.data_train["price_range"]):
                matrix = []
                count = sum((self.data_train["price_range"] ==
j) & (self.data_train[nama[i]] == 0))

matrix.append(count/sum(self.data_train["price_range"] == j))
                count = sum((self.data_train["price_range"] ==
j) & (self.data_train[nama[i]] == 1))

matrix.append(count/sum(self.data_train["price_range"] == j))
                count = sum((self.data_train["price_range"] ==
j) & (self.data_train[nama[i]] == 2))

matrix.append(count/sum(self.data_train["price_range"] == j))
                count = sum((self.data_train["price_range"] ==
j) & (self.data_train[nama[i]] == 3))

matrix.append(count/sum(self.data_train["price_range"] == j))
                kolom.append(matrix)
            list.append(kolom)
        return list

    def posterior(self, data_validation):
        prior = self.p_vj()
        likelihood = self.p_ai_vj()
        data_validation = self.categorical(data_validation)
        nama = ["battery_power", "mobile_wt", "px_height",
```

```
"px_width", "ram"]
        list = []
        for i in range(len(data_validation)):
            matrix = []
            for j in range(len(prior)):
                matrix.append(prior[j])
                for k in range(len(nama)):
                    matrix[j] = matrix[j] *
likelihood[k][j][data_validation[nama[k]][i]]

            list.append(matrix)
        return list

    def predict(self, data_validation):
        Posterior = self.posterior(data_validation)

        list = []
        for i in range(len(Posterior)):
            list.append(np.argmax(Posterior[i]))
        count = 0
        for i in range(len(data_validation)):
            if list[i] == data_validation["price_range"][i]:
                count += 1
        return list, count/len(data_validation)

    def get_result(self, data_val, index):
        posterior = self.posterior(data_val)
        list = []
        for i in range(len(Posterior)):
            list.append(np.argmax(Posterior[i]))

        return list[index]

    def save_model(self, filename):
        pickle.dump(self, open("../model/"+ filename, 'wb'))

categorical_data_validation =
pd.read_csv("../../data/data_validation.csv")
model = NaiveBayes()
hasil, persentase = model.predict(categorical_data_validation)
coba = model.get_result(categorical_data_validation, 2)
model.save_model("NB_model.pkl")
```

3. Perbandingan Algoritma Yang Diimplementasikan Dengan Menggunakan Pustaka

a. Algoritma KNN

Hasil akurasi dari implementasi KNN yang telah dibuat adalah 0.9428571428571428 dan hasil akurasi yang didapatkan menggunakan pustaka adalah 0.9428571428571428. Dari perbandingan antara hasil algoritma KNN yang telah diimplementasikan dengan hasil yang diperoleh menggunakan pustaka scikit-learn, terlihat bahwa keduanya memberikan akurasi yang sama, yaitu 0.9428571428571428. Hasil ini menunjukkan bahwa implementasi KNN yang telah dibuat mencapai kinerja yang setara dengan implementasi pustaka scikit-learn pada dataset yang digunakan.

Insight yang dapat diperoleh dari perbandingan ini adalah hasil yang serupa antara implementasi KNN yang sudah dibuat dan implementasi pustaka menunjukkan algoritma KNN berhasil dibuat dan memberikan keyakinan bahwa implementasi memiliki kinerja yang baik pada dataset.

b. Algoritma Naive-Bayes

Hasil akurasi dari implementasi Naive-Bayes yang telah dibuat adalah 0.73 dan hasil akurasi yang didapatkan menggunakan pustaka adalah 0.79. Dari perbandingan antara hasil algoritma Naive-Bayes yang telah diimplementasikan dengan hasil yang diperoleh menggunakan pustaka scikit-learn, terlihat bahwa keduanya memberikan akurasi yang berbeda. Hasil ini menunjukkan bahwa implementasi Naive-Bayes yang telah dibuat belum sempurna, karena keterbatasan dalam menentukan range kategori pada data train.

4. Kontribusi Setiap Anggota Dalam Kelompok

Task	NIM
KNN	13521018, 13521030
Naive-Bayes	13521010, 13521027

5. Lampiran

github: https://github.com/syrifaa/TubesAI2_KingSalman