Richard Rivaldo
13519185 – K04

Unsupervised Learning

1. Apa itu unsupervised learning?

   Algoritma *Unsupervised Learning* merupakan algoritma yang merupakan `lawan` dari algoritma *Supervised Learning*. Pada algoritma ini, model belajar dari data yang sama sekali tidak mempunyai label apapun. Model kemudian berusaha untuk mencari pola tertentu yang ada pada data tersebut sehingga nantinya bisa memberikan label pada setiap data yang ada.
   Pada dasarnya, algoritma ini berbasis pada kemampuan model dalam mencari kemiripan suatu data dengan data lain. Tujuan pelabelan pada algoritma ini adalah untuk memasukkan data ke dalam kategori atau klaster tertentu. Dengan demikian, persoalan utama dari algoritma ini adalah persoalan *clustering* atau klasterisasi, yaitu mengklasterisasikan data tidak berlabel ke dalam kategori-kategori tertentu berdasarkan pola dan struktur informasi yang didapat pada setiap data. Di sisi lain, algoritma ini juga bisa digunakan untuk mengetahui *density estimation*, yaitu mempelajari bagaimana sekumpulan data terdistribusi berdasarkan kriteria tertentu.
   Selain klasterisasi, *Unsupervised Learning* juga bisa digunakan untuk permasalahan asosiasi. *Association Rule* adalah metode *unsupervised learning* untuk mencari hubungan antara variabel-variabel yang ada pada sebuah data. Dalam hal ini, *association rule* menentukan himpunan asosiasi *item* yang berlangsung secara bersamaan antara dua buah variabel. Hal ini sering digunakan dalam strategi pemasaran, misalnya jika seseorang membeli X, maka berdasarkan *rule* yang ada, orang tersebut juga akan membeli Y.
   Permasalahan *clustering* sendiri bisa dibagi menjadi beberapa bagian, di antaranya adalah *exclusive* dan *overlapping clustering*, *hierarchical clustering*, dan *probabilistic clustering*. Adapun untuk asosiasi juga terbagi menjadi beberapa sub-algoritma, misalnya adalah algoritma *a priori*, reduksi dimensi (*dimensionality reduction*), *Principal Component Analysis* (PCA), *Singular Value Decomposition*, *Self-Organizing Maps*, dan *Autoencoders* yang berbasis *neural network* pada *Deep Learning*.
   Keuntungan dari algoritma ini terletak pada ketiadaan label itu sendiri. Pada saat ini, masih jauh lebih mudah untuk mendapatkan data tanpa label dibandingkan data yang sebaliknya. Dengan demikian, algoritma ini bisa digunakan untuk permasalahan kompleks yang memiliki data tanpa label. Namun, algoritma ini cenderung menghasilkan resiko yang tinggi pada inakurasi hasil yang diberikan. Selain itu, peran manusia dalam memvalidasi keluaran juga rentan terhadap kesalahan. Adapun algoritma ini juga sulit memberikan transparansi terhadap basis yang digunakan untuk klasterisasi dan asosiasi yang dilakukan.

2. Jelaskan bagaimana cara kerja dari algoritma yang anda implementasikan!
   a. DBScan
      DBScan atau *Density-Based Spatial Clustering for Applications with Noise* adalah salah satu algoritma *clustering* yang meggunakan dua buah parameter utama. Algoritma ini dimodelkan sebagai sebuah kelas dengan beberapa atribut. Berbeda dengan algoritma *Supervised* Learning, atribut *dataset* yang digunakan tidak dibagi menjadi *training* dan *test set*. Setelah *dataset* berhasil dimuat dan dibaca serta diubah ke dalam *array* Numpy, kemudian ditambahkan kolom baru yang nilainya diinisialisasi dengan nilai -1 untuk menandakan bahwa data tersebut belum diklasterisasikan.

Atribut lain yang dibutuhkan adalah dua parameter utama algoritma ini, yaitu *epsilon* dan *minimum points*. Epsilon dapat dikatakan sebagai radius dari lingkaran semu yang membentuk suatu klaster. Suatu lingkaran dikatakan sebagai sebuah klaster apabila jumlah data yang dilingkupinya adalah paling sedikitnya jumlah dari *minimum points*. Selain kedua atribut tersebut, ada juga atribut untuk menyimpan label untuk *cluster* dan *noise label*. Atribut *cluster label* digunakan untuk melakukan *tracking* terhadap jumlah klaster yang sudah dibuat, sedangkan *noise label* adalah atribut konstan untuk data-data yang bersifat *noisy*.

Untuk menentukan tetangga dari suatu data, maka metriks yang digunakan dalam perhitungan adalah jarak. Dalam implementasinya, digunakan jarak *Euclidean* serta secara jarak *Hamming* karena ada satu fitur kategorikal yang bersifat biner. Dengan demikian, hasil perhitungan dari jarak sebuah data adalah jarak *Euclidean* ditambah dengan 1 apabila fitur kategorikal kedua data tidak sama. Perhitungan jarak dilakukan untuk semua fitur terkecuali fitur label tambahan yang ditambahkan sebelumnya.

Untuk mencari semua data yang bisa masuk ke dalam hubungan ketetanggaan dengan suatu data, maka semua data diiterasi dan dihitung jaraknya dengan data tadi. Jika perhitungan jarak ini ternyata tidak melebihi nilai parameter *epsilon*, maka dapat dikatakan bahwa data tersebut bertetangga dengan data awal tadi. Untuk mempermudah komputasi, maka data yang disimpan adalah data indeks dari data tetangga tadi di dalam *dataset*. Suatu data sudah pasti merupakan tetangga dengan dirinya sendiri, dan data ini juga masuk ke dalam perhitungan *minimum points*.

Proses *fitting* dilakukan dengan melakukan iterasi terhadap semua data yang ada, dan untuk menyederhanakan algoritma, proses iterasi dimulai dari data pertama (tidak diacak). Jika label data tidak lagi bernilai -1, berarti data tersebut sudah diklasifikasikan, sehingga data ini bisa langsung dilewati dalam pengecekannya. Untuk suatu data tertentu, dicari semua tetangga yang dimiliki oleh data ini. Kemudian, jika banyak tetangga dari data ini kurang dari *minimum points*, maka data ini diklasterisasi sementara sebagai *noise* dan dilewati. Jika tidak, artinya sebuah klaster bisa dibentuk dan dengan demikian data tersebut merupakan *core points* yang bisa dilabeli dengan label *cluster* saat ini. Nilai label akan diinkremen terlebih dahulu, sehingga *noise* memiliki label 0 dan klaster pertama dimulai dari indeks satu.

Setelah tahap tersebut, algoritma dilanjutkan dengan tahap ekspansi yang biasanya diimplementasikan secara rekursif, tetapi pada implementasi yang dibuat dilakukan secara iteratif. Ekspansi ini dilakukan untuk mencari sebanyak-banyaknya *border points*, yaitu data yang *density*-reachable, yaitu secara tidak langsung bisa dicapai oleh suatu data tertentu melalui hubungan ketetanggaan dengan data lain. Ekspansi ini dilakukan dengan menggunakan bantuan sebuah *neighbor seed* dan *queue* berisikan daftar semua *neighbor* yang telah didapat.

Di dalam iterasi ini akan dilakukan pelabelan untuk semua *neighbor* yang ada. Jika data yang sedang dicek dari hasil *pop* terhadap *queue* tersebut ternyata telah dilabeli sebagai *noise*, artinya label data ini bisa diubah dan dimasukkan ke dalam klaster yang sama dengan data awal. Jika data yang sedang diperiksa sudah dilabeli selain -1 (belum diklasterisasi) dan 0 (*noise* pada pengecekkan sebelumnya), maka data ini bisa dilewati. Jika tidak, maka data bisa diklasterisasi dengan data awal.

Setelah itu, data tetangga ini kemudian diekspansi dan dicari semua tetangga dari data tetangga ini. Proses yang serupa kemudian dilakukan untuk tetangga dari data

tetangga tadi, dengan cara memasukkan tetangga tersebut ke dalam *seed* dan *queue* sebelumnya.
Jika pada *dataset* tersebut ingin ditambahkan data baru, atau dalam kata lain, ingin diketahui klaster suatu data baru pada *dataset* tersebut, maka prosesnya adalah sebagai berikut. Akan dicari semua tetangga dari data masukan dan tetangga ini akan diurutkan berdasarkan jaraknya dengan data masukan terurut menaik. Jika banyak tetangganya kurang dari nilai *epsilon*, maka data masukan dilabeli sebagai *noise*. Jika tidak, maka dari data terdekat, akan diperiksa apabila data tetangga tadi merupakan *core point*. Jika ya, maka data masukan akan diklasteriasasi dengan data tetangga tadi. Jika tidak ada data tetangga yang merupakan *core point*, maka data masukan diklasterisasi dengan data tetangga terdekat.

b. KMeans

Algortima KMeans dimuat dalam sebuah model yang memiliki beberapa atribut. Atribut pertama berupa *dataset* yang sama dengan algoritma DBScan sebelumnya. Namun, pada model ini *dataset* hanya mengandung fitur numerikal saja, tidak seperti DBScan yang juga mengandung atribut kategorikal berupa *gender*. Hal ini dikarenakan kalkulasi *error* untuk algoritma ini nantinya bisa menghasilkan *division by zero* akibat *mapping* kategori yang dilakukan sebelumnya.
Selain itu, model juga menerima parameter *k* yang menyatakan jumlah klaster yang nantinya diproses oleh model. Di dalam algoritma ini juga ditentukan beberapa konstanta secara *default*, yaitu *maximum iteration* dan *tolerance value*. *Maximum iteration* menyatakan jumlah iterasi maksimum yang bisa dilakukan oleh model ketika melakukan *clustering*. Adapun *tolerance value* merupakan *threshold* maksimum dari *sum of squared error* dari model.
Metriks jarak yang digunakan di dalam algoritma ini merupakan jarak *Euclidean*. Implementasi komputasi jarak ini dilakukan dengan menggunakan bantuan metode *norm* pada pustaka Numpy yang bisa menghitung jarak *Euclidean* dari dua buah data.
Proses *fitting* algoritma ini dimulai dengan menginisiasi *k* buah sentroid. Sentroid ini merupakan subset data yang dipilih dari *dataset* awal dan sentroid dapat dikatakan sebagai pusat rataan data di dalam sebuah klaster. Setelah inisialisasi sentroid dilakukan, proses dilanjutkan dengan menginisiasi klaster kosong untuk nantinya menampung semua data yang ada. Sesuai dengan penjelasan sebelumnya, jumlah klaster yang dihasilkan haruslah sesuai dengan parameter *k* yang diberikan. Kemudian, semua data yang ada di dalam *dataset* akan diklasterisasi berdasarkan jaraknya dengan semua sentroid yang ada. Sentroid yang memiliki jarak terdekat dengan data tersebut akan mengambil data tersebut ke dalam klasternya. Setelah dilakukan klasterisasi, maka sentroid lama disalin terlebih dahulu. Sentroid kemudian bisa diperbarui, dan nilai sentroid yang baru adalah nilai rataan semua data yang ada di dalam klaster sentroid tersebut.
Setelah memperbarui sentroid, maka akan diperiksa apabila pembaruan yang dilakukan berhasil membuat algoritma konvergen. Hal ini dilakukan dengan mengecek apabila nilai *error* yang terdapat antara sentroid baru dengan sentroid lama sudah tidak melebihi nilai tolernasi yang ditetapkan sebelumnya. *Error* dihitung dengan mengurangi sentroid baru dengan sentroid awal, lalu mencari rasio selisih tersebut dengan sentroid lama dan akhirnya dikalikan 100.
Jika jumlah setiap elemen di dalam hasil kalkulasi tersebut lebih besar dari nilai toleransi, maka artinya algoritma belum konvergen dan iterasi masih bisa dilakukan, sampai jumlah iterasi mencapai batas *maximum iteration*, atau sampai

algoritma mencapai konvergensi. Jika sebaliknya, maka algoritma sudah berhasil mencapai konvergensi dan sudah bisa dihentikan.
Untuk mengklasterisasi data masukan baru, maka hanya perlu diperhitungkan jarak antara semua sentroid akhir yang dihasilkan dari proses *fitting* dengan data masukan tersebut. Data ini akan masuk ke dalam klaster yang sentroidnya memiliki jarak terdekat dengan dirinya.

c. KMedoids

Algortima KMedoids terdiri dari beberapa jenis, misalnya adalah *Partition Around Medoids* yang umum digunakan, atau juga CLARA dan CLARANS yang menggunakan *resampling* terlebih dahulu dari *dataset*. Dalam implementasi ini, algoritma KMedoids yang digunakan adalah algoritma PAM. Algoritma ini dimodelkan dalam sebuah kelas yang memiliki atribut berupa *dataset* yang dibaca. *Dataset* yang diterima hanya menerima masukan data dengan fitur numerik saja. Selain itu, model juga akan memiliki atribut *k* yang menyatakan jumlah klaster sekaligus jumlah *medoids* yang digunakan oleh algoirtma.
Di dalam algoritma ini, metriks jarak yang digunakan untuk mengukur jarak dari suatu titik ke titik lain adalah jarak *Euclidean*. Seperti biasa, metriks jarak ini diimplementasikan dengan menggunakan *norm* yang terdapat pada bantuan pustaka aljabar linear Numpy.
Algoritma dimulai dengan menginisiasi sejumlah *k* buah medoids yang dipilih secara acak dari *dataset* yang ada. Medoids merupakan subset data yang ada dari *dataset* yang dipakai sebagai data referensi untuk proses *clustering*, mirip dengan sentroid pada algoritma KMeans.
Setelah medoids telah diinisialisasi, maka kemudian data akan diklasterisasi berdasarkan medoids tersebut. Proses klasterisasinya adalah dengan mengalokasikan data tersebut ke medoids dengan tingkat disimilaritas terkecil. Disimilaritas ini tidak lain dapat diukur dengan menggunakan metriks *Euclidean* tadi. Artinya, akan dicari medoid dengan jarak terdekat terhadap data tersebut, utnuk nantinya data ini dimasukkan ke dalam klaster medoid tersebut.
Dari proses klasterisasi tadi, maka sudah bisa didapatkan *total cost* akibat klasterisasi tersebut. *Total cost* yang dimaksud adalah total *error* atau total disimilaritas yang didapat dari suatu data dengan medoid tempat data tersebut diklasterisasikan. *Total cost* dihitung pada keseluruhan klaster dengan menggunakan metriks yang sama, sehingga dihasilkan perbedaan jarak total antara semua data dengan medoid nya.
Setelah kalkulasi *cost* telah dihitung, maka selanjutnya algoritma akan berusaha untuk mencari medoids yang lebih cocok. Untuk setiap medoid yang ada, algoritma akan mencoba mengganti medoid tersebut dengan satu per satu data non-medoid yang ada pada *dataset*. Berdasarkan medoids baru ini, akan dilakukan klasterisasi dan perhitungan *total cost* menggunakan cara yang sama dengan sebelumnya.
Jika *total cost* hasil perubahan medoid tersebut ternyata lebih kecil dari biaya yang dikeluarkan sebelumnya, maka medoids dan *total cost* hasil perubahan tersebut akan dicatat. Pencatatan ini akan digunakan untuk membandingkan pergantian medoids yang diproses satu per satu.
Setelah proses tersebut, maka mungkin saja beberapa atau bahkan seluruh medoids mengalami perubahan, atau bahkan tidak sama sekali. Untuk memeriksa apabila proses tersebut telah menyebabkan algoritma mencapai konvergensi, maka dibandingkan *total cost* awal dengan *total cost* dari hasil proses sebelumnya. Jika ternyata *total cost* yang baru memiliki nilai yang lebih besar atau sama dengan *total*

cost lama, maka artinya medoids baru yang diproses tadi tidak memberikan hasil tingkat disimilaritas yang lebih kecil. Dengan demikian, hal tersebut menunjukkan bahwa medoids lama sudah cukup baik sehingga algoritma bisa diselesaikan. Jika sebaliknya, maka algoritma akan terus berlanjut, dengan tentunya mengganti *total cost* dan medoids lama dengan *total cost* dan medoids yang baru.
Proses ini merupakan proses yang sangat iteratif dan memiliki kompleksitas waktu yang tinggi. Hal inilah yang menyebabkan algoritma ini kurang cocok digunakan untuk data yang berukuran besar. Kekurangan ini menyebabkan munculnya model CLARA dan CLARANS. Setelah proses tersebut berakhir, maka kemudian dilakukan klasterisasi lagi menggunakan medoids hasil proses tersebut.
Proses prediksi klaster dari input data baru mirip dengan proses yang ada pada algoritma KMeans. Dengan menggunakan data yang dihasilkan secara acak, akan dikalkulasi jarak antara semua medoids yang ada dengan data masukan tersebut. Klaster dari data ini adalah klaster tempat medoids dengan jarak terdekat dengan data masukan tersebut berada.

3. Bandingkan ketiga algoritma tersebut, kemudian tuliskan kelebihan dan kelemahannya!

Algoritma DBScan merupakan algoritma berbasis densitas poin pada suatu klaster, sedangkan algoritma KMeans dan KMedoids merupakan algoritma klasterisasi berbasis partisi data berdasarkan metriks tertentu. Secara komputasional, dapat dikatakan bahwa waktu komputasi untuk algoritma DBScan lebih kecil dibandingkan untuk algoritma KMeans dan KMedoids, meskipun hal ini tetap bergantung pada data dan algoritma yang digunakan.
Di sisi lain, KMeans bertujuan untuk meminimalisir *sum squared error*, sedangkan KMedoids bertujuan untuk meminimalisir disimilaritas dari data yang ingin diklasterisasi dengan data pusat dari suatu klaster tertentu. Kedua algoritma merupakan algoritma yang memiliki *a priori* yang telah ditentukan, yaitu konstanta *k* yang menandakan jumlah klaster pada algoritma tersebut. Hal ini tidak berlaku pada algoritma DBScan, yang jumlah klasternya tidak diketahui *a priori* dan bertumbuh secara dinamis sesuai dengan data yang diberikan, juga berdasarkan parameter awal algoritma tersebut.
Pada KMeans dan KMedoids, data acuan yang digunakan dalam penentuan hubungan ketetanggaan untuk membuat sebuah klaster merupakan data yang terdapat di dalam *dataset* dan data ini biasanya dipilih secara acak untuk memulai proses yang ada. Namun, data pembaruan data acuan pada ketiga algoritma tersebut memiliki proses yang berbeda-beda. Misalnya, pada KMeans maka data acuan akan diperbarui dengan nilai rataan data di suatu klaster, sedangkan pada KMedoids, data acuan adalah data yang bisa meminimumkan disimilaritas. KMeans juga lebih cenderung terkena dampak oleh *outlier* dan *noises* pada data, tidak seperti DBScan dan KMedoids.

Algoritma	Kelebihan	Kelemahan
DBScan	1. Algoritma ini hanya membutuhkan dua buah parameter untuk di-*tune* dan tidak membutuhkan jumlah klaster sejak awal. Algoritma ini juga hanya membutuhkan *domain knowledge* yang minimum	1. Algoritma ini dapat dikatakan sebagai algoritma yang non-deterministik. 2. Algoritma ini tidak terlalu baik dalam menentukan klaster data-data berjarak dekat, namun memiliki

	untuk menentukan parameter tersebut. 2. Algoritma ini tidak terdampak oleh keberadaan *outliers* dan *noises* yang ada pada data. 3. Oleh karena itu, algoritma ini bisa mendapatkan bentuk klaster yang mungkin tidak bisa didapatkan algoritma KMeans dan KMedoids.	tingkat densitas yang jauh berbeda, sehingga bisa mengakibatkan terbentuknya satu klaster besar, atau banyak sekali klaster kecil, tergantung dari nilai *epsilon*. 3. Tidak terlalu baik untuk data yang berjumlah banyak dan berdimensi tinggi.
KMeans	1. Bisa diimplementasikan dengan cukup mudah dan bekerja dengan baik serta efisien untuk data berukuran besar. 2. Menjamin adanya konvergensi hasil klasterisasi dan akan memberi hasil yang lebih baik jika memiliki distribusi Gaussian. 3. Mudah beradaptasi dengan data baru dan bisa melakukan generalisasi untuk bentuk-bentuk klaster tertentu. 4. Sederhana dan cukup fleksibel dalam mengubah klaster yang sudah terbentuk.	1. Jumlah klaster dalam bentuk parameter *k* harus ditentukan terlebih dahulu dan model cenderung dependen terhadap nilai ini. 2. Kesulitan dalam melakukan klasterisasis untuk data dengan ukuran dan densitas yang berbeda. 3. Sentroid bisa tergeser oleh *outlier*, bahkan *outlier* bisa membentuk sentroid tersendiri karena menggunakan *mean* data sebagai sentroidnya. 4. Adanya kemungkinan terkena *curse of dimensionality*.
KMedoids	1. KMedoids, terutama KMedoids PAM, dapat dikatakan cukup baik dalam menghadapi *outlier* dengan *medoids* dibandingkan KMeans. 2. Dapat diimplementasi dan dimengerti dengan cukup sederhana. 3. KMedoids tergolong sebagai algoritma yang cukup cepat dan selalu konvergen.	1. KMedoids cenderung kesulitan dalam melakukan klasterisasi untuk klaster dengan bentuk *non-spherical* (*arbitrary shaped*). 2. Algoritma yang non-deterministik karena *medoids* awal dipilih secara acak. 3. Membutuhkan parameter awal *k* sejak awal, sama seperti KMeans.

Richard Rivaldo
13519185 – K04

4. Jelaskan penerapan dari algoritma unsupervised! (misal di bidang kesehatan atau industri)

   Salah satu anak cabang Google, *Google News,* menggunakan algoritma *Unsupervised Learning* untuk mengelompokkan berita-berita dengan topik yang sama ke dalam satu kategori tertentu. Dengan demikian, berita pada topik yang sama lebih mudah dicari. Selain itu, banyak sekali aplikasi algoritma ini pada berbagai macam pengolahan citra dan *computer vision*, misalnya pada persepsi visual mesin, *image recognition*, dan *image segmentation*. Hal ini sangat bermanfaat dan berdampak besar pada perkembangan alat-alat radiologi dan patologi untuk memberikan diagnosis yang lebih cepat dan akurat.

   Algoritma ini juga bisa digunakan untuk mencari dan mendeteksi anomali yang ada pada suatu data yang berukuran besar. Hal ini dapat dilakukan dengan mencari data yang posisi dan pengelompokkannya jauh berbeda dengan semua kelompok yang lain, sehingga dapat dikatakan data ini merupakan *noise* atau *outlier*. Hal ini tidak saja berguna untuk statistik, tetapi juga untuk pertahanan keamanan dari *security breaching* dan penjagaan kualitas suatu peralatan beresiko tinggi.

   Dalam dunia bisnis, algoritma ini juga bisa digunakan untuk melakukan analisis terhadap konsumen yang memakai atau menggunakan servis tertentu. Hal ini disebut dengan *customer segmentation*. Dengan mengklasterisasi semua pelanggan berdasarkan kelompok tertentu, maka perusahaan bisa menarik keputusan bisnis terkait dengan pembuatan atau peluncuran produk baru, atau strategi *pricing* sehingga bisa meningkatkan *return of investment* yang lebih baik.

   Terakhir, Netflix dan banyak aplikasi lainnya seperti Spotify menggunakan algoirtma ini sebagai bentuk *recommendation system*. Dengan menggunakan data pengguna pada transaksi-transaksi yang dilakukan sebelumnya, algoritma ini bisa mencari *trend* yang sedang dimiliki oleh pengguna tersebut. Hal ini menyebabkan perusahaan bisa memperkenalkan produk baru yang mungkin juga disukai dan akan dipakai oleh pengguna. Penggunaan sistem rekomendasi yang relevan akan sangat membantu perusahaan dalam membangun dan memperluas jangkauan pasar baru.

Richard Rivaldo
13519185 – K04

Referensi Lain

Comparison of K Means, K Medoids, DBSCAN Algorithms Using DNA Microarray Dataset oleh C. Kondal Raj.

DBSCAN Advantages and Disadvantages oleh TheDataPost.

Difference between K-Means and DBScan Clustering oleh GeeksforGeeks.

k-Means Advantages and Disadvantages oleh Google Developers.

K-Means and K-Medoids oleh University of Leicester.

K-means, DBSCAN, GMM, Agglomerative clustering — Mastering the popular models in a segmentation problem oleh Indraneel Dutta Baruah.

K-medoids Clustering oleh Nikunj Bansal.

ML | K-Medoids clustering with solved example oleh GeeksforGeeks.

Unsupervised Learning oleh MathWorks.

Unsupervised Machine Learning oleh JavatPoint.

What is Unsupervised Learning? oleh IBM Cloud Education.