

Laporan Tugas Besar Pembelajaran Mesin

IF-42-03



Disusun Oleh :

1301184219	Sya Raihan Heggi
1301184366	Indra Wahyudi

S1 INFORMATIKA
FAKULTAS INFORMATIKA
UNIVERSITAS TELKOM
BANDUNG
2021

DAFTAR ISI

PENDAHULUAN	3
1.1 Latar Belakang Masalah.....	3
1.2 Tujuan	3
METODE	4
2.1 Data Preprocessing.....	4
2.2 Classification Bersalju Besok	13
2.3 Classification Bersalju Hari Ini	19
DAFTAR PUSTAKA	28

PENDAHULUAN

1.1 Latar Belakang Masalah

Saat ini ketersediaan data sangat berlimpah baik yang dihasilkan dari penggunaan teknologi informasi atau pengumpulan data yang berkaitan dengan semua bidang kehidupan, hal ini menimbulkan kebutuhan untuk mendapatkan informasi apa yang dapat didapatkan dari data yang kita proses ini, sehingga nantinya data ini akan dilakukan eksplorasi, analisis, dan ekstraksi informasi itu sendiri dari data, teknik-teknik yang dapat digunakan untuk pengekstrakan pengetahuan dapat menggunakan bantuan *machine learning* yaitu dengan metode Clustering dan Classification, yang dapat digunakan untuk menemukan kategori atau klasifikasi kemungkinan data tersebut.

Klasterisasi adalah metode untuk melakukan pengkategorisasian objek yang ada didalam data ini menjadi beberapa kluster yang memiliki kemiripan atau karakter yang sama, dan kemudian ada Klasifikasi merupakan metode pemrosesan untuk menemukan sebuah model atau fungsi yang menjelaskan dan mencirikan konsep atau kelas data, untuk kepentingan tertentu. Pada tugas besar ini algoritma yang akan digunakan untuk melakukan Clustering adalah K-Means Clustering dan untuk Classification akan membandingkan antara algoritma Naive Bayes dan Decision Tree Classification, dataset yang digunakan pada tugas besar ini adalah dataset data yang berkaitan dengan prediksi apakah akan turun salju besok hari atau tidak dataset sendiri terdiri dari (127277,23) data, yang kemudian akan dibagi menjadi (109095, 24) untuk train set (kelebihan 1 atribut karena ada id yang akan di drop nantinya) dan (18182, 23) untuk tes set. Pemilihan algoritma k means dikarenakan merupakan sebuah metode clustering yang paling sederhana dan umum dan juga karena K-means mempunyai kemampuan mengelompokkan data dalam jumlah yang cukup besar dengan waktu komputasi yang cepat dan efisien. Dan metode Naïve Bayes adalah klasifikasi statistic yang dapat digunakan untuk memprediksi suatu kelas

1.2 Tujuan

Tujuan dari penelitian ini adalah untuk melakukan clustering dan classification terhadap data Salju, dan kemudian melakukan proses pengolahan data dan analisa mengenai performansi dari proses yang sudah dilakukan.

METODE

2.1 Data Preprocessing

Pada tahapan ini akan dilakukan eksplorasi data, pengenalan terhadap data, dan kemudian melakukan pemrosesan terhadap data yaitu **Data Cleaning**, **Data Transformation**, **Data Reduction**, untuk tahapan pertama adalah mengenal dataset yang sudah diberikan dataset salju ini terdiri dari 2 bagian yaitu training dan test sehingga kita dapat menggunakannya untuk klasifikasi dan clustering, untuk klasifikasi kita bisa membagi training data lagi menjadi train dan validation kemudian set test namun diputuskan pada penelitian kali ini pembagian data hanya menjadi training dan test dan untuk melakukan validasi dapat menggunakan **Akurasi**, **Precision**, **Recall**, dan **F1-Score**, sementara untuk clustering hal ini tidak terlalu berpengaruh karena seperti prinsip dan cara kerjanya clustering tidak terkait dengan prediktor namun bisa melabelkan namun untuk kualitasnya nanti bisa kita cek berdasarkan kualitas dari kluster dyang dibuat hal ini bisa dihitung dengan **SSE** atau **Silhouette Coefficient**,

2.1.1 Analisis Informasi Dataset

```
In [3]: df_train.head()
```

Out[3]:

	id	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	...	Kelembaban
0	1	01/06/2014	C4	10.4	15.5	4.8	NaN	NaN	WSW	24.0	...	7
1	2	15/07/2014	C10	9.0	17.0	8.0	2.6	7.4	NaN	NaN	...	8
2	3	16/02/2011	C46	18.2	32.0	0.0	NaN	NaN	ESE	44.0	...	6
3	4	08/08/2012	C36	7.3	24.5	0.0	8.4	10.4	SSW	54.0	...	2
4	5	29/10/2016	C7	5.9	20.3	0.0	3.6	12.6	N	37.0	...	5

5 rows x 24 columns

```
In [4]: df_test.head()
```

Out[4]:

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	ArahAnginSam	...	I
0	04/11/2010	C39	11.0	27.5	0.0	NaN	6.4	WSW	46.0	W	...	
1	26/03/2015	C35	10.0	19.9	0.2	NaN	NaN	WNW	56.0	W	...	
2	22/03/2016	C18	9.2	27.2	0.0	5.2	10.4	SSW	33.0	NE	...	
3	09/12/2011	C31	17.7	27.0	0.0	4.6	6.7	SW	35.0	E	...	
4	20/05/2017	C14	2.3	7.9	88.0	NaN	NaN	NW	46.0	W	...	

5 rows x 23 columns

Pada dataset ini terdapat 23 atribut dan kurang lebih 127277 data yang sudah dibagi menjadi dua dataset yaitu *Training* dan *Test*, dan untuk lebih jelasnya mengenai jumlah data dapat dilihat menggunakan `.shape` seperti pada gambar berikut ini.

```
In [5]: df_train.shape
```

Out[5]: (109095, 24)

```
In [6]: df_test.shape
```

Out[6]: (18182, 23)

Bila diperhatikan lagi ada perbedaan jumlah atribut dari train dan test namun menurut saya ini tidak berpengaruh karena atribut yang berbeda ini hanya id saja dan bisa di drop dan digunakan ke 23 lainnya yang sama dengan atribut pada dataset, membahas mengenai 23 atribut dari analisa didapati 23 atribut yang ada adalah berikut ini.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18182 entries, 0 to 18181
Data columns (total 23 columns):
#   Column                                Non-Null Count  Dtype
---  -
0   Tanggal                                18182 non-null  object
1   KodeLokasi                            18182 non-null  object
2   SuhuMin                                18017 non-null  float64
3   SuhuMax                                18017 non-null  float64
4   Hujan                                  17795 non-null  float64
5   Penguapan                             10326 non-null  float64
6   SinarMatahari                         9464 non-null   float64
7   ArahAnginTerkencang                   16901 non-null  object
8   KecepatanAnginTerkencang              16908 non-null  float64
9   ArahAngin9am                          16874 non-null  object
10  ArahAngin3pm                          17686 non-null  object
11  KecepatanAngin9am                     17984 non-null  float64
12  KecepatanAngin3pm                     17828 non-null  float64
13  Kelembaban9am                         17852 non-null  float64
14  Kelembaban3pm                         17634 non-null  float64
15  Tekanan9am                            16317 non-null  float64
16  Tekanan3pm                            16329 non-null  float64
17  Awan9am                               11140 non-null  float64
18  Awan3pm                               10726 non-null  float64
19  Suhu9am                               17963 non-null  float64
20  Suhu3pm                               17740 non-null  float64
21  BersaljuHariIni                       17795 non-null  object
22  BersaljuBesok                         17763 non-null  object
dtypes: float64(16), object(7)
memory usage: 3.2+ MB
```

Dari atribut yang dimiliki bisa dilihat bahwa ada beberapa atribut yang mempunyai tipe object atau data Kategorik/Nominal hingga nanti perlu ada dilakukan transformasi baik itu menggunakan **Label Encode** maupun **One-Hot-Encode**, kemudian akan di cek apakah data ini memiliki *missing value*, *duplicate value*, dan *wrong values* pertama-tama akan dicek apakah dari dataset ii memiliki missing value dan dimanakah missing value itu berada untuk lebih jelasnya ada pada gambar berikut ini.

<pre>In [9]: df_train.isnull().sum() Out[9]: id 0 Tanggal 0 KodeLokasi 0 SuhuMin 1122 SuhuMax 929 Hujan 2431 Penguapan 47024 SinarMatahari 52379 ArahAnginTerkencang 7744 KecepatanAnginTerkencang 7696 ArahAngin9am 7923 ArahAngin3pm 3197 KecepatanAngin9am 1353 KecepatanAngin3pm 2303 Kelembaban9am 2002 Kelembaban3pm 3374 Tekanan9am 11327 Tekanan3pm 11308 Awan9am 41844 Awan3pm 44471 Suhu9am 1340 Suhu3pm 2698 BersaljuHariIni 2431 BersaljuBesok 2431 dtype: int64</pre>	<pre>In [10]: df_test.isnull().sum() Out[10]: Tanggal 0 KodeLokasi 0 SuhuMin 165 SuhuMax 165 Hujan 387 Penguapan 7856 SinarMatahari 8718 ArahAnginTerkencang 1281 KecepatanAnginTerkencang 1274 ArahAngin9am 1308 ArahAngin3pm 496 KecepatanAngin9am 198 KecepatanAngin3pm 354 Kelembaban9am 330 Kelembaban3pm 548 Tekanan9am 1865 Tekanan3pm 1853 Awan9am 7042 Awan3pm 7456 Suhu9am 219 Suhu3pm 442 BersaljuHariIni 387 BersaljuBesok 419 dtype: int64</pre>
---	---

dilihat dari sebaran missing value ini akan sangat merugikan bagi kita untuk melakukan drop data karena mengurangi jumlah dari dataset yang bisa kita gunakan yang akan berakibat kepada overfitting atau underfitting sehingga diputuskan penanganan dari missing value ini akan dilakukan dengan melakukan pengisian nilai dengan Mean/Modus

dari kolom tersebut, kemudian akan dicek apakah terdapat duplicate value pada dataset yang dapat dilihat dengan gambar berikut ini.

Check Duplicate Value

```
In [11]: df_train.duplicated().sum()
```

```
Out[11]: 0
```

```
In [12]: df_test.duplicated().sum()
```

```
Out[12]: 0
```

kemudian akan dicek apakah ada masukan yang wrong values maksud wrong values disini adalah nilai yang berada pada atribut tidak sesuai seperti contoh berikut ini.

```
df['tumor-size'].value_counts()

30-34    60
25-29    54
20-24    50
15-19    30
Oct-14    28
40-44    22
35-39    19
50-54     8
0-4       8
05-Sep     4
45-49     3
Name: tumor-size, dtype: int64
```

Pada gambar sebelumnya dapat dilihat dari atribut tumor-size seharusnya berisi tentang ukuran dari sebuah tumor namun didalamnya ada nilai yang berisi tanggal hal seperti ini harus ditangani dengan mengubah dengan nilai yang sesuai atau menggantinya dengan nilai mean/modus, namun pada dataset salju yang sudah diberikan bisa dibilang untuk wrong values ini tidak ada sebagai contoh berikut ini.

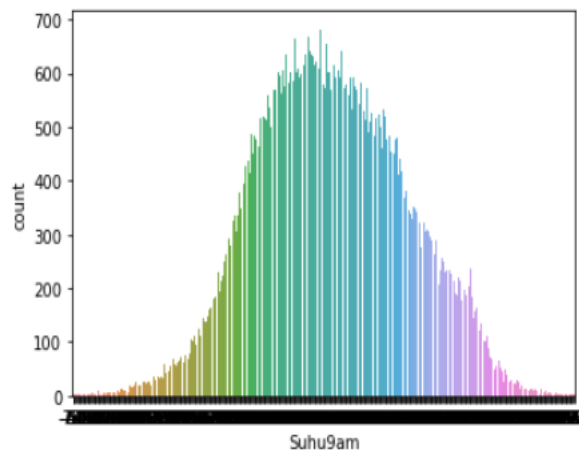
```
In [20]: df_train['ArahAnginTer kencang'].value_counts()
```

```
Out[20]: W      7491
SE      7078
N       6955
S       6931
E       6902
SSE     6882
WSW     6824
SW      6656
SSW     6495
WNW     6202
ENE     6125
NW      6087
ESE     5429
NE      5342
NNW     5025
NNE     4927
Name: ArahAnginTer kencang, dtype: int64
```

bisa dilihat pada data ini nilainya sesuai karena arah angin ini ditangani dengan mata angin dan parameter yang dimasukan merupakan arah mata angin dari segala penjuru sisi, kemudian yang terakhir dilakukan pada eksplorasi data dilihat persebaran dari data yang ada dan juga tingkat skewnessnya apakah data terdistribusi dengan baik atau tidak.

```
In [38]: ▶ sn.countplot(x=df_train['Suhu9am'])
```

```
Out[38]: <matplotlib.axes._subplots.AxesSubplot at 0x24e62d26f48>
```



```
In [58]: ▶ df_train.skew()
```

```
Out[58]: id                0.000000
SuhuMin                0.018027
SuhuMax                0.220794
Hujan                 10.059372
Penguapan              3.916398
SinarMatahari         -0.493987
KecepatanAnginTerkencang 0.875967
KecepatanAngin9am      0.783795
KecepatanAngin3pm      0.622092
Kelembaban9am         -0.481203
Kelembaban3pm          0.033181
Tekanan9am            -0.098483
Tekanan3pm            -0.049309
Awan9am               -0.230423
Awan3pm               -0.227323
Suhu9am                0.085140
Suhu3pm                0.238056
dtype: float64
```

Skewness disini menunjukkan kecenderungan dari persebaran data bisa jadi lebih condong ke satu sisi atau satu sisi sehingga dapat diketahui bagaimana sebaran datanya.

2.1.2 Data Cleansing

Pada tahapan ini kita akan membersihkan data dari eksplorasi yang sudah kita lakukan sebelumnya dalam hal ini menangani **Outliers, Missing Value, Duplicate Value** pertama-tama yang akan ditangani adalah Missing Value seperti yang sudah dijelaskan karena tingkat missing value yang tinggi dapat merugikan bila kita melakukan drop maka akan dilakukan input nilai dengan mean/modus dari kolom tersebut kurang lebihnya untuk data numerik akan diisi dengan nilai Mean dan untuk Object akan diisi dengan modus metode ini akan dilaksanakan kurang lebih seperti berikut ini.

```

In [60]: df_train['Tanggal'].fillna(df_train['Tanggal'].mode().iloc[0], inplace=True)
df_train['KodeLokasi'].fillna(df_train['KodeLokasi'].mode().iloc[0], inplace=True)
df_train['ArahAnginTerkencang'].fillna(df_train['ArahAnginTerkencang'].mode().iloc[0], inplace=True)
df_train['ArahAngin9am'].fillna(df_train['ArahAngin9am'].mode().iloc[0], inplace=True)
df_train['ArahAngin3pm'].fillna(df_train['ArahAngin3pm'].mode().iloc[0], inplace=True)
df_train['BersaljuHariIni'].fillna(df_train['BersaljuHariIni'].mode().iloc[0], inplace=True)
df_train['BersaljuBesok'].fillna(df_train['BersaljuBesok'].mode().iloc[0], inplace=True)

In [61]: df_train['SuhuMin'].fillna(df_train['SuhuMin'].mean(), inplace=True)
df_train['SuhuMax'].fillna(df_train['SuhuMax'].mean(), inplace=True)
df_train['Hujan'].fillna(df_train['Hujan'].mean(), inplace=True)
df_train['Penguapan'].fillna(df_train['Penguapan'].mean(), inplace=True)
df_train['SinarMatahari'].fillna(df_train['SinarMatahari'].mean(), inplace=True)
df_train['KecepatanAnginTerkencang'].fillna(df_train['KecepatanAnginTerkencang'].mean(), inplace=True)
df_train['KecepatanAngin9am'].fillna(df_train['KecepatanAngin9am'].mean(), inplace=True)
df_train['KecepatanAngin3pm'].fillna(df_train['KecepatanAngin3pm'].mean(), inplace=True)
df_train['Tekanan9am'].fillna(df_train['Tekanan9am'].mean(), inplace=True)
df_train['Tekanan3pm'].fillna(df_train['Tekanan3pm'].mean(), inplace=True)
df_train['Kelembaban9am'].fillna(df_train['Kelembaban9am'].mean(), inplace=True)
df_train['Kelembaban3pm'].fillna(df_train['Kelembaban3pm'].mean(), inplace=True)
df_train['Awan9am'].fillna(df_train['Awan9am'].mean(), inplace=True)
df_train['Awan3pm'].fillna(df_train['Awan3pm'].mean(), inplace=True)
df_train['Suhu9am'].fillna(df_train['Suhu9am'].mean(), inplace=True)
df_train['Suhu3pm'].fillna(df_train['Suhu3pm'].mean(), inplace=True)

```

Kemudian dapat dipastikan kembali apakah missing value ini sudah teratasi atau belum dengan mengecek kembali jumlah dari missing valuenya yang dapat dijelaskan kurang lebih seperti berikut ini.

```

In [64]: df_train.isnull().sum()

Out[64]: id                0
Tanggal                0
KodeLokasi             0
SuhuMin                0
SuhuMax                0
Hujan                  0
Penguapan              0
SinarMatahari          0
ArahAnginTerkencang    0
KecepatanAnginTerkencang 0
ArahAngin9am           0
ArahAngin3pm           0
KecepatanAngin9am      0
KecepatanAngin3pm      0
Kelembaban9am          0
Kelembaban3pm          0
Tekanan9am             0
Tekanan3pm             0
Awan9am                0
Awan3pm                0
Suhu9am                0
Suhu3pm                0
BersaljuHariIni        0
BersaljuBesok          0
dtype: int64

```

Kemudian lakukan hal ini juga untuk data train, selanjutnya akan dilakukan drop data yang bernilai duplicate dapat dilakukan dengan berikut ini.

Drop Duplicate Values

```
In [66]: df_train.drop_duplicates(keep=False,inplace=True)  
df_test.drop_duplicates(keep=False,inplace=True)
```

```
In [67]: df_train.duplicated().sum()
```

```
Out[67]: 0
```

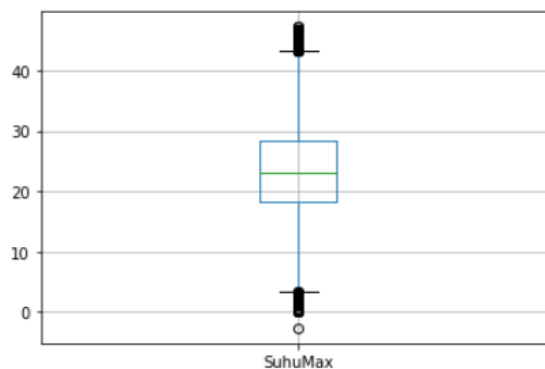
```
In [68]: df_test.duplicated().sum()
```

```
Out[68]: 0
```

Kemudian pastikan kembali bila nilai duplicate sudah tidak ada pada dataset, selanjutnya akan ditangani masalah outlier untuk hal ini bisa menggunakan boxplot untuk melihat apakah ada outlier pada kolom yang dimiliki seperti berikut ini.

```
In [81]: df_train_final.boxplot(column = ['SuhuMax'])
```

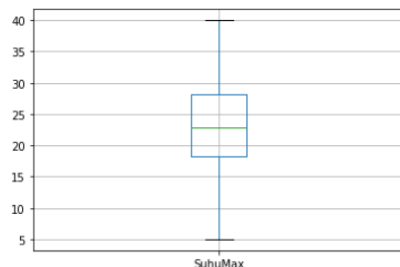
```
Out[81]: <matplotlib.axes._subplots.AxesSubplot at 0x24e75f0ae88>
```



Tangani dengan membuat nilai yang termasuk outlier dan hasilnya kurang lebih akan seperti berikut ini.

```
In [82]: df_train_final.drop(df_train_final[((df_train_final['SuhuMax'] > 40) | (df_train_final['SuhuMax'] < 5))].index,inplace=True)  
df_train_final.boxplot(column = ['SuhuMax'])
```

```
Out[82]: <matplotlib.axes._subplots.AxesSubplot at 0x24e78281308>
```



Dan tahapan cleansing yang dilakukan kurang lebih hal-hal yang sudah dijelaskan sebelumnya.

2.1.3 Data Transformation

Pada tahapan ini kita akan mengubah data yang bukan numerik menjadi numerik berdasarkan pengamatan yang dilakukan data dapat diubah menjadi numerik dengan menggunakan LabelEncoder karena tidak ada yang memiliki hierarki dan beberapa object sudah terpisah sehingga dapat dilakukan kurang lebih berikut ini.

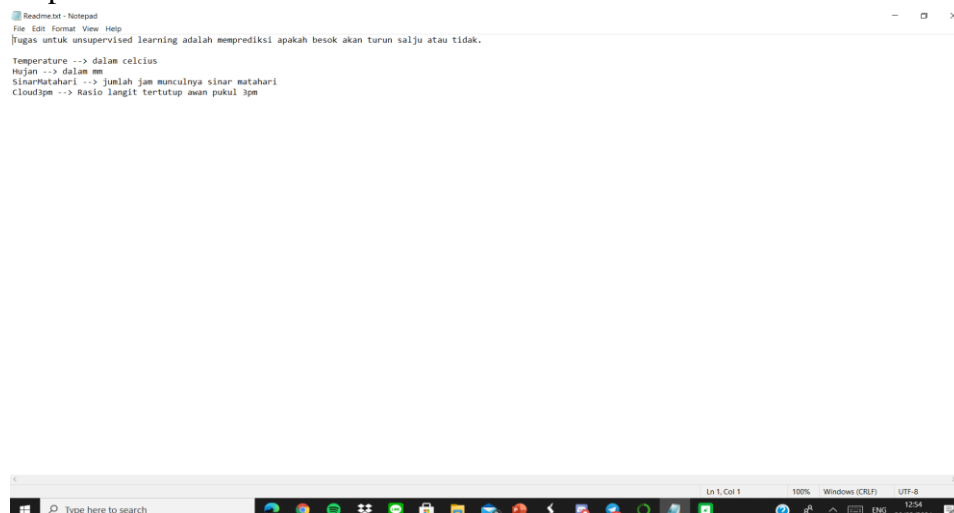
```
In [*]: from sklearn.preprocessing import LabelEncoder
        from sklearn.preprocessing import MinMaxScaler

In [*]: le = LabelEncoder()
        df_train['Tanggal'] = le.fit_transform(df_train['Tanggal'])
        df_train['KodeLokasi'] = le.fit_transform(df_train['KodeLokasi'])
        df_train['ArahAnginTerkencang'] = le.fit_transform(df_train['ArahAnginTerkencang'])
        df_train['ArahAngin9am'] = le.fit_transform(df_train['ArahAngin9am'])
        df_train['ArahAngin3pm'] = le.fit_transform(df_train['ArahAngin3pm'])
        df_train['BersaljuHariIni'] = le.fit_transform(df_train['BersaljuHariIni'])
        df_train['BersaljuBesok'] = le.fit_transform(df_train['BersaljuBesok'])

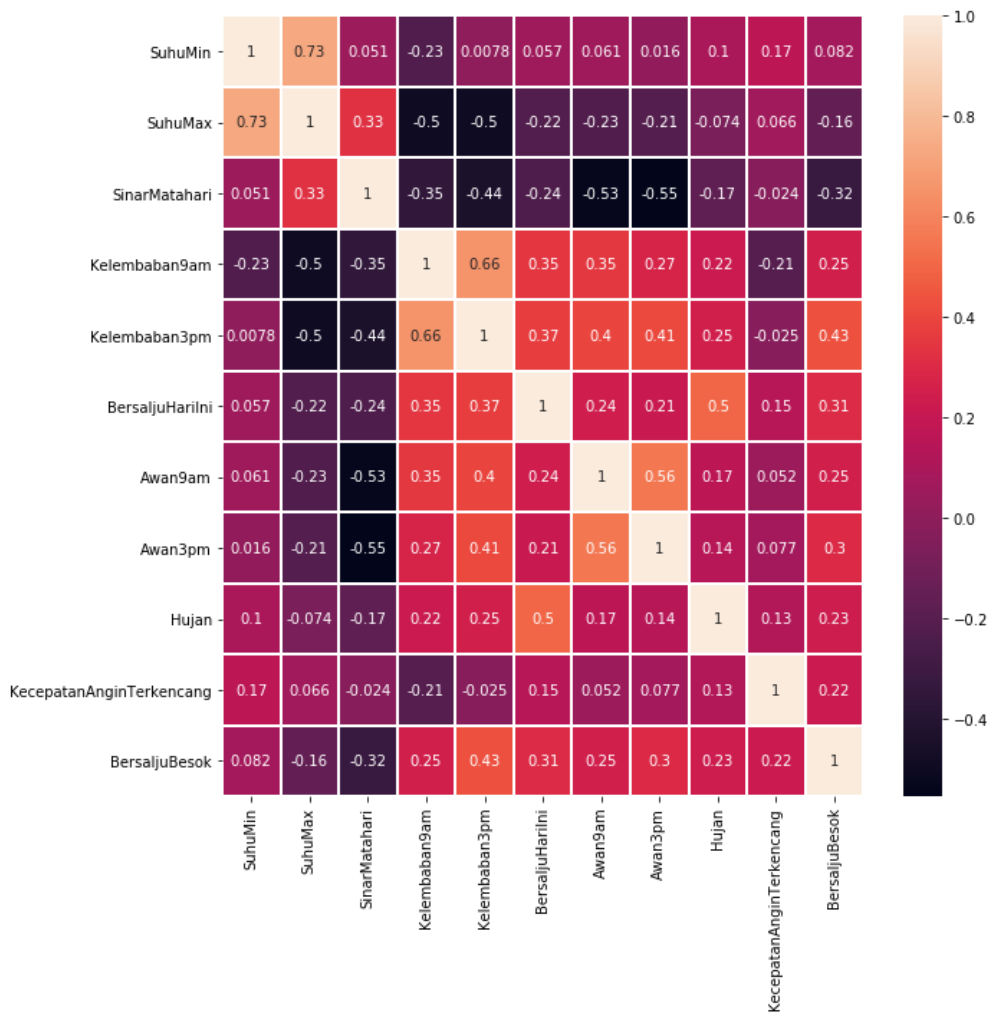
In [*]: df_test['Tanggal'] = le.fit_transform(df_test['Tanggal'])
        df_test['KodeLokasi'] = le.fit_transform(df_test['KodeLokasi'])
        df_test['ArahAnginTerkencang'] = le.fit_transform(df_test['ArahAnginTerkencang'])
        df_test['ArahAngin9am'] = le.fit_transform(df_test['ArahAngin9am'])
        df_test['ArahAngin3pm'] = le.fit_transform(df_test['ArahAngin3pm'])
        df_test['BersaljuHariIni'] = le.fit_transform(df_test['BersaljuHariIni'])
        df_test['BersaljuBesok'] = le.fit_transform(df_test['BersaljuBesok'])
```

2.1.4 Data Reduction

Pada tahapan ini akan dilakukan pemilihan atribut mana saja yang akan digunakan pertimbangan yang dilakukan adalah dengan catatan yang ada pada dataset readme seperti berikut ini.



dan kemudian hasil ini dibandingkan dengan heatmap yang didapati pada dataset yang kurang lebih sudah saya seleksi menjadi seperti berikut ini.



Dan akhirnya diputuskan untuk memilih atau melibatkan Kelembaban3pm dan Awan3pm untuk melakukan clustering klasifikasi nantinya alasan mengapa keduanya terpilih adalah pertama memiliki nilai heatmap yang lumayan tinggi terhadap BersaljuBesok dan juga keduanya memiliki keterkaitan karena waktunya sama dan bila kita merujuk kepada peristiwa yang ada di alam Awan dan Kelembaban merupakan salah satu penyebab terjadinya salju sehingga kedua ini nantinya akan digunakan.

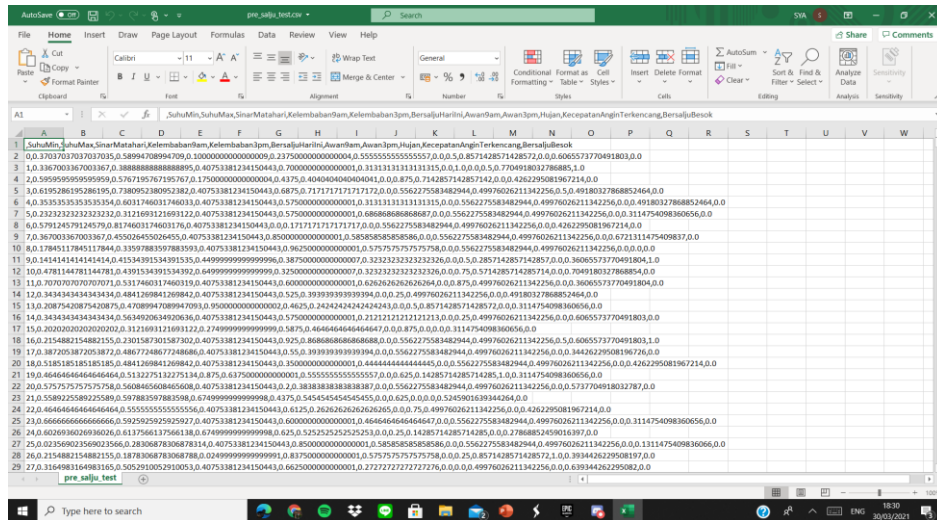
Setelah melewati tahapan tadi data kurang lebih dataset yang akan siap untuk diproses adalah berikut ini.

```
In [92]: databaru.head()
```

```
Out[92]:
```

	Kelembaban3pm	Awan3pm
0	0.355556	0.303228
1	0.777778	0.800000
2	0.622222	0.303228
3	0.400000	0.400000
4	0.355556	0.303228

Tetapi data dengan heatmap yang terpilih pada tahapan pertama akan disimpan karena nanti akan diperlukan untuk penggunaan klasifikasi.



Untuk pembagian data, dataset akan menjadi training set dan test set yang sudah dibagi dari awal sebelumnya sehingga pembagian dataset awal sehingga tidak ada perubahan lebih lanjut untuk split, untuk rasionya sebagai berikut ini Training 85,71462244 % Testing 14,28537756 %.

Namun bila dapat dilakukan penggabungan data dan kemudian melakukan splitting manual serta melakukan klasifikasi menggunakan Decision Tree maka splitting data training dan test dapat dibagi dengan besaran berikut ini.

- **Bersalju Besok**

```

bestAccuracy = 0
bestIndex = 0
i = 0
for x in split_test_size:
    x_train, x_test, y_train, y_test = train_test_split(df_cek_x, df_cek_y, test_size=x, random_state=1)
    classifier = DecisionTreeClassifier(max_depth=6)
    classifier.fit(x_train, y_train)

    y_pred = classifier.predict(x_test)
    accuracy_check = (accuracy_score(y_test, y_pred)*100)
    print(str(split_test_size.index(x))+" Dengan Akurasi : "+str(accuracy_check))
    if accuracy_check > bestAccuracy:
        bestAccuracy = accuracy_check
        bestIndex = split_test_size.index(x)

0 Dengan Akurasi : 99.83822509922048
1 Dengan Akurasi : 99.84593163216176
2 Dengan Akurasi : 99.8266786403773
3 Dengan Akurasi : 99.826673086182

print("Split Size Terbaik Untuk Dataset Adalah\nTrain : "+str((1-split_test_size[bestIndex])*100)+"%\nTest : "+str((split_test_size[bestIndex])*100)+"%\nAkurasi Dihilaskan : "+str(bestAccuracy))

Split Size Terbaik Untuk Dataset Adalah
Train : 60.0%
Test : 40.0%
Akurasi Dihilaskan : 99.84593163216176

```

- **Bersalju Hari Ini**

[illegible]

2.2 Classification Bersalju Besok

Klasifikasi adalah pemrosesan untuk menemukan sebuah model atau fungsi yang menjelaskan dan mencirikan konsep atau kelas data, untuk kepentingan tertentu. Tujuan ‘classification’ adalah untuk menganalisa data historis yang disimpan dalam database dan secara otomatis menghasilkan suatu model yang bisa memprediksi perilaku di masa mendatang. Pada penelitian ini akan menggunakan 2 algoritma classification yaitu Decision Tree Classification (ID3) dan Naïve Bayes. Alasannya menggunakan 2 algoritma tersebut adalah ingin **membandingkan nilai validasi** dari kedua algoritma tersebut, kemudian pada penelitian ini juga akan mempertimbangkan apakah **pengaruh scalling terhadap kedua algoritma tersebut**, dan terakhir adalah berapa parameter yang optimum untuk kedua algoritma tersebut misalnya pada Decision Tree berapakah max_depthnya, kemudian untuk melakukan validasi akan digunakan perhitungan Akurasi, Precision, Recall, dan F1-Score, penggunaan akurasi merupakan validasi secara keseluruhan data benar terhadap nilai keseluruhan data, kemudian menggunakan Precision dan Recall untuk mengetahui bahwa tingkat benar itu terhadap data yang hanya benar dan hanya salah berapa sehingga hasilnya lebih merinci, dan terakhir di gunakan F1-Score untuk membandingkan dengan hasil akurasi karena pada F1-Score terdapat perpaduan antara nilai *True Positive* dan *True Negative* yang berada pada lingkup data.

2.2.1 Pemilihan Atribut dan Alasannya

Alasan pemilihan atribut kurang lebih masih sama seperti yang digunakan pada tahapan clustering, sebenarnya data yang digunakan juga sudah disiapkan dari tahapan tersebut namun yang membedakan pada tahapan klasifikasi ini tidak melakukan pemilihan kembali dari *best correlation value* di heatmap sehingga atribut yang diambil adalah.

```
In [33]: df_train_x.head()
```

Out[33]:

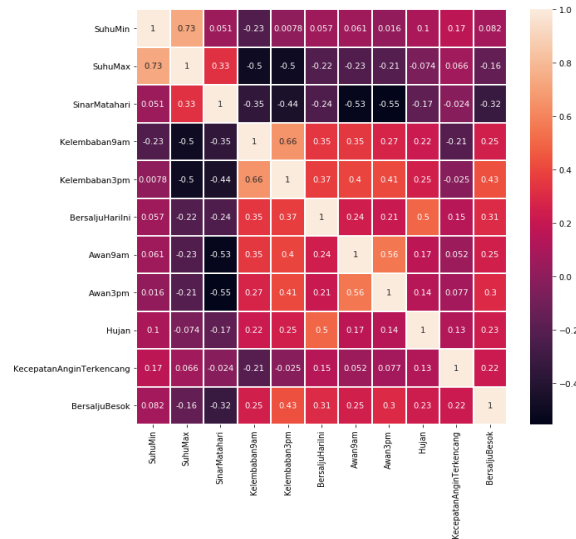
	SuhuMin	Kelembaban9am	Kelembaban3pm	BersaljuHarini	Awan9am	Awan3pm	Hujan	KecepatanAngin	Terkencang
0	0.612795	0.457143	0.355556	0.0	0.494544	0.303228	0.0		0.66000
1	0.414141	0.657143	0.777778	0.0	0.888889	0.800000	0.0		0.58064
2	0.218855	0.500000	0.622222	0.0	0.494544	0.303228	0.0		0.56000
3	0.313131	0.742857	0.400000	0.0	0.555556	0.400000	0.0		0.56000
4	0.360269	0.614286	0.355556	0.0	0.494544	0.303228	0.1		0.64000

```
In [34]: df_train_y.head()
```

Out[34]:

	BersaljuBesok
0	0.0
1	1.0
2	0.0
3	0.0
4	0.0

Untuk lebih jelasnya mengapa bisa terpilih atribut tersebut dapat dilihat hasil dari heatmap yang ada dari dataset yang dimiliki.



Dari Heatmap tersebut ada 2 nilai yang dirasa kurang optimum karena bernilai negative yaitu SinarMatahari dan SuhuMax, dikarenakan hal tersebut maka diputuskan untuk tidak mengikutsertakan atribut tersebut.

2.2.2 Decision Tree Classification

Decision Tree Clasification adalah sebuah model klasifikasi dimana akan dibuatkan sebuah tree yang berisi keputusan dan nantinya akan terbuat sebuah rule yang menentukan hasilnya tahapan yang dilakukan adalah pertama mempersiapkan data baik itu yang Sudah di scaling maupun tidak, kemudian siapkan classifier.

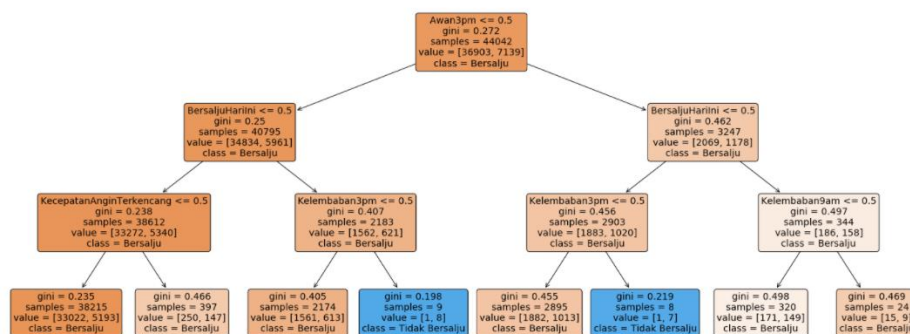
```
In [35]: classifier = DecisionTreeClassifier(max_depth=3)
classifier.fit(df_train_x.astype(int),df_train_y.astype(int))

Out[35]: DecisionTreeClassifier(max_depth=3)
```

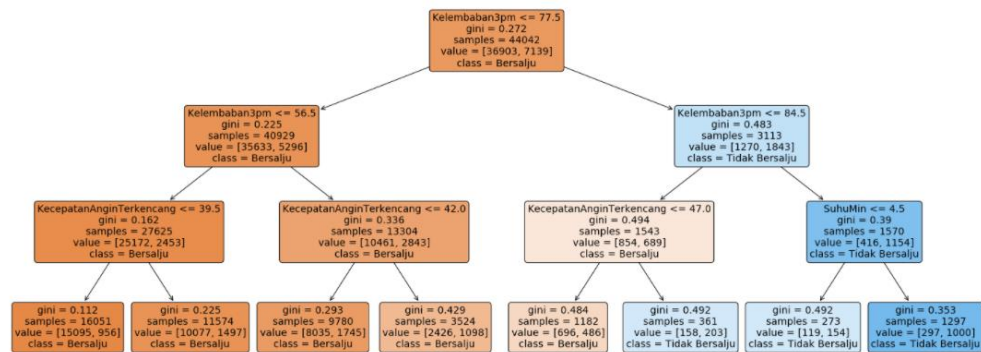
Menggunakan Decision Tree yang Depthnya Maksimum 3 Karena Hasil dari Validasinya Optimum dengan kedalaman Maksimum 3, karena bila dinaikan atau di turunkan nilai validasi F1-Score, Recall, Precision, dan Akurasi Mengalami penurunan

Pada penelitian ini dilakukan beberapa kali percobaan dan ditemukan untuk dataset ini memiliki nilai optimum bila max_depth = 3 oleh karena itu kita gunakan untuk membangun data tersebut, selanjutnya kita visualisasikan tree yang dibangun dan hasilnya seperti bagaimana.

• Tree Data Di Scaling



- **Tree Data Tidak Di Scaling**



Selanjutnya adalah dilakukan prediksi data dengan train dan test set, pada penelitian ini tidak dilakukan split karena data yang digunakan sudah disiapkan sebelumnya sebagai test dan train oleh karena itu bisa langsung digunakan.

- **Data Di Scaling**

Hasil yang didapatkan untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

Validasi Training Set

```
In [541]: M model_acc = accuracy_score(df_train_y.astype(int),y_train_pred)
           print(model_acc*100)
80.03042550292902

In [542]: M cm = confusion_matrix(db_train_y, y_train_pred)
           print(cm)
[[36806  4709]
 [ 1500  1027]]

In [543]: M print(classification_report(df_train_y.astype(int),y_train_pred))
```

	precision	recall	f1-score	support
0	0.87	0.90	0.88	36903
1	0.36	0.29	0.32	7139
accuracy			0.80	44042
macro avg	0.61	0.59	0.60	44042
weighted avg	0.78	0.80	0.79	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapatkan adalah berikut ini.

Validasi Test Set

```
M acc = accuracy_score(df_test_y.astype(int), y_pred)
print(acc*100)
81.19766556711494

M cm = confusion_matrix(df_test_y, y_pred)
print(cm)
[[6075  711]
 [ 771  325]]

M print(classification_report(df_test_y.astype(int), y_pred))
```

	precision	recall	f1-score	support
0	0.89	0.90	0.89	6786
1	0.31	0.30	0.30	1096
accuracy			0.81	7882
macro avg	0.60	0.60	0.60	7882
weighted avg	0.81	0.81	0.81	7882

Hasil yang didapatkan adalah pada Training Set didapatkan akurasi 80 % dan pada Test Set didapatkan 81% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

- **Data Tidak Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

Validasi Training Set

```

> model_acc = accuracy_score(df_train_y.astype(int),y_train_pred)
> print(model_acc*100)

85.56832114799509

> cm = confusion_matrix(df_train_y, y_train_pred)
> print(cm)

[[36329  574]
 [ 5782 1357]]

> print(classification_report(df_train_y.astype(int),y_train_pred))

```

	precision	recall	f1-score	support
0	0.86	0.98	0.92	36903
1	0.70	0.19	0.30	7139
accuracy			0.86	44042
macro avg	0.78	0.59	0.61	44042
weighted avg	0.84	0.86	0.82	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

```

> acc = accuracy_score(df_test_y.astype(int), y_pred)
> print(acc*100)

87.09718345597564

> cm = confusion_matrix(df_test_y, y_pred)
> print(cm)

[[6690  96]
 [ 921 175]]

> print(classification_report(df_test_y,y_pred))

```

	precision	recall	f1-score	support
0	0.88	0.99	0.93	6786
1	0.65	0.16	0.26	1096
accuracy			0.87	7882
macro avg	0.76	0.57	0.59	7882
weighted avg	0.85	0.87	0.84	7882

Hasil yang didapati adalah pada Training Set didapati akurasi 85 % dan pada Test Set didapati 87% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

2.2.3 Gaussian Naïve Bayes Classification

Naïve Bayes adalah salah satu metode klasifikasi dimana data akan dihitung dan dipertimbangkan berdasarkan probabilitasnya, dengan perhitungan dan aturan Naïve Bayes dan selain itu metode ini bisa dibilang metode yang sederhana dan hasilnya cukup baik, pada penelitian kali ini digunakan versi perhitungan menggunakan Gaussian.

Untuk mempermudah proses pengerjaan maka diperlukan perubahan dulu dari bentuk dataframe menjadi array (**ini opsional karena tidak dirubah pun bisa di proses**) oleh karena itu dilakukan perintah berikut ini.

Ubah data menjadi array untuk dapat diproses library dengan Gaussian Naive Bayes

```
df_train_x = df_train_x.to_numpy()
df_train_y = df_train_y.to_numpy()
df_test_x = df_test_x.to_numpy()
df_test_y = df_test_y.to_numpy()
```

Selanjutnya adalah inisialisasi classifier yang dibangun dari Gaussian Naïve Bayes untuk classifier ini digunakan library yang sudah disediakan oleh sklearn dengan perintah seperti berikut ini.

Inisialisasi Classifier dan Latih Model

```
classifier = GaussianNB()
classifier.fit(df_train_x, df_train_y)
```

Selanjutnya kita buat model dan hasilkan prediksi baik untuk dataset training dan dataset test yang dihasilkan seperti berikut ini.

- **Data Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

Validasi Training Set

```
y_train_pred = classifier.predict(df_train_x)

model_acc = accuracy_score(df_train_y, y_train_pred)
print(model_acc*100)

81.59484128786158

cm = confusion_matrix(df_train_y, y_train_pred)
print(cm)

[[33839  3064]
 [ 5042  2097]]

print(classification_report(df_train_y, y_train_pred))
```

	precision	recall	f1-score	support
0.0	0.87	0.92	0.89	36903
1.0	0.41	0.29	0.34	7139
accuracy			0.82	44042
macro avg	0.64	0.61	0.62	44042
weighted avg	0.80	0.82	0.80	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

Validasi Test Set

```
➤ y_pred = classifier.predict(df_test_x)

➤ acc = accuracy_score(df_test_y, y_pred)
  print(acc*100)
83.63359553412839

➤ cm = confusion_matrix(df_test_y, y_pred)
  print(cm)
[[6257  529]
 [ 761  335]]

➤ print(classification_report(df_test_y, y_pred))
```

	precision	recall	f1-score	support
0.0	0.89	0.92	0.91	6786
1.0	0.39	0.31	0.34	1096
accuracy			0.84	7882
macro avg	0.64	0.61	0.62	7882
weighted avg	0.82	0.84	0.83	7882

Hasil yang didapati adalah pada Training Set didapati akurasi 82 % dan pada Test Set didapati 84% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

- **Data Tidak Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

Validasi Training Set

```
➤ y_train_pred = classifier.predict(df_train_x)

➤ model_acc = accuracy_score(df_train_y,y_train_pred)
  print(model_acc*100)
81.59484128786158

➤ cm = confusion_matrix(df_train_y, y_train_pred)
  print(cm)
[[33839  3064]
 [ 5042  2097]]

➤ print(classification_report(df_train_y,y_train_pred))
```

	precision	recall	f1-score	support
0	0.87	0.92	0.89	36903
1	0.41	0.29	0.34	7139
accuracy			0.82	44042
macro avg	0.64	0.61	0.62	44042
weighted avg	0.80	0.82	0.80	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

Validasi Test Set

```
y_pred = classifier.predict(df_test_x)

acc = accuracy_score(df_test_y, y_pred)
print(acc*100)

86.1837097183456

cm = confusion_matrix(df_test_y, y_pred)
print(cm)

[[6592  194]
 [ 895 201]]

print(classification_report(df_test_y, y_pred))
```

	precision	recall	f1-score	support
0	0.88	0.97	0.92	6786
1	0.51	0.18	0.27	1096
accuracy			0.86	7882
macro avg	0.69	0.58	0.60	7882
weighted avg	0.83	0.86	0.83	7882

Hasil yang didapati adalah pada Training Set didapati akurasi 82 % dan pada Test Set didapati 86% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

2.2.3 Kesimpulan

Dari percobaan yang dilakukan dapat disimpulkan 2 algoritma ini tidak terlalu berpengaruh bila dilakukan scalling kenapa karena hasilnya malah menunjukan pada dataset ini jika dilakukan scalling mengalami penurunan nilai validasi yang dilakukan, untuk algoritma yang digunakan untuk saat ini Decision Tree menghasilkan hasil klasifikasi lebih baik bila data tidak di scalling namun bila data di scalling Naïve Bayes Gaussian menjadi lebih baik (**Hal ini dapat terjadi karena pada Decision Tree dikatakan tidak terlalu berpengaruh terhadap proses scalling/normalisasi**), dan nilai yang dihasilkan pun masih sama berada di rentang 80-90% akurasi, sehingga bisa dikatakan algoritma ini keduanya baik dan dapat disesuaikan dengan konteks dari dataset, selanjutnya parameter optimum untuk Decision Tree sendiri adalah membuat tree dengan max_depth = 3 karena bila kurang atau lebih mengalami penurunan nilai yang signifikan.

2.3 Classification Bersalju Hari Ini

Klasifikasi adalah pemrosesan untuk menemukan sebuah model atau fungsi yang menjelaskan dan mencirikan konsep atau kelas data, untuk kepentingan tertentu. Tujuan 'classification' adalah untuk menganalisa data historis yang disimpan dalam database dan secara otomatis menghasilkan suatu model yang bisa memprediksi perilaku di masa mendatang. Pada penelitian ini akan menggunakan 2 algoritma classification yaitu Decision Tree Classification (ID3) dan Naïve Bayes. Alasannya menggunakan 2

algoritma tersebut adalah ingin **membandingkan nilai validasi** dari kedua algoritma tersebut, kemudian pada penelitian ini juga akan mempertimbangkan apakah **pengaruh scalling terhadap kedua algoritma tersebut**, dan terakhir adalah berapa parameter yang optimum untuk kedua algoritma tersebut misalnya pada Decision Tree berapakah max_depthnya, kemudian untuk melakukan validasi akan digunakan perhitungan Akurasi, Precision, Recall, dan F1-Score, penggunaan akurasi merupakan validasi secara keseluruhan data benar terhadap nilai keseluruhan data, kemudian menggunakan Precision dan Recall untuk mengetahui bahwa tingkat benar itu terhadap data yang hanya benar dan hanya salah berapa sehingga hasilnya lebih merinci, dan terakhir di gunakan F1-Score untuk membandingkan dengan hasil akurasi karena pada F1-Score terdapat perpaduan antara nilai *True Positive* dan *True Negative* yang berada pada lingkup data.

2.3.1 Pemilihan Atribut dan Alasannya

Alasan pemilihan atribut kurang lebih masih sama seperti yang digunakan pada tahapan clustering, sebenarnya data yang digunakan juga sudah disiapkan dari tahapan tersebut namun yang membedakan pada tahapan klasifikasi ini tidak melakukan pemilihan kembali dari *best correlation value* di heatmap sehingga atribut yang diambil adalah.

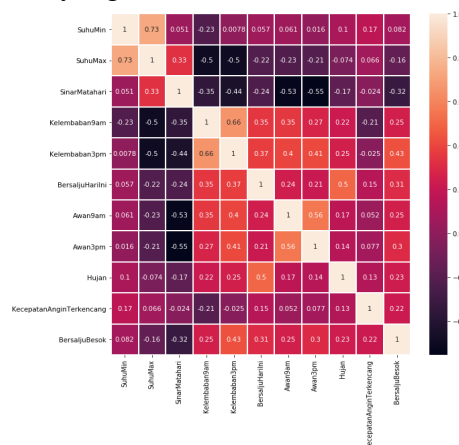
```
In [590]: db_train_x.head()
Out[590]:
```

	Kelembaban9am	Kelembaban3pm	Awan9am	Awan3pm	Hujan
0	0.457143	0.355556	0.494544	0.303228	0.0
1	0.657143	0.777778	0.888889	0.800000	0.0
2	0.500000	0.622222	0.494544	0.303228	0.0
3	0.742857	0.400000	0.555556	0.400000	0.0
4	0.614286	0.355556	0.494544	0.303228	0.1

```
In [591]: db_train_y.head()
Out[591]:
```

	BersaljuHariIni
0	0.0
1	0.0
2	0.0
3	0.0
4	0.0

Untuk lebih jelasnya mengapa bisa terpilih atribut tersebut dapat dilihat hasil dari heatmap yang ada dari dataset yang dimiliki.



Dari Heatmap tersebut ada yang akan digunakan antara lain hujan, kelembaban, dan suhu karena memiliki nilai korelasi yang cukup baik sehingga dapat digunakan karena memiliki keterkaitan dengan klasifikasi salju hari ini, sehingga atribut-atribut ini lah yang cocok untuk digunakan dalam melakukan proses klasifikasi, dan kembali lagi pemilihan

ini juga didasari dengan membaca faktor-faktor yang menyebabkan terjadinya salju *Now forms when the atmospheric temperature is at or below freezing (0 degrees Celsius or 32 degrees Fahrenheit) and there is a minimum amount of moisture in the air* [8] berdasarkan kutipan pada sumber 8 diketahui salju terbentuk karena pengaruh kelembaban dan suhu yang berada di atmosfer menyebabkan hal ini juga mendukung untuk memilih atribut tersebut dan kemudian tingkat hujan sendiri memiliki korelasi atau hubungan dengan kejadian salju karena dengan terjadinya hujan maka akan terjadi kelembaban dan penurunan suhu oleh karena itu bisa digunakan dalam penelitian kali ini.

2.3.2 Decision Tree Classification

Decision Tree Clasification adalah sebuah model klasifikasi dimana akan dibuatkan sebuah tree yang berisi keputusan dan nantinya akan terbuat sebuah rule yang menentukan hasilnya tahapan yang dilakukan adalah pertama mempersiapkan data baik itu yang Sudah di scalling maupun tidak, kemudian siapkan classifier.

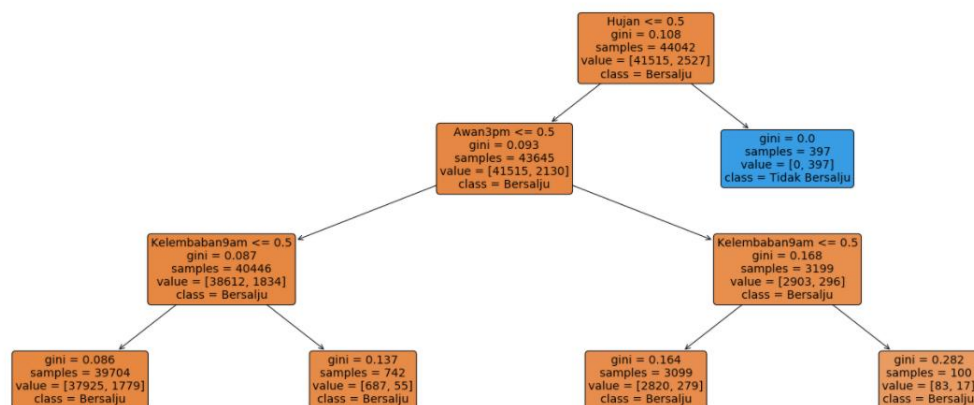
```
In [35]: > classifier = DecisionTreeClassifier(max_depth=3)
> classifier.fit(df_train_x.astype(int),df_train_y.astype(int))

Out[35]: DecisionTreeClassifier(max_depth=3)
```

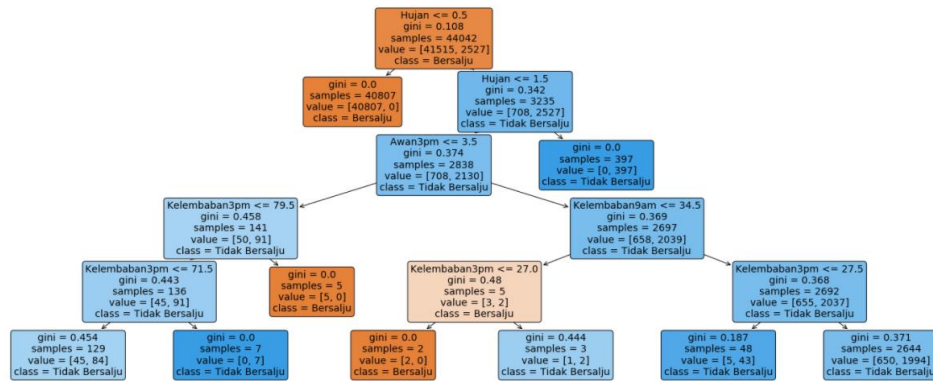
Menggunakan Decision Tree yang Depthnya Maksimum 3 Karena Hasil dari Validasinya Optimum dengan kedalaman Maksimum 3, karena bila dinaikan atau di turunkan nilai validasi F1-Score, Recall, Precision, dan Akurasi Mengalami penurunan

Pada penelitian ini dilakukan beberapa kali percobaan dan ditemukan untuk dataset ini memiliki nilai optimum dimulai dari max_depth = 3 namun untuk Data tidak discalling dapat menggunakan max_depth = 5 oleh karena itu kita gunakan untuk membangun data tersebut, selanjutnya kita visualisasikan tree yang dibangun dan hasilnya seperti bagaimana.

- **Tree Data Di Scalling**



- **Tree Data Tidak Di Scalling**




Selanjutnya adalah dilakukan prediksi data dengan train dan test set, pada penelitian ini tidak dilakukan split karena data yang digunakan sudah disiapkan sebelumnya sebagai test dan train oleh karena itu bisa langsung digunakan.


- **Data Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.


```
Validasi Training Set
```

```
[596] ▶  ML
      model_acc = accuracy_score(db_train_y, y_train_pred)
      print(model_acc*100)

      100.0
```

```
[597] ▶  ML
      cm = confusion_matrix(db_train_y, y_train_pred)
      print(cm)

      [[41515   0]
       [   0 2527]]
```

```
[598] ▶  ML
      print(classification_report(db_train_y, y_train_pred))
```

	precision	recall	f1-score	support
	0.0	1.00	1.00	41515
	1.0	1.00	1.00	2527
accuracy			1.00	44042
macro avg	1.00	1.00	1.00	44042
weighted avg	1.00	1.00	1.00	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

```
Validasi Test Set
```

```
[599] ▶ MI

acc = accuracy_score(db_test_y, y_pred)
print(acc*100)

96.2572951027658
```

```
[600] ▶ MI

cm = confusion_matrix(db_test_y, y_pred)
print(cm)

[[7587 295]
 [ 0  0]]
```

```
[601] ▶ MI

print(classification_report(db_test_y, y_pred))
```

	precision	recall	f1-score	support	
	0.0	1.00	0.96	0.98	7882
	1.0	0.00	0.00	0.00	0
accuracy				0.96	7882
macro avg	0.50	0.48	0.49		7882
weighted avg	1.00	0.96	0.98		7882

Hasil yang didapati adalah pada Training Set didapati akurasi 100 % dan pada Test Set didapati 96% (**Terjadi Overfitting namun tidak terlalu tinggi hanya terpaut 4%**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan terjadi overfitting namun nilai yang dihasilkan tidak terlalu jauh berbeda sehingga bisa dikatakan modelnya tidak terlalu buruk.

- **Data Tidak Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

```
Validasi Training Set

[619] > MI
model_acc = accuracy_score(db_train_y, y_train_pred)
print(model_acc*100)

94.17374324508424

[620] > MI
cm = confusion_matrix(db_train_y, y_train_pred)
print(cm)

[[38949  2566]
 [    0  2527]]

[621] > MI
print(classification_report(db_train_y, y_train_pred))
```

	precision	recall	f1-score	support
0	1.00	0.94	0.97	41515
1	0.50	1.00	0.66	2527
accuracy			0.94	44042
macro avg	0.75	0.97	0.82	44042
weighted avg	0.97	0.94	0.95	44042

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

```
Validasi Test Set

[622] > MI
acc = accuracy_score(db_test_y, y_pred)
print(acc*100)

100.0

[623] > MI
cm = confusion_matrix(db_test_y, y_pred)
print(cm)

[[7882]]

[624] > MI
print(classification_report(db_test_y, y_pred))
```

	precision	recall	f1-score	support
0	1.00	1.00	1.00	7882
accuracy			1.00	7882
macro avg	1.00	1.00	1.00	7882
weighted avg	1.00	1.00	1.00	7882

Hasil yang didapati adalah pada Training Set didapati akurasi 94 % dan pada Test Set didapati 100% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter

validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

2.3.3 Gaussian Naïve Bayes Classification

Naïve Bayes adalah salah satu metode klasifikasi dimana data akan dihitung dan dipertimbangkan berdasarkan probabilitasnya, dengan perhitungan dan aturan Naïve Bayes dan selain itu metode ini bisa dibilang metode yang sederhana dan hasilnya cukup baik, pada penelitian kali ini digunakan versi perhitungannya menggunakan Gaussian, selanjutnya adalah inisialisasi classifier yang dibangun dari Gaussian Naïve Bayes untuk classifier ini digunakan library yang sudah disediakan oleh sklearn dengan perintah seperti berikut ini.

Inisialisasi Classifier dan Latih Model

```
classifier = GaussianNB()  
classifier.fit(df_train_x, df_train_y)
```

Selanjutnya kita buat model dan hasilkan prediksi baik untuk dataset training dan dataset test yang dihasilkan seperti berikut ini.

- **Data Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

```
Validasi Training Set  
[609] ▶ MI  
model_acc = accuracy_score(db_train_y, y_train_pred)  
print(model_acc*100)  
97.59774760455929  
[610] ▶ MI  
cm = confusion_matrix(db_train_y, y_train_pred)  
print(cm)  
[[40457 1058]  
 [    0 2527]]  
[611] ▶ MI  
print(classification_report(db_train_y, y_train_pred))  
  
              precision    recall  f1-score   support  
  
    0.0         1.00      0.97      0.99       41515  
    1.0         0.70      1.00      0.83        2527  
  
 accuracy          0.85          0.99          0.98       44042  
  macro avg          0.85          0.99          0.91       44042  
  weighted avg          0.98          0.98          0.98       44042
```

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.


```

Validasi Test Set

[612] > ML
      acc = accuracy_score(db_test_y, y_pred)
      print(acc*100)

87.2113676731794

[613] > ML
      cm = confusion_matrix(db_test_y, y_pred)
      print(cm)

[[6874 1008]
 [   0    0]]

[614] > ML
      print(classification_report(db_test_y, y_pred))

              precision    recall  f1-score   support

         0.0         1.00         0.87         0.93        7882
         1.0         0.00         0.00         0.00         0

 accuracy
 macro avg         0.50         0.44         0.47        7882
 weighted avg         1.00         0.87         0.93        7882

```

Hasil yang didapati adalah pada Training Set didapati akurasi 97 % dan pada Test Set didapati 87% (**Terjadi Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score dari hasil yang didapatkan terjadi overfitting pada data ini kemungkinan terjadi karena atribut ada yang terganggu oleh proses scalling, dan juga kemungkinan karena atribut yang dipilih sensitive terhadap scalling.

- **Data Tidak Di Scalling**

Hasil yang didapati untuk model yang dibangun dan diuji Kembali dengan nilai yang ada di training set adalah berikut ini.

```

Validasi Training Set

[628] > ML
      model_acc = accuracy_score(db_train_y, y_train_pred)
      print(model_acc*100)

97.59774760455929

[629] > ML
      cm = confusion_matrix(db_train_y, y_train_pred)
      print(cm)

[[40457 1058]
 [   0 2527]]

[630] > ML
      print(classification_report(db_train_y, y_train_pred))

              precision    recall  f1-score   support

         0         1.00         0.97         0.99        41515
         1         0.70         1.00         0.83         2527

 accuracy
 macro avg         0.85         0.99         0.91        44042
 weighted avg         0.98         0.98         0.98        44042

```

Kemudian dilakukan test ke dataset test dan hasil yang didapati adalah berikut ini.

```

Validasi Test Set

[631] > MI
      acc = accuracy_score(db_test_y, y_pred)
      print(acc*100)

100.0

[632] > MI
      cm = confusion_matrix(db_test_y, y_pred)
      print(cm)

[[7882]]

[633] > MI
      print(classification_report(db_test_y, y_pred))

              precision    recall  f1-score   support

      0               1.00        1.00        1.00        7882

 accuracy               1.00        1.00        1.00        7882
 macro avg              1.00        1.00        1.00        7882
 weighted avg           1.00        1.00        1.00        7882

```

Hasil yang didapati adalah pada Training Set didapati akurasi 97 % dan pada Test Set didapati 100% (**Tidak Overfitting**) dilengkapi dengan beberapa parameter validasi lainnya precision, recall, dan F1-Score sehingga bisa dikatakan tidak terjadi overfitting atau underfitting karena hasilnya tidak berbeda terlalu jauh.

2.3.4 Kesimpulan

Dari percobaan yang dilakukan dapat disimpulkan 2 algoritma ini tidak terlalu berpengaruh bila dilakukan scalling kenapa karena hasilnya malah menunjukan pada dataset ini jika dilakukan scalling mengalami penurunan nilai validasi yang dilakukan, untuk algoritma yang digunakan untuk saat ini Decision Tree menghasilkan hasil klasifikasi lebih baik bila data tidak di scalling namun bila data di scalling Naïve Bayes Gaussian menjadi lebih baik (**Hal ini dapat terjadi karena pada Decision Tree dikatakan tidak terlalu berpengaruh terhadap proses scalling/normalisasi**), dan nilai yang dihasilkan pun masih sama berada di rentang 80-90% akurasinya, sehingga bisa dikatakan algoritma ini keduanya baik dan dapat disesuaikan dengan konteks dari dataset , selanjutnya parameter optimum untuk Decision Tree sendiri adalah membuat tree dengan `max_depth = 3` dan `max_depth = 5` karena bila kurang atau lebih mengalami penurunan nilai yang signifikan, namun pada percobaan klasifikasi BersaljuHariIni terjadi beberapa Overfitting dari model yang dibangun namun pada Decision Tree tidak terlalu signifikan karena hanya sekitar 4% perbedaannya, namun dengan data tidak di scalling pada Naïve Bayes terdapat perbedaan 10% Overfitting yang bisa dikatakan cukup tinggi, hal ini dapat disebabkan karena atribut yang digunakan terpengaruhi oleh proses scalling tersebut.

2.4 Saran

Untuk penelitian atau percobaan berikutnya bisa membandingkan lebih lagi mengenai optimalisasi dari algoritmanya bagaimana karena pada percobaan ini Cuma beberapa saja yang diuji, dan kemudian bisa dibandingkan lagi dengan metode klasifikasi yang lebih

kompleks, dan terakhir adalah mungkin dapat dibandingkan dengan metode Decision Tree atau Naïve Bayes yang lain karena kedua algoritma tersebut memiliki jenis-jenis yang berbeda-beda, dan mungkin untuk membedakan mengenai data scalling dan non-scalling bisa lebih lengkap lagi parameter yang digunakan seperti Analisis dari jenis data yang dikandungnya yang tidak dijelaskan pada laporan ini.

DAFTAR PUSTAKA

- [1] [An easy guide to choose the right Machine Learning algorithm](#), diakses 11/05/2021
- [2] [Do you know how to choose the right machine learning algorithm among 7 different types?](#) , diakses 11/05/2021
- [3] [Introduction to Data Mining](#)
- [4] [How to Choose the Right Machine Learning Algorithm for Your Application](#), diakses 11/05/2021
- [5] Jadhav, S. D., & Channe, H. P. (2016). Comparative study of K-NN, naive Bayes and decision tree classification techniques. *International Journal of Science and Research (IJSR)*, 5(1), 1842-1845.
- [6] Ashari, A., Paryudi, I., & Tjoa, A. M. (2013). Performance comparison between Naïve Bayes, decision tree and k-nearest neighbor in searching alternative design in an energy simulation tool. *International Journal of Advanced Computer Science and Applications (IJACSA)*, 4(11).
- [7] Pranckevičius, T., & Marcinkevičius, V. (2017). Comparison of naive bayes, random forest, decision tree, support vector machines, and logistic regression classifiers for text reviews classification. *Baltic Journal of Modern Computing*, 5(2), 221.
- [8] [How Snow Form](#), diakses 11/05/2021.
- [9] McCabe, G. J., Clark, M. P., & Hay, L. E. (2007). Rain-on-snow events in the western United States. *Bulletin of the American Meteorological Society*, 88(3), 319-328..