LAPORAN TUGAS BESAR MACHINE LEARNING

Ghozy Ghulamul Afif - 1301170379 - IF-41-05
jamessaldo@student.telkomuniversity.ac.id

1. FORMULASI MASALAH

1.1 Artikulasi Masalah

- *Clustering*

  Pada proses *clustering* kali ini, algoritma yang digunakan adalah *k-means*. Sedangkan, *feature* yang akan digunakan untuk model pertama adalah '*minimum_nights*', '*room_type*' dan '*price*' dengan klaster sebanyak 3 (k=3), dan untuk model kedua menggunakan atribut yang sama namun dengan klaster sebanyak 2 (k=2). Proses yang dilakukan pada data tersebut adalah *preprocessing data, clustering,* dan *evaluating* yang akan dijabarkan pada bab berikutnya.
- *Classification*

  Pada proses klasifikasi kali ini, algoritma yang digunakan adalah *Naive Bayes* dan *k-Nearest Neighbors*. Atribut yang digunakan untuk *feature* adalah '*minimum_nights*', '*room_type*' serta '*price*' sedangkan kelas yang digunakan sebagai tujuan dari *feature* tersebut adalah hasil dari *clustering* pada model pertama dan model kedua sebagai *label*. Proses yang dilakukan pada data tersebut adalah *preprocessing data*, *classification,* dan *evaluating* yang akan dijelaskan pada bab berikutnya.

1.2 Sumber Data

Data yang digunakan pada pembelajaran kali ini adalah air_bnb.csv, dimana data tersebut memiliki 22552 record dengan 16 atribut.

1.3 Potensi Masalah

Data yang digunakan memiliki *outlier* dan *missing values* sehingga sebelum melakukan *clustering* dan *classification* harus melalui *preprocessing* terlebih dahulu.

1.4 Potensi Bias dan Etika

Data yang dipakai tidak satupun isinya menyinggung suatu kaum atau kelompok.

2. CLUSTERING

2.1 Preprocessing

2.1.1 Data Cleansing

Pada tahap ini, sebelum data diolah pada tahapan klasterisasi, data dibersihkan terlebih dahulu. Namun sebelum pembersihan data, data di *import* terlebih dahulu sebagai berikut :

- Pengambilan data air_bnb.csv untuk diolah nantinya, dan dimasukkan ke variabel df.

```
# import data air_bnb
df = pd.read_csv('air_bnb.csv')
```

Setelah data berhasil di-*import*, berikut tahapan-tahapan pada *data cleansing* :

- Sebelum data di klasterisasi dengan dua model yang berbeda, beberapa kolom yang memiliki jumlah total label yang begitu besar (terlalu unik) akan dihapus terlebih dahulu karena akan mempengaruhi hasil optimum dari klasterisasi.

```
df = df.drop(['id', 'name', 'host_name', 'host_id', 'latitude', 'longitude', 'last_review'],axis=1)
```

- Setelah kolom yang memiliki banyak label dihapus, dilakukannya *labeling data* untuk kolom yang berisikan *string* menjadi *integer*.

```
# labeling data
encode = pre.LabelEncoder()
df['room_type'] = encode.fit_transform(df['room_type'])
df['neighbourhood_group'] = encode.fit_transform(df['neighbourhood_group'])
df['neighbourhood'] = encode.fit_transform(df['neighbourhood'])
```

- Setelah semua data kolom berisikan *integer*/*float*, dilakukannya *imputation of missing data* untuk mengisi data yang kosong di setiap baris dengan rata-rata (*mean*) dari tiap kolom.

```
# Fillin NaN data with mean datas
df = df.fillna(df.mean())
```

- Pada pembelajaran kali ini, *scaling datas* menggunakan *library* Normalizer.

```
# Normalize datas
norm = Normalizer()
df = pd.DataFrame(norm.fit_transform(df))
df.columns = ['neighbourhood_group', 'neighbourhood', 'room_type', 'price', 'minimum_nights',
              'number_of_reviews', 'reviews_per_month', 'calculated_host_listings_count', 'availability_365']
```

- Kemudian setelah melakukan *scaling-datas*, dilakukannya pendeteksian *outlier* menggunakan *z-score*. Dalam penggunaan *z-score*, nilai yang lebih dari 3 memiliki arti bahwa data tersebut memiliki nilai diluar rata-rata

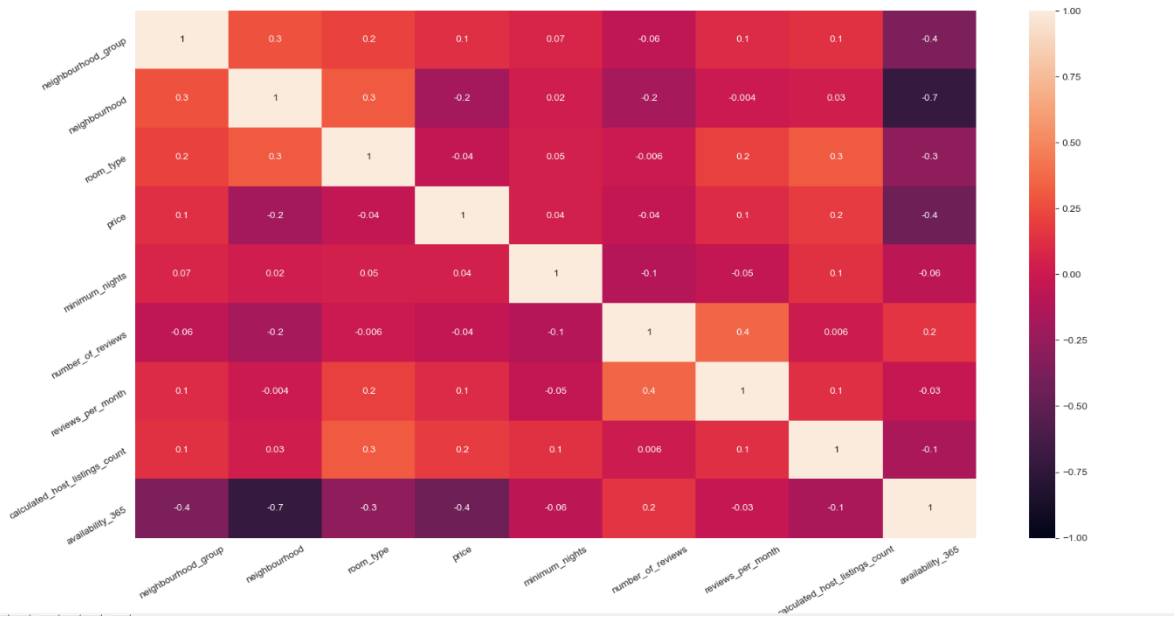
sehingga data tersebut akan dihapus satu baris.

```
# Check the outlier datas
z = np.abs(stats.zscore(df))
df = df[(z < 3).all(axis=1)]
```

2.1.2 Data Exploration

Setelah melakukan *data cleansing* didapatkan total keseluruhan *records* sebanyak 20274 *records* yang semula sebanyak 22552 *records*. Hal tersebut menandakan data sudah bersih dari nilai-nilai yang tidak diperlukan maka dari itu data lebih ringkas dan sudah bisa diolah pada tahapan klasterisasi.

- Setelah berhasil dilakukan *data cleansing*, digunakannya sebuah *library* seaborn untuk menampilkan tabel korelasi dengan menggunakan *heatmap table*.

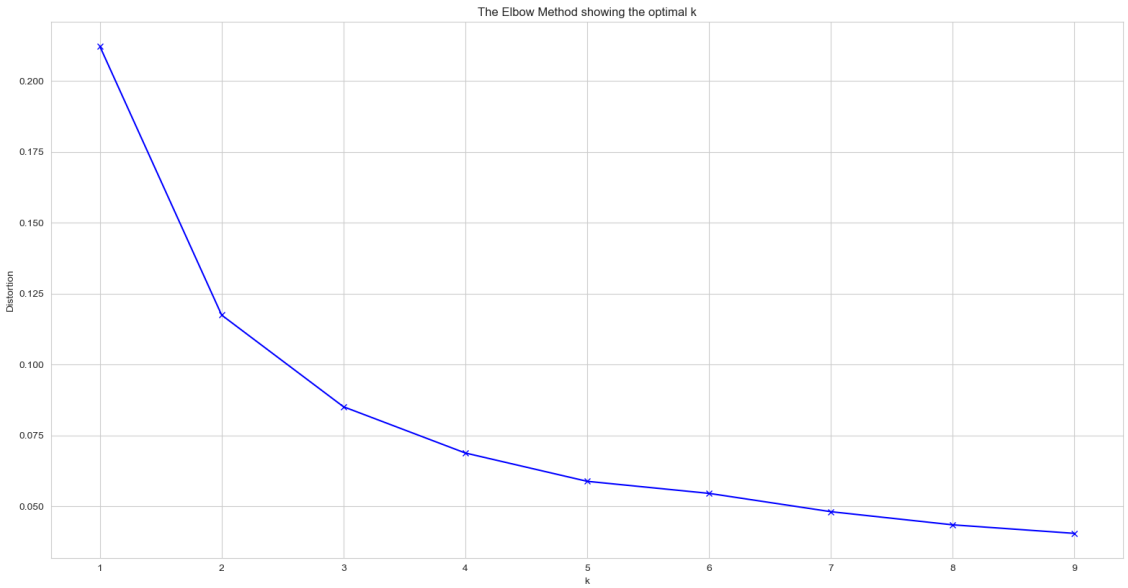


Dapat dilihat bahwa korelasi untuk setiap kolom memiliki nilai bervariasi antara nilai -1 sampai 1. Semakin nilai korelasi mendekati angka satu berarti data kolom tersebut memiliki keidentikan yang lebih tinggi, dan semakin rendah keidentikan data tersebut ketika mendekati angka minus satu. Sehingga semakin identik data kolom tersebut, maka dapat dikatakan data kolom tersebut dapat mewakili satu sama lain.

Dalam pemilihan fitur yang akan digunakan, dipilih tiga fitur yaitu *room_type*, *price*, *minimum_nights* karena memiliki nilai korelasi yang tidak cukup identik datanya sehingga dapat digunakan untuk dilakukan

klasterisasi.

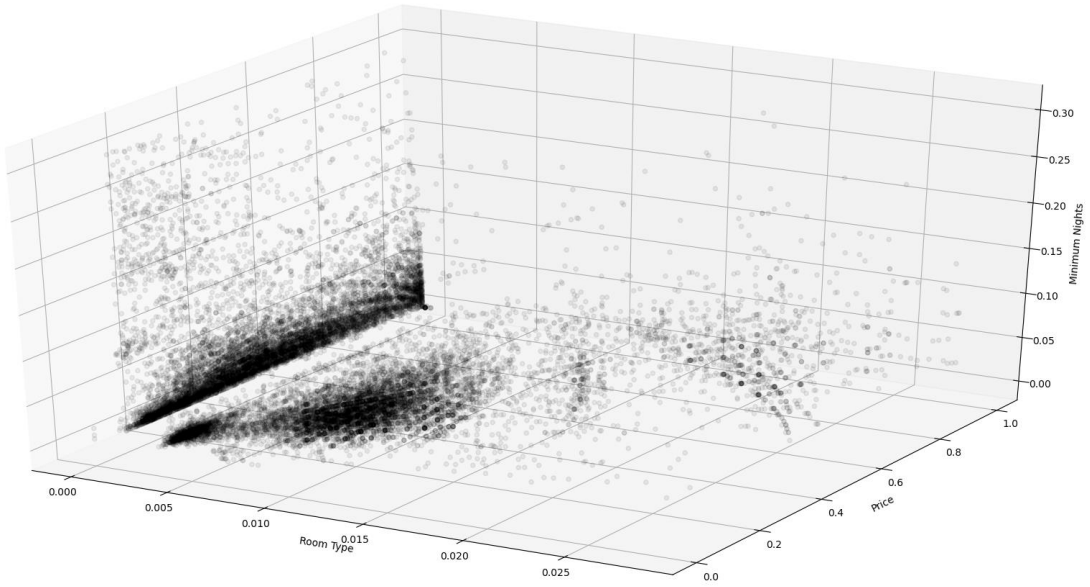
- Dalam memilih banyaknya klaster pada tahapan *k-means*, digunakannya *elbow method* untuk menentukan jumlah klaster. Penentuan jumlah klaster dapat dilihat dengan melihat bentuk yang paling serupa dengan bentuk siku (*elbow*).



Setelah menganalisis hasil diagram *elbow method* diatas, dapat ditentukan bahwa jumlah klaster terbaik ketika klaster berjumlah dua, namun hal tersebut belum bisa dipastikan karena kelemahan dari *elbow method* sendiri adalah terdapatnya sifat ambiguitas sehingga tidak bisa dipastikan hanya dengan melihat grafik kurva saja. Sehingga dalam menentukan jumlah klaster terbaik harus dilakukan tahapan lebih lanjut.

2.2 Data Visualization

Sebelum masuk proses klasterisasi, data divisualisasikan terlebih dahulu untuk melihat bentuk persebaran datanya. Berikut persebaran data dengan sumbu X adalah representasi data *room_type*, sumbu Y adalah representasi data *price*, dan sumbu Z adalah representasi data *minimum_nights* :



2.3 Data Clustering

Pada tahapan klasterisasi, digunakannya algoritma *k-means* dalam melakukan *clustering*. *K-means* adalah salah satu algoritma klasterisasi (*unsupervised*) yang termasuk tertua dan sangat populer di kalangan praktisi karena kemudahan implementasi dan kecepatan prosesnya. Cara kerja metode *k-means* yaitu dengan membagi data kedalam beberapa kelompok, dimana data dalam satu kelompok mempunyai karakteristik yang sama satu dengan yang lainnya dan data akan berbeda dengan yang ada di kelompok lain.

Kelebihan algoritma *k-Nearest Neighbor* :

- Sederhana dalam pengimplementasian.
- Waktu yang dibutuhkan untuk klasterisasi cepat.
- Umum digunakan.

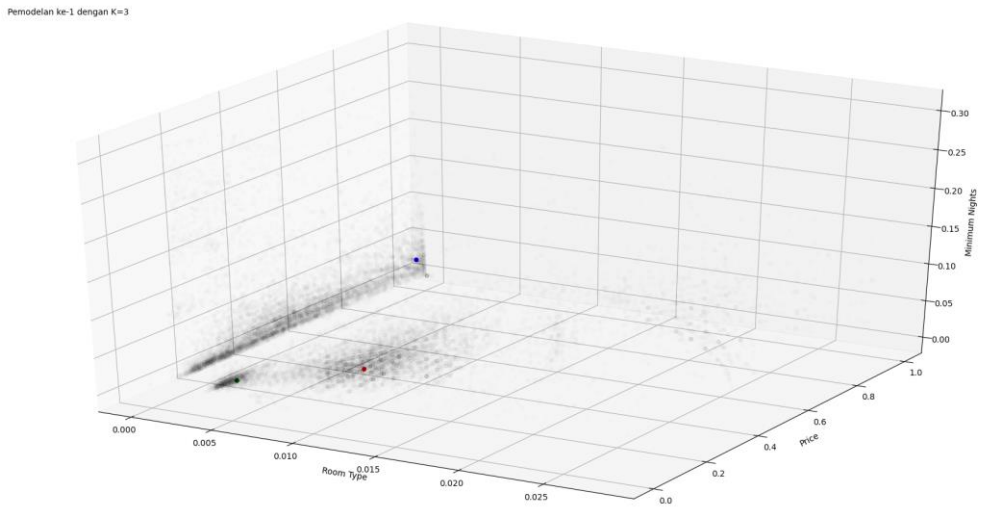
Kekurangan algoritma *k-Nearest Neighbor* :

- Klaster ditentukan oleh titik centroid di awal secara acak, dengan kata lain jika nilai acak kurang baik, maka hasil klasterisasinya pun tidak optimal.
- Dapat terjebak dalam masalah *curse of dimensionality*, yaitu terlalu banyak atribut.

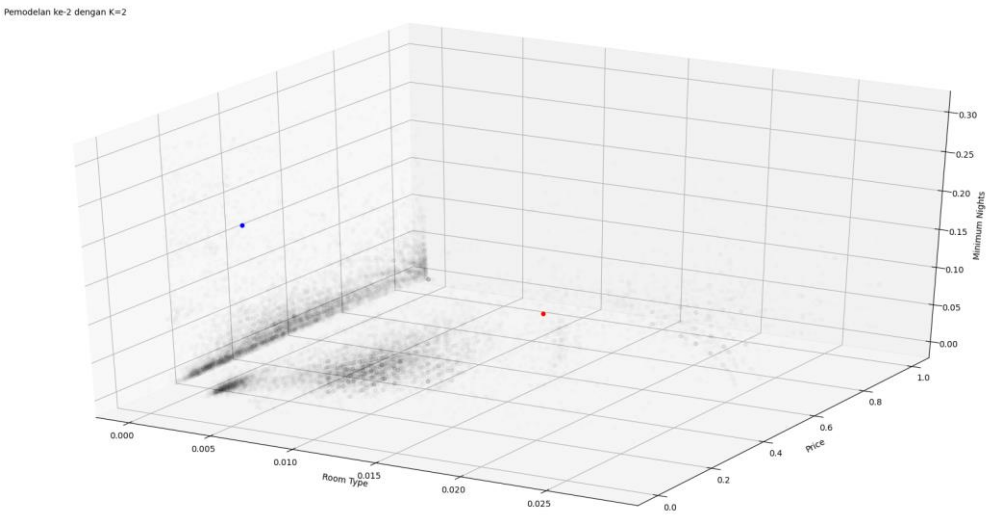
- Jika terdapat banyak titik, maka proses pencarian klaster akan semakin lama.

Berikut tahapan-tahapan yang akan dilalui pada proses *clustering* :

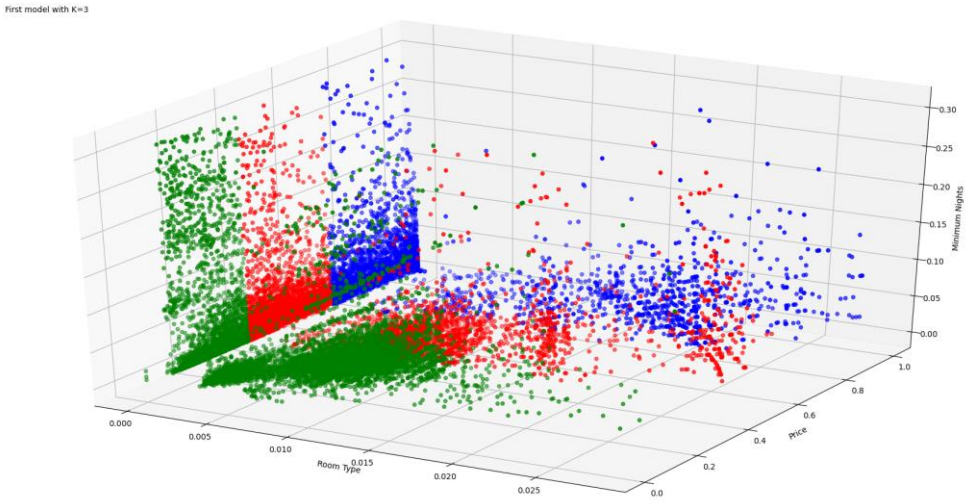
- Karena metode yang digunakan adalah *k-means* maka langkah awal adalah dengan mencari nilai optimum jumlah klaster. Mencari nilai K dapat dilakukan dengan *elbow method* seperti yang telah dilakukan sebelumnya. Pada percobaan klasterisasi ini, jumlah klaster yang digunakan sebanyak tiga untuk model pertama dan dua untuk model kedua, dimana pada tahapan akhir akan dilakukan evaluasi untuk menentukan nilai optimum jumlah klaster.
- Setelah menentukan jumlah klaster yang akan digunakan untuk setiap model, langkah selanjutnya adalah dilakukannya pemilihan titik centroid dengan memilih secara acak dari fitur yang akan digunakan.
- Kemudian titik centroid yang sudah ditentukan akan divisualisasikan ke dalam grafik sesuai dengan warnanya masing-masing. Untuk model pertama penempatan ketiga titik centroid berwarna merah, hijau dan biru dapat terlihat seperti pada gambar berikut.



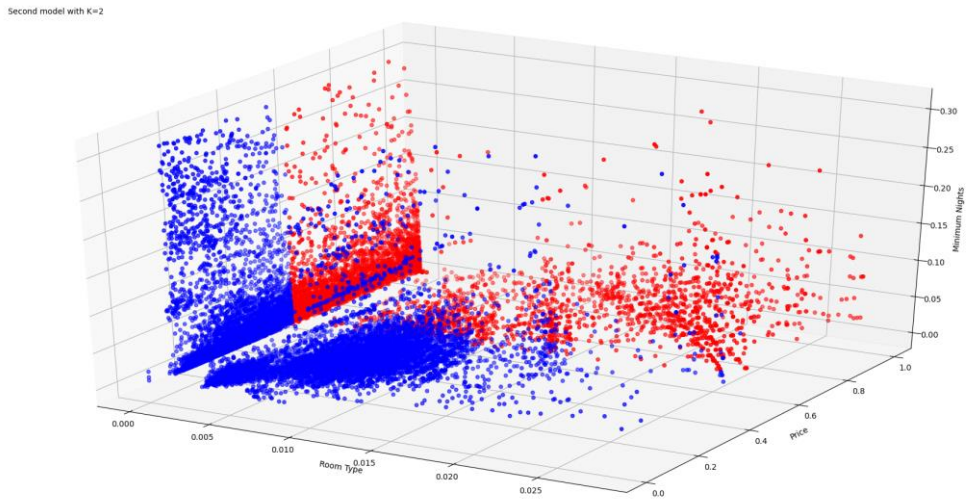
Untuk model kedua, penempatan kedua titik centroid berwarna merah dan biru dapat terlihat seperti pada gambar berikut.



- Setelah menentukan titik centroid pertama dengan pemilihan acak, langkah selanjutnya adalah dengan menentukan centroid terdekat dari titik yang akan di klasterisasi dengan cara menghitung jarak euclidean titik ke tiap centroid.
- Titik centroid yang sudah ditentukan sebelumnya akan berubah pada setiap iterasi seiring dengan perulangan yang dilakukan. Perubahan titik centroid dilakukan dengan cara menghitung rata-rata dari setiap titik di klaster tiap centroid. Perulangan akan berhenti ketika titik centroid sudah tidak berubah lagi. Untuk hasil klasterisasi data pada model pertama dengan tiga titik centroid dapat dilihat pada gambar di bawah ini.



Sedangkan, untuk hasil klasterisasi data pada model kedua dengan dua titik centroid dapat dilihat pada gambar di bawah ini.



2.4 Data Evaluation

Untuk melakukan evaluasi dari hasil klasterisasi di atas, setelah menggunakan *elbow method* pada tahapan sebelumnya untuk mengetahui jumlah klaster yang digunakan, evaluasi yang digunakan pada tahapan ini adalah dengan menghitung nilai SSE (*Sum of Square)* dan membandingkan nilai error mana yang paling kecil dengan menggunakan nilai K masing-masing model. Penggunaan *Sum Square Error* untuk evaluasi *clustering* karena SSE adalah kriteria termudah untuk mengukur *clustering* dengan menggunakan jumlah dari total jarak antara sebaran data dengan centroidnya untuk setiap klaster yang terbentuk. Semakin kecil nilai SSE berarti merupakan bentuk model yang baik. Dapat dilihat bahwa model pertama memiliki nilai error lebih kecil dengan menggunakan jumlah klaster sebanyak tiga (k=3) dibandingkan dengan jumlah klaster sebanyak dua (k=2).

```
Nilai SSE untuk model pertama :  194.9286212290246
Nilai SSE untuk model kedua :  373.6913918823225
```

3. CLASSIFICATION

3.1 Preprocessing

3.1.1 Data Preparation

Pada klasifikasi kali ini, digunakannya fitur yang sama seperti pada proses klasterisasi yaitu fitur *room_type*, *price*, dan *minimum_nights* dan klasifikasi menggunakan dari hasil klasterisasi pada kedua model

sebelumnya sebagai label. Karena proses *data preprocessing* telah dilakukan sebelum melakukan tahapan klasterisasi, maka untuk klasifikasi hanya perlu tinggal menyimpan ketiga fitur ke dalam variabel X serta menyimpan hasil klasterisasi pada kedua model ke dalam array y[]. Ketiga fitur tersebut secara logika saling mempengaruhi satu sama lain, contohnya fitur *price*, tipe ruangan yang akan disewakan pasti mempengaruhi harga sewanya, begitu juga dengan berapa malam penyewaan akan mempengaruhi harga sewanya. Berikut proses pengelompokan data yaitu data X sebagai *feature*-nya, sedangkan data y sebagai target atau kelasnya.

```
# Data preparation
X = df[['room_type', 'price', 'minimum_nights']]
y = [cluster1,cluster2]
```

3.1.2 Feature Engineering

Karena sudah dapat dipastikan semua nilai sudah dinormalisasi dan data sudah siap untuk di klasifikasi, maka akan mulai masuk ke proses *splitting data.*

3.1.3 Splitting Data

Sebelum dimulainya proses pengklasifikasian, data harus dibagi terlebih dahulu menjadi data *train* dan juga data *test.* Dalam melakukan *splitting data* digunakan *library* untuk membagi data tersebut, data yang dibagi pun masing-masing untuk fitur maupun targetnya. Data X dan y[i] dipisah dengan ukuran 80% untuk data latih dan 20% untuk data uji. Dapat dilihat pada kode di bawah bahwa xTrain merupakan data fitur yang akan dijadikan data latih bersama dengan data target dengan variabel yTrain. Sedangkan xTest merupakan data target yang akan mengeluarkan *output* berupa prediksi dari kelasnya yang akan dibandingkan dengan yTest yang merupakan data uji kelas yang sesungguhnya.

```
xTrain, xTest, yTrain, yTest = train_test_split(X, y[i], test_size = 0.2, random_state=100)
```

Setelah dilakukannya *splitting data* kita dapat memasuki proses klasifikasi.

3.2 Classification

Pada pembelajaran kali ini, metode klasifikasi yang digunakan ada dua yaitu *k-Nearest Neighbors* dan *Naive Bayes.* Nantinya kedua metode tersebut akan dibandingkan mana yang paling baik menurut nilai akurasinya masing-masing.

3.2.1 *k-Nearest Neighbors*

Algoritma ***k-Nearest Neighbor*** adalah algoritma *supervised learning* dimana hasil dari instance yang baru diklasifikasikan berdasarkan mayoritas dari kategori **k**-tetangga terdekat.

Tujuan dari algoritma ini adalah untuk mengklasifikasikan objek baru berdasarkan atribut dan *sample-sample* dari *training data*. Algoritma *k-Nearest Neighbor* menggunakan *Neighborhood Classification* sebagai nilai prediksi dari nilai *instance* yang baru.

Kelebihan algoritma *k-Nearest Neighbor* :

- Sederhana dalam pengimplementasian.
- Mampu dalam mengolah data latih yang kotor.
- Efektif untuk data yang memiliki ukuran besar.

Kekurangan algoritma *k-Nearest Neighbor* :

- Perlu pendefinisian nilai k.
- Cukup berat dalam melakukan pengklasifikasian.

3.2.1.1 Prediction Process

Untuk model pertama pada proses klasifikasi menggunakan metode *k-Nearest Neighbor*. Pada metode *k-Nearest Neighbor* hal yang akan dilakukan dijabarkan melalui point-point berikut ini.

- Pertama panggil fungsi dari *k-Nearest Neighbor* itu sendiri.
```
# KNN
neigh = KNeighborsClassifier(n_neighbors=3)
```
- Setelah itu, dilakukan pelatihan pada data latih yang sudah ada.
```
neigh.fit(xTrain, yTrain)
```
- Kemudian karena *data train* sudah selesai dilatih, barulah memprediksi *output* dari data uji yang sudah ada. Prediksi tersebut dimasukkan ke array y_pred.
```
y_pred = neigh.predict(xTest)
```

3.2.1.2 Accuracy

Setelah dapat hasil prediksinya, yang perlu dilakukan adalah menghitung akurasinya dengan cara menghitung akurasi, f1-score, presisi maupun recall.

```
print('NILAI AKURASI K-NEAREST NEIGHBORS')
print('Nilai akurasi model ke-',i+1,' = ', accuracy_score(yTest, y_pred))
print('Nilai F1 model ke-',i+1,' = ', f1_score(yTest, y_pred, average='macro'))
print('Nilai Precision model ke-',i+1,' = ', precision_score(yTest, y_pred, average='macro'))
print('Nilai Recall model ke-',i+1,' = ', recall_score(yTest, y_pred, average='macro'))
```

Dari perhitungan di atas mendapatkan hasil sebagai berikut.

```
NILAI AKURASI K-NEAREST NEIGHBORS
Nilai akurasi model ke- 1  =  0.9977805178791616
Nilai F1 model ke- 1  =  0.997200734365005
Nilai Precision model ke- 1  =  0.997660943872592
Nilai Recall model ke- 1  =  0.9967479276729714
```

```
NILAI AKURASI K-NEAREST NEIGHBORS
Nilai akurasi model ke- 2  =  0.998766954377312
Nilai F1 model ke- 2  =  0.9985780452572253
Nilai Precision model ke- 2  =  0.9988889562588985
Nilai Recall model ke- 2  =  0.9982687805766993
```

3.2.2 *Naive Bayes*

Algoritma Naive Bayes merupakan algoritma yang berdasarkan pada teorema Bayes dengan asumsi independensi antara setiap pasangan fitur. Klasifikasi *Naive Bayes* bekerja dengan baik dalam banyak situasi dunia nyata seperti klasifikasi dokumen dan penyaringan spam.

Kelebihan algoritma *Naive Bayes* :

- Hanya membutuhkan sedikit data latih untuk mendapatkan prediksi yang tepat.
- Mampu melakukan klasifikasi dalam waktu yang sangat cepat.

Kekurangan algoritma *Naive Bayes* :

- Algoritma *Naive Bayes* dikenal sebagai *bad estimator*.

3.2.2.1 Prediction Process

Untuk model pertama pada proses klasifikasi menggunakan metode *naive bayes.* Pada metode *naive bayes* hal yang akan dilakukan dijabarkan melalui point-point berikut ini.

- Pertama panggil fungsi dari *Naive bayes* itu sendiri.

```
# NB
nb = GaussianNB()
```

- Setelah itu, dilakukan pelatihan pada data latih yang sudah ada.

```
nb.fit(xTrain, yTrain)
```

- Kemudian karena *data train* sudah selesai dilatih, barulah memprediksi *output* dari data uji yang sudah ada. Prediksi tersebut dimasukkan ke array y_pred.

```
y_pred = nb.predict(xTest)
```

3.2.2.2 Accuracy

Setelah dapat hasil prediksinya, yang perlu dilakukan adalah menghitung akurasinya dengan cara menghitung akurasi, f1-score, presisi maupun recall.

```
print('NILAI AKURASI NAIVE BAYES')
print('Nilai akurasi model ke-',i+1,' = ', accuracy_score(yTest, y_pred))
print('Nilai F1 model ke-',i+1,' = ', f1_score(yTest, y_pred, average='macro'))
print('Nilai Precision model ke-',i+1,' = ', precision_score(yTest, y_pred, average='macro'))
print('Nilai Recall model ke-',i+1,' = ', recall_score(yTest, y_pred, average='macro'))
```

Dari perhitungan di atas mendapatkan hasil sebagai berikut.

```
NILAI AKURASI NAIVE BAYES
Nilai akurasi model ke- 1  =  0.9815043156596794
Nilai F1 model ke- 1  =  0.9808609598583696
Nilai Precision model ke- 1  =  0.982294946933198
Nilai Recall model ke- 1  =  0.979600898628128
```

```
NILAI AKURASI NAIVE BAYES
Nilai akurasi model ke- 2  =  0.9866831072749692
Nilai F1 model ke- 2  =  0.984626926531073
Nilai Precision model ke- 2  =  0.9854400270493329
Nilai Recall model ke- 2  =  0.9838253641167978
```

3.3 Evaluation

Untuk evaluasi pada klasifikasi kali ini yang dapat disimpulkan oleh penulis adalah dengan menggunakan data yang sama dan proses pengolahan data yang sama, ternyata lebih tinggi nilai akurasi jika klasifikasi menggunakan metode *k-Nearest Neighbor*. Dapat dilihat dari hasil masing-masing nilai akurasi pada kedua metode. Semua hasil nilai baik itu *f1-score*, akurasi, presisi maupun recall metode *k-Nearest Neighbor* memiliki nilai yang lebih tinggi daripada hasil metode *naive bayes.* Oleh karena itu, dapat dievaluasikan bahwa penggunaan metode *k-Nearest Neighbor* lebih baik dibandingkan dengan *naive bayes* karena nilai akurasinya yang tinggi. Kemudian untuk perbandingan antara model pertama dan model kedua, kedua algoritma menunjukkan bahwa model kedua akan mendapatkan nilai akurasi yang lebih tinggi dibandingkan dengan nilai pada model pertama untuk kedua algoritma.

4. CONCLUSION

Untuk proses *clustering*, setelah dilihat nilai evaluasinya yang berupa nilai error dari masing-masing metode ternyata nilai error paling kecil dimiliki oleh model pertama. Dapat disimpulkan bahwa *clustering*

yg paling baik adalah dengan menggunakan fitur *room_type*, *price*, *minimum_nights* dengan jumlah klaster sebanyak tiga. Dibandingkan dengan penggunaan fitur yang sama namun dengan jumlah klaster sebanyak dua, penggunaan klaster sebanyak tiga menghasilkan SSE yang lebih kecil dimana SSE untuk model pertama senilai 195 sedangkan SSE untuk model kedua senilai 374.

Sedangkan, dalam proses klasifikasi didapatkan *k-Nearest Neighbor* memiliki nilai akurasi yang lebih tinggi dibandingkan *Naive Bayes*. Namun meski begitu perbedaan hasil nilai akurasi untuk *k-Nearest Neighbor* dengan *Naive Bayes* tidaklah signifikan, dimana *k-Nearest Neighbor* dengan nilai rata-rata 99% untuk *f1-score*, akurasi, presisi, dan recall. Sedangkan untuk *naive bayes* memiliki nilai rata-rata 98% untuk *f1-score*, akurasi, presisi, dan recall sehingga menjadikan perbedaan diantara kedua algoritma tersebut tidaklah signifikan (hanya berbeda 1%).