Tugas Besar Machine Learning

*Clustering dan Classification*



Disusun oleh

Muhammad Faisal Amir

1301198497

IFX-43-Gab

FAKULTAS TEKNIK INFORMATIKA

TELKOM UNIVERSITY

2020

**Persiapan Data**

* Dataset : air\_bnb.csv (22586 baris)
* Kolom tabel yang di pilih : ['room\_type', 'price', 'minimum\_nights', 'number\_of\_reviews'], alasan memilih kolom tabel tersebut yaitu saling berhubungan untuk dilakukannya proses clustering
* Kolom “room\_type” merupakan tipe data str, sehingga perlu mengencodenya menjadi number agar dapat di proses saat clustering data

**Persiapan Pembuatan Program**

* Python 3.7 SDK - <https://www.python.org/downloads/>
* PyCharm JetBrain Community (IDE) - <https://www.jetbrains.com/pycharm/download/download-thanks.html?platform=windows&code=PCC>
* Git (Windows) - <https://git-scm.com/download/win>
* Repository di Github : <https://github.com/amirisback/Mini-ML-Air-BnB>

**Library yang Digunakan untuk Membuat Program**

* numpy
* pandas
* sklearn.preprocessing import LabelEncoder
* sklearn.model\_selection import train\_test\_split
* sklearn.preprocessing import StandardScaler
* sklearn.neighbors import KneighborsClassifier
* sklearn.metrics import classification\_report, confusion\_matrix

**Kajian Pustaka**

* Clustering

Clustering adalah metode penganalisaan data, yang sering dimasukkan sebagai salah satu metode Data Mining, yang tujuannya adalah untuk mengelompokkan data dengan karakteristik yang sama ke suatu ‘wilayah’ yang sama dan data dengan karakteristik yang berbeda ke ‘wilayah’ yang lain.

Ada beberapa pendekatan yang digunakan dalam mengembangkan metode clustering. Dua pendekatan utama adalah clustering dengan pendekatan partisi dan clustering dengan pendekatan hirarki.

Clustering dengan pendekatan partisi atau sering disebut dengan partition-based clustering mengelompokkan data dengan memilah-milah data yang dianalisa ke dalam cluster-cluster yang ada. Clustering dengan pendekatan hirarki atau sering disebut dengan hierarchical clustering mengelompokkan data dengan membuat suatu hirarki berupa dendogram dimana data yang mirip akan ditempatkan pada hirarki yang berdekatan dan yang tidak pada hirarki yang berjauhan. Di samping kedua pendekatan tersebut, ada juga clustering dengan pendekatan automatic mapping (Self-Organising Map/SOM)

* Classification

Classification dapat didefinisikan sebagai proses memprediksi kelas atau kategori dari nilai yang diamati atau titik data yang diberikan. Output yang dikategorikan dapat memiliki bentuk seperti “Black” atau “White” atau “spam” atau “no spam”. Secara matematis, classification adalah tugas mendekati fungsi pemetaan (f) dari variabel input (X) ke variabel output (Y). Ini pada dasarnya milik pembelajaran mesin yang diawasi di mana target juga disediakan bersama dengan set data input.

* Kmeans

Salah satu metode yang banyak digunakan dalam melakukan clustering dengan partisi ini adalah metode k-means. Secara umum metode k-means ini melakukan proses pengelompokan dengan prosedur sebagai berikut:

1. Tentukan jumlah cluster
2. Alokasikan data secara random ke cluster yang ada
3. Hitung rata-rata setiap cluster dari data yang tergabung di dalamnya
4. Alokasikan kembali semua data ke cluster terdekat
5. Ulang proses nomor 3, sampai tidak ada perubahan atau perubahan yang terjadi masih sudah di bawah treshold.

Prosedur dasar ini bisa berubah mengikuti pendekatan pengalokasian data yang diterapkan, apakah crisp atau fuzzy. Setelah meneliti clustering dari sudut yang lain, saya menemukan bahwa k-means clustering mempunyai beberapa kelemahan.

* KNNeighbours

K-nearest neighbors atau knn adalah algoritma yang berfungsi untuk melakukan klasifikasi suatu data berdasarkan data pembelajaran (train data sets), yang diambil dari k tetangga terdekatnya (nearest neighbors). Dengan k merupakan banyaknya tetangga terdekat.

K-nearest neighbors melakukan klasifikasi dengan proyeksi data pembelajaran pada ruang berdimensi banyak. Ruang ini dibagi menjadi bagian-bagian yang merepresentasikan kriteria data pembelajaran. Setiap data pembelajaran direpresentasikan menjadi titik-titik c pada ruang dimensi banyak.

* Klasifikasi Terdekat (Nearest Neighbor Classification)

Data baru yang diklasifikasi selanjutnya diproyeksikan pada ruang dimensi banyak yang telah memuat titik-titik c data pembelajaran. Proses klasifikasi dilakukan dengan mencari titik c terdekat dari c-baru (nearest neighbor). Teknik pencarian tetangga terdekat yang umum dilakukan dengan menggunakan formula jarak euclidean. Berikut beberapa formula yang digunakan dalam algoritma knn.

* Euclidean Distance

Jarak Euclidean adalah formula untuk mencari jarak antara 2 titik dalam ruang dua dimensi.



* Hamming Distance

Jarak Hamming adalah cara mencari jarak antar 2 titik yang dihitung dengan panjang vektor biner yang dibentuk oleh dua titik tersebut dalam block kode biner.

* Manhattan Distance

Manhattan Distance atau Taxicab Geometri adalah formula untuk mencari jarak d antar 2 vektor p,q pada ruang dimensi n.

* Minkowski Distance

Minkowski distance adalah formula pengukuran antar 2 titik pada ruang vektor normal yang merupakan hibridisasi yang mengeneralisasi euclidean distance dan mahattan distance.

* Teknik pencarian tetangga terdekat disesuaikan dengan dimensi data, proyeksi, dan kemudahan implementasi oleh pengguna.

**From Problem To Solution**

* Artikulasikan Masalah Anda dengan Jelas
* Identifikasi Sumber Data Anda
* Identifikasi Potensi Masalah Pembelajaran
* Pikirkan Potensi Bias dan Etik

**Analisis Masalah**

Masalah yang akan di selesaikan pada program kali yaitu penulis akan mengelompokkan level dari setiap ruangan dengan bantuan fitur yang sudah di tetapkan.

**Sumber Data**

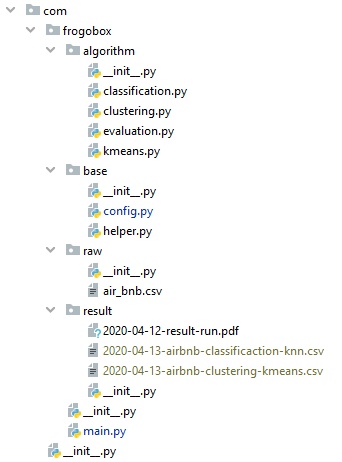
dataset air\_bnb yang berisi 22586 baris dan 16 Kolom dengan rincian : “id,name,host\_id,host\_name,neighbourhood\_group,neighbourhood,latitude,longitude,room\_type,price,minimum\_nights,number\_of\_reviews,last\_review,reviews\_per\_month,calculated\_host\_listings\_count,availability\_365”, penulis memilih 4 fitur yang akan di gunakan dalam proses clustering dan classification. Berikut fitur yang di pilih oleh penulis : ['room\_type', 'price', 'minimum\_nights', 'number\_of\_reviews']

**Potensi Masalah Pembelajaran**

Potensi masalah yang akan muncul dalam proses pembelajaran nanti yaitu ada beberapa data yang kosong, hal tersebut dapat menggangu proses pembelajaran sehingga harus di lakukan eksplorasi dengan cara di delete (drop), mean, median atau mod.

**Program yang Telah Dibuat**

* Stukture File



Disini penulis memisah misah kode program agar mudah di baca dan di pahami, terdapat package com sebagai root\_folder kemudian di isi dengan package frogobox sebagai folder menunjukkan identitas penulis, lalu di ikuti dengan package algoritmh, base, raw, dan result, disini posisi package sejajar. File executeable yang bisa di jalankan berada di frogobox/main.py. semua kode program yang terpisah sudah di integrasikan di dalam file **main.py**.

1. Algorithm

Dalam package ini terdapat semua kode program mengenai algoritma yang telah di buat, dapat di lihat di atas, terdapat file classification.py, clustering.py, evaluation.py, kmeans.py

1. Base

Package ini berisi kumpulan fungsi dan method serta constant data untuk mempermudah pengerjaan kode program

1. Raw

Pada package ini penulis menyimpan source datasetnya yang berisi air\_bnb.cv yang akan di proses nantinya oleh kode program

1. Result

Package ini berisi data hasil eksplorasi dari clustering dan classification dan hasil dari running program yang telah di buat, setiap kali program di jalankan akan menggenarate 2 file dengan nama file yang sudah di beri timestamp tanggal dimana program tersebut dijalankan

* Kode Program
* Classification.py

|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 |  |
| 18 | **import** pandas **as** pd |
| 19 | **from** sklearn.model\_selection **import** train\_test\_split |
| 20 | **from** sklearn.preprocessing **import** StandardScaler |
| 21 | **from** sklearn.neighbors **import** KNeighborsClassifier |
| 22 |  |
| 23 | **from** com.frogobox.algorithm.evaluation **import** evaluation |
| 24 | **from** com.frogobox.base.config **import** \* |
| 25 | **from** com.frogobox.base.helper **import** cast\_to\_int, print\_border\_line |
| 26 |  |
| 27 |  |
| 28 | *# Output hasil di console* |
| 29 | **def** classification\_print(data\_frame, save\_path\_file): |
| 30 | print\_border\_line() |
| 31 | print(**"Result Classification : "** + save\_path\_file) |
| 32 | print\_border\_line() |
| 33 | print(data\_frame) |
| 34 | print() |
| 35 |  |
| 36 |  |
| 37 | *# Generate csv files classification* |
| 38 | **def** classification\_write\_to\_csv(data\_frame, save\_path\_file): |
| 39 | data\_frame.to\_csv(save\_path\_file, index=**False**) |
| 40 |  |
| 41 |  |
| 42 | *#* |
| 43 | **def** classification\_create\_result(column\_label, data\_test, y\_prediction, save\_path\_file): |
| 44 | array\_item\_column = [[] **for** x **in** range(len(column\_label))] |
| 45 |  |
| 46 | **for** column **in** data\_test: |
| 47 | array\_item\_column[0].append(cast\_to\_int(column[0])) |
| 48 | array\_item\_column[1].append(cast\_to\_int(column[1])) |
| 49 | array\_item\_column[2].append(cast\_to\_int(column[2])) |
| 50 | array\_item\_column[3].append(cast\_to\_int(column[3])) |
| 51 |  |
| 52 | classification\_data\_frame = pd.DataFrame({column\_label[0]: array\_item\_column[0], |
| 53 | column\_label[1]: array\_item\_column[1], |
| 54 | column\_label[2]: array\_item\_column[2], |
| 55 | column\_label[3]: array\_item\_column[3], |
| 56 | COLUMN\_CLASS: y\_prediction}) |
| 57 |  |
| 58 | classification\_print(classification\_data\_frame, save\_path\_file) |
| 59 | classification\_write\_to\_csv(classification\_data\_frame, save\_path\_file) |
| 60 |  |
| 61 |  |
| 62 | **def** classification(path\_data\_result\_clustering, column\_label, save\_path\_file): |
| 63 | dataset = pd.read\_csv(path\_data\_result\_clustering) |
| 64 |  |
| 65 | x\_dataset = dataset.iloc[:, :-1].values |
| 66 | y\_dataset = dataset.iloc[:, len(column\_label)].values |
| 67 |  |
| 68 | x\_training, x\_test, y\_training, y\_test = train\_test\_split(x\_dataset, y\_dataset, test\_size=CLASSIFICATION\_SIZE) |
| 69 |  |
| 70 | scaler = StandardScaler() |
| 71 | scaler.fit(x\_training) |
| 72 |  |
| 73 | x\_training = scaler.transform(x\_training) |
| 74 | temp\_test = x\_test |
| 75 | x\_test = scaler.transform(x\_test) |
| 76 |  |
| 77 | classifier = KNeighborsClassifier(n\_neighbors=CLASSIFICATION\_NEIGHBOR) |
| 78 | classifier.fit(x\_training, y\_training) |
| 79 | y\_prediction = classifier.predict(x\_test) |
| 80 |  |
| 81 | classification\_create\_result(column\_label, temp\_test, y\_prediction, save\_path\_file) |
| 82 |  |
| 83 | evaluation(y\_test, y\_prediction) |

* Clustering.py

|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 | **import** pandas **as** pd |
| 18 | **from** sklearn.preprocessing **import** LabelEncoder |
| 19 | **import** matplotlib.pyplot **as** plt |
| 20 | **from** com.frogobox.base.config **import** \* |
| 21 | **from** matplotlib **import** style |
| 22 | **from** com.frogobox.algorithm.kmeans **import** Kmeans |
| 23 | **from** com.frogobox.base.helper **import** cast\_to\_int, print\_border\_line |
| 24 |  |
| 25 | style.use(**'ggplot'**) |
| 26 |  |
| 27 |  |
| 28 | **def** clustering\_print(data\_frame, save\_path\_file): |
| 29 | print\_border\_line() |
| 30 | print(**"Result Clustering : "** + save\_path\_file) |
| 31 | print\_border\_line() |
| 32 | print(data\_frame) |
| 33 | print() |
| 34 |  |
| 35 |  |
| 36 | **def** clustering\_write\_to\_csv(data\_frame, save\_path\_file): |
| 37 | data\_frame.to\_csv(save\_path\_file, index=**False**) |
| 38 |  |
| 39 |  |
| 40 | **def** clustering\_create\_result(column\_label, kmeans, save\_path\_file): |
| 41 | array\_item\_column = [[] **for** x **in** range(len(column\_label))] |
| 42 | result\_label\_cluster = [] |
| 43 |  |
| 44 | **for** classification **in** kmeans.classes: |
| 45 | **for** column **in** kmeans.classes[classification]: |
| 46 | array\_item\_column[0].append(column[0]) |
| 47 | array\_item\_column[1].append(column[1]) |
| 48 | array\_item\_column[2].append(column[2]) |
| 49 | array\_item\_column[3].append(column[3]) |
| 50 | label\_cluster = classification + 1 |
| 51 | result\_label\_cluster.append(COLUMN\_CLASS + **"\_"** + str(label\_cluster)) |
| 52 |  |
| 53 | clustering\_data\_frame = pd.DataFrame({column\_label[0]: array\_item\_column[0], |
| 54 | column\_label[1]: array\_item\_column[1], |
| 55 | column\_label[2]: array\_item\_column[2], |
| 56 | column\_label[3]: array\_item\_column[3], |
| 57 | COLUMN\_CLASS: result\_label\_cluster}) |
| 58 |  |
| 59 | clustering\_print(clustering\_data\_frame, save\_path\_file) |
| 60 | clustering\_write\_to\_csv(clustering\_data\_frame, save\_path\_file) |
| 61 |  |
| 62 |  |
| 63 | **def** create\_image\_result\_cluster(kmeans): |
| 64 | print(**"Drawing Cluster ..."**) |
| 65 |  |
| 66 | **for** centroid **in** kmeans.centroids: |
| 67 | plt.scatter(kmeans.centroids[centroid][0], |
| 68 | kmeans.centroids[centroid][1], |
| 69 | marker=**"o"**, |
| 70 | color=**"k"**, |
| 71 | s=150, |
| 72 | linewidths=5) |
| 73 |  |
| 74 | **for** classification **in** kmeans.classes: |
| 75 | color = COLORS\_CLUSTERING[classification] |
| 76 | **for** featureset **in** kmeans.classes[classification]: |
| 77 | plt.scatter(featureset[0], featureset[1], marker=**"x"**, color=color, s=150, linewidths=5) |
| 78 |  |
| 79 | plt.show() |
| 80 |  |
| 81 |  |
| 82 | **def** clustering(path\_file\_raw\_dataset, column\_label, save\_path\_file): |
| 83 | label\_encoder = LabelEncoder() |
| 84 |  |
| 85 | raw\_data\_set = pd.read\_csv(path\_file\_raw\_dataset) |
| 86 | raw\_data\_set = raw\_data\_set[column\_label] *# ['room\_type', 'price', 'minimum\_nights', 'number\_of\_reviews']* |
| 87 |  |
| 88 | raw\_data\_set[column\_label[0]] = label\_encoder.fit\_transform(raw\_data\_set[column\_label[0]]) |
| 89 |  |
| 90 | x = raw\_data\_set.values |
| 91 | kmeans = Kmeans() |
| 92 | kmeans.fit(x) |
| 93 | clustering\_create\_result(column\_label, kmeans, save\_path\_file) |
| 94 | *# create\_image\_result\_cluster(kmeans)* |
| 95 |  |
| 96 |  |
| 97 |  |

* Evaluation.py

|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 |  |
| 18 | **from** sklearn.metrics **import** classification\_report, confusion\_matrix |
| 19 |  |
| 20 | **from** com.frogobox.base.helper **import** print\_border\_line |
| 21 |  |
| 22 |  |
| 23 | **def** evaluation(y\_test, y\_prediction): |
| 24 | *# evaluasi* |
| 25 | print\_border\_line() |
| 26 | print(**"Result Confusion Matrix : "**) |
| 27 | print\_border\_line() |
| 28 | print(confusion\_matrix(y\_test, y\_prediction)) |
| 29 | print() |
| 30 | print\_border\_line() |
| 31 | print(**"Result Classification Report :"**) |
| 32 | print\_border\_line() |
| 33 | print(classification\_report(y\_test, y\_prediction)) |

* Kmeans.py

|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 |  |
| 18 | **import** numpy **as** np |
| 19 |  |
| 20 | **from** com.frogobox.base.config **import** \* |
| 21 |  |
| 22 | np.seterr(divide=**'ignore'**, invalid=**'ignore'**) |
| 23 |  |
| 24 |  |
| 25 | **class** Kmeans: |
| 26 | **def** \_\_init\_\_(self, k=KMEANS\_K, tolerance=KMEANS\_TOLERANCE, max\_iterations=KMEANS\_MAX\_ITERARIONS): |
| 27 | self.k = k |
| 28 | self.tolerance = tolerance |
| 29 | self.max\_iterations = max\_iterations |
| 30 |  |
| 31 | **def** fit(self, data): |
| 32 |  |
| 33 | self.centroids = {} |
| 34 |  |
| 35 | *# Initiation Centroid* |
| 36 | **for** i **in** range(self.k): |
| 37 | self.centroids[i] = data[i] |
| 38 |  |
| 39 | *# Do Iterations* |
| 40 | **for** i **in** range(self.max\_iterations): |
| 41 | self.classes = {} |
| 42 | **for** i **in** range(self.k): |
| 43 | self.classes[i] = [] |
| 44 |  |
| 45 | *# Find the closest distance to the centroid then select* |
| 46 | **for** features **in** data: |
| 47 | distances = [np.linalg.norm(features - self.centroids[centroid]) **for** centroid **in** self.centroids] |
| 48 | classification = distances.index(min(distances)) |
| 49 | self.classes[classification].append(features) |
| 50 |  |
| 51 | previous = dict(self.centroids) |
| 52 |  |
| 53 | *# Average cluster data points to recalculate centroids* |
| 54 | **for** classification **in** self.classes: |
| 55 | self.centroids[classification] = np.average(self.classes[classification], axis=0) |
| 56 |  |
| 57 | isOptimal = **True** |
| 58 |  |
| 59 | **for** centroid **in** self.centroids: |
| 60 |  |
| 61 | original\_centroid = previous[centroid] |
| 62 | curr = self.centroids[centroid] |
| 63 |  |
| 64 | **if** np.sum((curr - original\_centroid) / original\_centroid \* CENTROID\_PERCENTAGE) > self.tolerance: |
| 65 | isOptimal = **False** |
| 66 |  |
| 67 | *# Stop looping if there is no centroid change* |
| 68 | **if** isOptimal: |
| 69 | **break** |
| 70 |  |
| 71 | **def** prediction(self, data): |
| 72 | distances = [np.linalg.norm(data - self.centroids[centroid]) **for** centroid **in** self.centroids] |
| 73 | classification = distances.index(min(distances)) |
| 74 | **return** classification |

* Config.py

|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 |  |
| 18 | *# Path file data* |
| 19 | **from** datetime **import** datetime |
| 20 |  |
| 21 | *# Constant* |
| 22 | FORMAT\_DATE = **"%Y-%m-%d"** |
| 23 | DATE\_TODAY = str(datetime.today().strftime(FORMAT\_DATE)) |
| 24 | FILE\_NAME\_RAW\_DATA\_SET = **"raw/air\_bnb.csv"** |
| 25 | FILE\_NAME\_RESULT\_CLUSTERING = **"result/"** + DATE\_TODAY + **"-airbnb-clustering-kmeans.csv"** |
| 26 | FILE\_NAME\_RESULT\_CLUSTERING\_RANDOM = **"result/"** + DATE\_TODAY + **"-airbnb-clustering-kmeans-random.csv"** |
| 27 | FILE\_NAME\_RESULT\_CLASSIFICATION = **"result/"** + DATE\_TODAY + **"-airbnb-classificaction-knn.csv"** |
| 28 | FILE\_NAME\_RESULT\_CLASSIFICATION\_RANDOM = **"result/"** + DATE\_TODAY + **"-airbnb-classificaction-knn-random.csv"** |
| 29 | BORDER\_LINE = **"-----------------------------------------------------------------------"** |
| 30 | IDENTITY\_NAME = **"Muhammad Faisal Amir"** |
| 31 | IDENTITY\_NIM = **"1301198497"** |
| 32 | IDENTITY\_CLASS = **"IFX-43-GAB"** |
| 33 | IDENTITY\_MAJORS = **"S1 Informatika 2019"** |
| 34 | IDENTITY\_UNIVERSITY = **"Telkom University"** |
| 35 | IDENTITY\_DATASET = **"AIR BNB"** |
| 36 |  |
| 37 | KMEANS\_K = 3 |
| 38 | KMEANS\_TOLERANCE = 0.001 |
| 39 | KMEANS\_MAX\_ITERARIONS = 300 |
| 40 |  |
| 41 | CENTROID\_PERCENTAGE = 100.0 |
| 42 |  |
| 43 | COLORS\_CLUSTERING = 10 \* [**"g"**, **"r"**, **"c"**, **"b"**, **"k"**] |
| 44 |  |
| 45 | CLASSIFICATION\_SIZE = 0.30 |
| 46 | CLASSIFICATION\_NEIGHBOR = 2 |
| 47 |  |
| 48 | *# Title column table dataset* |
| 49 | COLUMN\_ID = **"id"** *# int* |
| 50 | COLUMN\_NAME = **"name"** *# str* |
| 51 | COLUMN\_HOST\_ID = **"host\_id"** *# int* |
| 52 | COLUMN\_HOST\_NAME = **"host\_name"** *# str* |
| 53 | COLUMN\_NEIGHBOURHOOD\_GROUP = **"neighbourhood\_group"** *# str* |
| 54 | COLUMN\_NEIGHBOURHOOD = **"neighbourhood"** *# str* |
| 55 | COLUMN\_LATITUDE = **"latitude"** *# int* |
| 56 | COLUMN\_LONGITUDE = **"longitude"** *# int* |
| 57 | COLUMN\_ROOM\_TYPE = **"room\_type"** *# str* |
| 58 | COLUMN\_PRICE = **"price"** *# int* |
| 59 | COLUMN\_MINIMUM\_NIGHTS = **"minimum\_nights"** *# int* |
| 60 | COLUMN\_NUMBER\_OF\_REVIEWS = **"number\_of\_reviews"** *# int* |
| 61 | COLUMN\_LAST\_REVIEW = **"last\_review"** *# str* |
| 62 | COLUMN\_REVIEWS\_PER\_MONTH = **"reviews\_per\_month"** *# int* |
| 63 | COLUMN\_CALCULATED\_HOST = **"calculated\_host\_listings\_count"** *# int* |
| 64 | COLUMN\_AVAILABILITY\_365 = **"availability\_365"** *# int* |
| 65 | COLUMN\_CLASS = **"room\_level"** |
| 66 |  |
| 67 | COLUMN\_0 = COLUMN\_ROOM\_TYPE |
| 68 | COLUMN\_1 = COLUMN\_PRICE |
| 69 | COLUMN\_2 = COLUMN\_MINIMUM\_NIGHTS |
| 70 | COLUMN\_3 = COLUMN\_NUMBER\_OF\_REVIEWS |
| 71 |  |
| 72 | DATA\_SET\_FEATURES = [COLUMN\_0, COLUMN\_1, COLUMN\_2, COLUMN\_3] |
| 73 |  |
| 74 | LIST\_INT\_COLUMN = [COLUMN\_ID, |
| 75 | COLUMN\_HOST\_ID, |
| 76 | COLUMN\_LATITUDE, |
| 77 | COLUMN\_LONGITUDE, |
| 78 | COLUMN\_PRICE, |
| 79 | COLUMN\_MINIMUM\_NIGHTS, |
| 80 | COLUMN\_NUMBER\_OF\_REVIEWS, |
| 81 | COLUMN\_REVIEWS\_PER\_MONTH, |
| 82 | COLUMN\_CALCULATED\_HOST, |
| 83 | COLUMN\_AVAILABILITY\_365] |
| 84 | *#* |

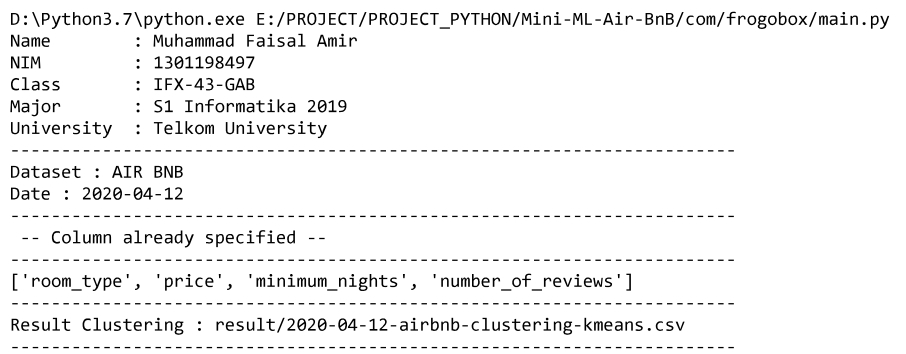
* Helper.py

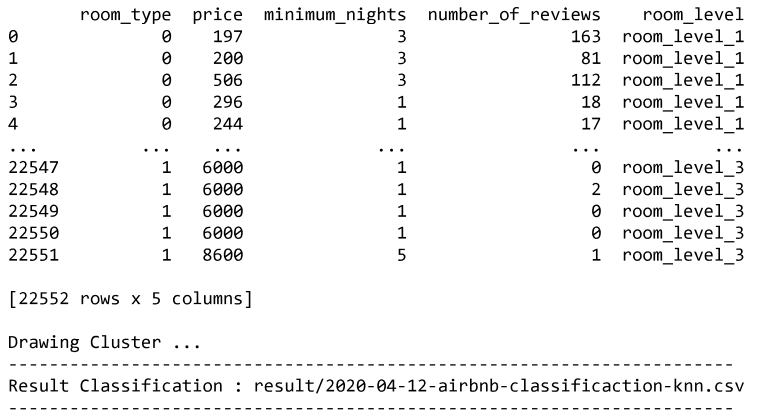
|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 | **import** random |
| 18 |  |
| 19 | **from** com.frogobox.base.config **import** \* |
| 20 |  |
| 21 |  |
| 22 | **def** cast\_to\_int(x): |
| 23 | **if** type(x) **is** str: |
| 24 | **return** len(x) |
| 25 | **elif** type(x) **is** float: |
| 26 | **return** int(x) |
| 27 | **else**: |
| 28 | **return** x |
| 29 |  |
| 30 |  |
| 31 | **def** get\_list\_column\_length(): |
| 32 | **return** len(LIST\_INT\_COLUMN) |
| 33 |  |
| 34 |  |
| 35 | **def** random\_number(): |
| 36 | **return** random.randint(0, get\_list\_column\_length()-1) |
| 37 |  |
| 38 |  |
| 39 | **def** print\_border\_line(): |
| 40 | print(BORDER\_LINE) |

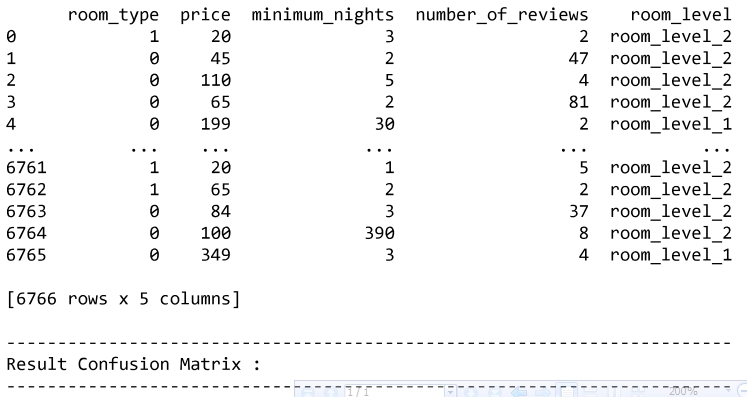
* Main.py

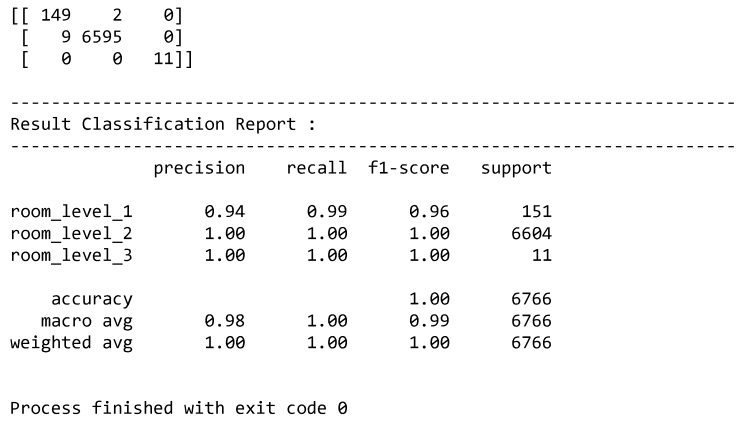
|  |  |
| --- | --- |
| 1 | *#* |
| 2 | *# Created by Faisal Amir* |
| 3 | *# FrogoBox Inc License* |
| 4 | *# -----------------------------------------* |
| 5 | *# Mini-ML-Air-BnB* |
| 6 | *# Copyright (C) 08/04/2020.* |
| 7 | *# All rights reserved* |
| 8 | *# -----------------------------------------* |
| 9 | *# Name : Muhammad Faisal Amir* |
| 10 | *# E-mail : faisalamircs@gmail.com* |
| 11 | *# Github : github.com/amirisback* |
| 12 | *# LinkedIn : linkedin.com/in/faisalamircs* |
| 13 | *# -----------------------------------------* |
| 14 | *# FrogoBox Software Industries* |
| 15 | *#* |
| 16 | *# /* |
| 17 | **import** random |
| 18 | **from** com.frogobox.algorithm.classification **import** classification |
| 19 | **from** com.frogobox.algorithm.clustering **import** clustering |
| 20 | **from** com.frogobox.base.config **import** \* |
| 21 | **from** com.frogobox.base.helper **import** print\_border\_line, get\_list\_column\_length, random\_number |
| 22 |  |
| 23 |  |
| 24 | **def** identity(): |
| 25 | print(**"Name \t\t: "** + IDENTITY\_NAME) |
| 26 | print(**"NIM \t\t: "** + IDENTITY\_NIM) |
| 27 | print(**"Class \t\t: "** + IDENTITY\_CLASS) |
| 28 | print(**"Major \t\t: "** + IDENTITY\_MAJORS) |
| 29 | print(**"University \t: "** + IDENTITY\_UNIVERSITY) |
| 30 | print\_border\_line() |
| 31 | print(**"Dataset : "** + IDENTITY\_DATASET) |
| 32 | print(**"Date : "** + DATE\_TODAY) |
| 33 |  |
| 34 |  |
| 35 | **def** logic(): |
| 36 | print\_border\_line() |
| 37 | print(**" -- Column already specified -- "**) |
| 38 | print\_border\_line() |
| 39 | print(DATA\_SET\_FEATURES) |
| 40 | clustering(FILE\_NAME\_RAW\_DATA\_SET, DATA\_SET\_FEATURES, FILE\_NAME\_RESULT\_CLUSTERING) |
| 41 | classification(FILE\_NAME\_RESULT\_CLUSTERING, DATA\_SET\_FEATURES, FILE\_NAME\_RESULT\_CLASSIFICATION) |
| 42 |  |
| 43 |  |
| 44 | **def** main(): |
| 45 | identity() |
| 46 | logic() |
| 47 |  |
| 48 |  |
| 49 |  |
| 50 | **if** \_\_name\_\_ == **"\_\_main\_\_"**: |
| 51 | main() |
| 52 | *#* |

* Output Program









* Hasil data explorasi



Terdapat 3 clasifikasi yang lolos yaitu room\_level\_1, room\_level\_2, dan room\_level 3

* Evaluasi
* Kesimpulan