

Nama = Akhmad Dian Muzaki

NIM = 1941720205

Kelas = TI-2G

Link = <https://github.com/muzakidian/ML-Deteksi-Salju>

UAS Machine Learning Deteksi Salju

Classification

2021

1. READ DATA

□ Read data salju_train.csv

```
[2]: df_train = pd.read_csv('salju_train.csv')
df_train.head()
```

	Id	Tanggal	Ketinggian	SuhuMin	SuhuMax	Kejadian	Pengukuran	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	ArahAnginRata	ArahAnginRata	KecepatanAnginRata	KecepatanAnginRata	KetinggianRata	KetinggianRata
0	1	01/08/2016	04	15.4	15.5	4.0	NaH	NaH	WVV	24.0	NaH	WVV	0.0	13.0	78.0	
1	2	19/07/2016	010	8.0	17.5	8.0	3.0	7.4	NaH	NaH	SW	WVV	12.0	20.0	80.0	
2	3	16/02/2017	048	18.2	22.0	8.0	NaH	NaH	SE	44.0	SE	SE	19.0	29.0	81.0	
3	4	08/08/2016	038	7.5	24.5	0.0	8.4	10.4	SW	54.0	N	SW	12.0	19.0	30.0	
4	5	20/10/2016	07	9.0	20.1	8.0	3.0	12.0	N	37.0	WVV	SE	22.0	19.0	69.0	

□ Read data salju_test.csv

```
[6]: df_test = pd.read_csv('salju_test.csv')
df_test.head()
```

	Tanggal	Ketinggian	SuhuMin	SuhuMax	Kejadian	Pengukuran	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	ArahAnginRata	ArahAnginRata	KecepatanAnginRata	KecepatanAnginRata	KetinggianRata	KetinggianRata
0	04/11/2016	039	11.0	27.0	0.0	NaH	0.4	WVV	40.0	VV	V	20.0	28.0	70.0	
1	20/03/2016	025	16.0	18.0	0.0	NaH	NaH	WVV	30.0	VV	WVV	24.0	23.0	78.0	
2	22/03/2016	018	8.2	27.2	0.0	6.2	10.4	SW	30.0	NE	N	13.0	16.0	59.0	
3	04/12/2016	031	17.7	27.0	0.0	4.0	0.7	SW	33.0	S	SE	20.0	15.0	55.0	
4	20/03/2017	014	2.3	7.8	28.0	NaH	NaH	WVV	40.0	VV	WVV	15.0	8.0	55.0	

2. EXPLORATION DATA

□ Menghapus kolom yang tidak diperlukan pada data train dan data tes

	SuhuMin	SuhuMax	Kejadian	Pengukuran	SinarMatahari	ArahAnginTerkencang	KecepatanAnginTerkencang	ArahAnginRata	ArahAnginRata	KecepatanAnginRata	KecepatanAnginRata	KetinggianRata	KetinggianRata
0	15.4	15.5	4.0	NaH	NaH	WVV	24.0	NaH	WVV	0.0	13.0	78.0	78.0
1	8.0	17.5	8.0	3.0	7.4	NaH	NaH	SW	WVV	12.0	20.0	80.0	80.0
2	18.2	22.0	8.0	NaH	NaH	SE	44.0	SE	SE	19.0	29.0	81.0	81.0
3	7.5	24.5	0.0	8.4	10.4	SW	54.0	N	SW	12.0	19.0	30.0	30.0
4	9.0	20.1	8.0	3.0	12.0	N	37.0	WVV	SE	22.0	19.0	69.0	69.0
5	11.0	27.0	0.0	0.4	0.4	WVV	40.0	VV	V	20.0	28.0	70.0	70.0
6	16.0	18.0	0.0	NaH	NaH	WVV	30.0	VV	WVV	24.0	23.0	78.0	78.0
7	8.2	27.2	0.0	6.2	10.4	SW	30.0	NE	N	13.0	16.0	59.0	59.0
8	17.7	27.0	0.0	4.0	0.7	SW	33.0	S	SE	20.0	15.0	55.0	55.0
9	2.3	7.8	28.0	NaH	NaH	WVV	40.0	VV	WVV	15.0	8.0	55.0	55.0
10	15.4	15.5	4.0	NaH	NaH	WVV	24.0	NaH	WVV	0.0	13.0	78.0	78.0
11	8.0	17.5	8.0	3.0	7.4	NaH	NaH	SW	WVV	12.0	20.0	80.0	80.0
12	18.2	22.0	8.0	NaH	NaH	SE	44.0	SE	SE	19.0	29.0	81.0	81.0
13	7.5	24.5	0.0	8.4	10.4	SW	54.0	N	SW	12.0	19.0	30.0	30.0
14	9.0	20.1	8.0	3.0	12.0	N	37.0	WVV	SE	22.0	19.0	69.0	69.0
15	11.0	27.0	0.0	0.4	0.4	WVV	40.0	VV	V	20.0	28.0	70.0	70.0
16	16.0	18.0	0.0	NaH	NaH	WVV	30.0	VV	WVV	24.0	23.0	78.0	78.0
17	8.2	27.2	0.0	6.2	10.4	SW	30.0	NE	N	13.0	16.0	59.0	59.0
18	17.7	27.0	0.0	4.0	0.7	SW	33.0	S	SE	20.0	15.0	55.0	55.0
19	2.3	7.8	28.0	NaH	NaH	WVV	40.0	VV	WVV	15.0	8.0	55.0	55.0

- Mengelompokkan data numerik pada data train dan data test

```
[12] df_train._get_numeric_data().columns

Index(['SuhuMin', 'SuhuMax', 'Hujan', 'Penguapan', 'SinarMatahari',
       'KecepatanAnginTer kencang', 'KecepatanAngin9am', 'KecepatanAngin3pm',
       'Kelembaban9am', 'Kelembaban3pm', 'Tekanan9am', 'Tekanan3pm', 'Awan9am',
       'Awan3pm', 'Suhu9am', 'Suhu3pm'],
      dtype='object')

[13] df_test._get_numeric_data().columns

Index(['SuhuMin', 'SuhuMax', 'Hujan', 'Penguapan', 'SinarMatahari',
       'KecepatanAnginTer kencang', 'KecepatanAngin9am', 'KecepatanAngin3pm',
       'Kelembaban9am', 'Kelembaban3pm', 'Tekanan9am', 'Tekanan3pm', 'Awan9am',
       'Awan3pm', 'Suhu9am', 'Suhu3pm'],
      dtype='object')
```

- Meng-drop kolom yang tidak diperlukan.

```
> cols = [0,1]
df_test.drop(df_test.columns[cols], axis=1, inplace=True)
df_test
```

	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTer kencang	KecepatanAnginTer kencang	ArahAngin9am	ArahAngin3pm	KecepatanAngin9am	KecepatanAngin3pm	Kelembaban9am	Kelembaban3pm	Tekanan9am	Tekanan3pm
0	91.0	27.0	0.0	NaN	8.4	WNW	40.0	W	W	20.0	20.0	20.0	50.0	1013.0	
1	10.0	16.0	0.2	NaN	NaN	WNW	08.0	W	NW	24.0	33.0	70.0	30.0	1017.0	
2	0.0	27.0	0.0	0.0	10.4	SWW	33.0	NE	N	13.0	16.0	80.0	21.0	1016.0	
3	17.7	27.0	0.0	4.0	0.7	SW	30.0	E	SSE	20.0	15.0	50.0	47.0	1016.2	
4	2.0	7.0	30.0	NaN	NaN	SW	40.0	W	WNW	13.0	8.0	80.0	80.0	NaN	
...
10177	7.0	26.0	0.0	0.0	13.2	NE	31.0	ENE	NW	22.0	13.0	80.0	27.0	1017.1	
10178	10.4	26.0	0.0	0.0	11.0	SW	27.0	NE	WNW	17.0	30.0	80.0	30.0	1022.0	
10179	0.7	22.0	0.0	NaN	NaN	NaN	NaN	NaN	SW	0.0	4.0	24.0	20.0	NaN	
10180	12.0	26.7	0.0	0.0	10.0	NE	30.0	SW	ENE	2.0	20.0	80.0	40.0	1021.0	
10181	10.0	19.0	0.0	NaN	NaN	SE	31.0	SW	SSE	16.0	10.0	87.0	74.0	NaN	

10182 rows x 15 columns

- Menyimpan data train dan data test yang sudah diolah ke variabel baru yaitu train dan test

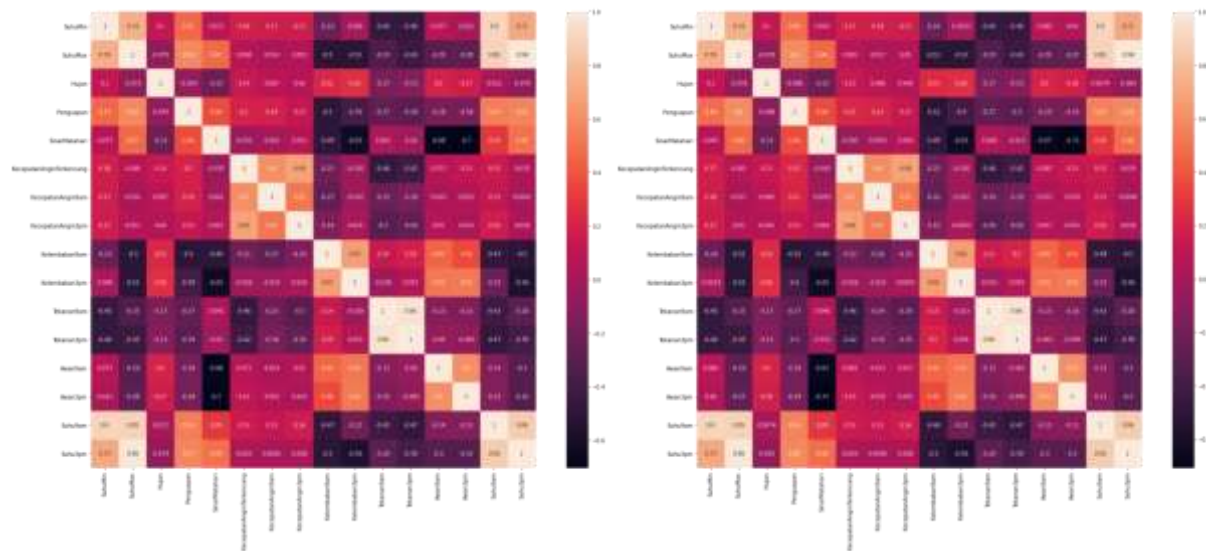
```
[14] train = df_train.copy()
test = df_test.copy()
```

3. PREPARATION DATA TRAIN & TEST

Dalam menyiapkan data train dan test, dilakukan beberapa hal:

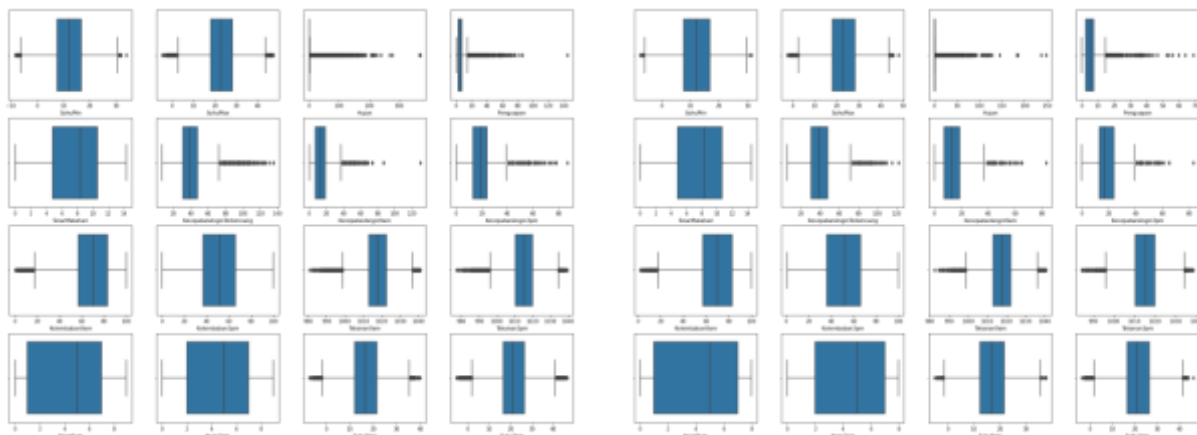
I. Korelasi Matriks

Matriks ini berfungsi untuk menampilkan korelasi setiap fitur.



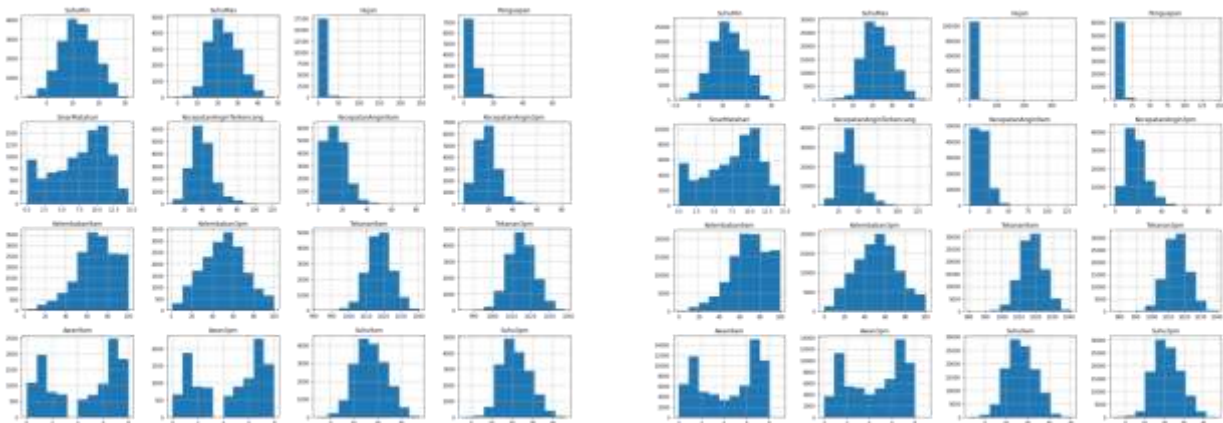
II. Finding Outliers

Menemukan outliers pada kedua data untuk dibersihkan nanti.



III. Distribution Data

Menampilkan persebaran data setiap fiturnya.



- IV. Count Plot
Menampilkan count plot untuk fitur yang tipenya kategorikal.
- V. Finding Missing Values
Mencari value yang bernilai NaN / Null untuk di-*impute* nanti.
- VI. Dealing Missing Values
Untuk data numerik, value kosong diisi dengan mean dari kolom tersebut.
Untuk data kategorik, value kosong diisi dengan modus dari kolom tersebut.
- VII. Dealing Outliers
Dengan library scipy, dilakukan *handle* outliers / pencilan untuk kolom yang memiliki outliers.
- VIII. Feature Engineering
 - A. Binning Atribut “Arah Angin”
Melakukan binning untuk atribut ArahAnginTerkencang, ArahAngin9am, dan ArahAngin3pm menjadi ['W', 'E', 'S', 'N']
 - B. Encode Categorical Data
Mengubah data kategorikal menjadi data numerik.
 - C. Scaling
Karena value semua data berada pada range 0~1000, maka dilakukan scalling data untuk mempersempit range menjadi 0~1 saja.

4. CLASSIFICATION

- Hasil akurasi dan confusion matrix menggunakan metode K-Neighbors Classifier

```
[105] from sklearn import metrics

print("Accuracy Dengan KNeighbors Classifier: ", metrics.accuracy_score(y_test, y_pred))

Accuracy Dengan KNeighbors Classifier: 0.8410289464432205

[106] from sklearn.metrics import classification_report, confusion_matrix

print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))

[[12980  631]
 [ 2082 1373]]
      precision    recall  f1-score   support

      0.0         0.86      0.95      0.91      13611
      1.0         0.69      0.40      0.50       3455

 accuracy         0.84      17066
 macro avg        0.77      0.68      0.70      17066
 weighted avg     0.83      0.84      0.82      17066
```

- Hasil akurasi dan confusion matrix menggunakan metode Decision Tree Classifier

```
[108] print("Accuracy Dengan Decision Tree Classifier: ", metrics.accuracy_score(y_test, y_pred))
```

Accuracy Dengan Decision Tree Classifier: 0.7664947849525372

```
[109] print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11322  2289]
 [ 1696  1759]]
```

	precision	recall	f1-score	support
0.0	0.87	0.83	0.85	13611
1.0	0.43	0.51	0.47	3455
accuracy			0.77	17066
macro avg	0.65	0.67	0.66	17066
weighted avg	0.78	0.77	0.77	17066

- Hasil akurasi dan confusion matrix menggunakan metode Naive-Bayes Classifier

```
[111] print("Accuracy Dengan Naive-Bayes Classifier: ", metrics.accuracy_score(y_test, y_pred))
```

Accuracy Dengan Naive-Bayes Classifier: 0.8023555607640923

```
[112] print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[11558  2053]
 [ 1320  2135]]
```

	precision	recall	f1-score	support
0.0	0.90	0.85	0.87	13611
1.0	0.51	0.62	0.56	3455
accuracy			0.80	17066
macro avg	0.70	0.73	0.72	17066
weighted avg	0.82	0.80	0.81	17066

- Hasil akurasi dan confusion matrix menggunakan metode Random Forest Classifier

```
[114] print("Accuracy Dengan Random Forest Classifier: ", metrics.accuracy_score(y_test, y_pred))
```

Accuracy Dengan Random Forest Classifier: 0.8534513066916677

```
[115] print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

```
[[12925  686]
 [ 1815  1640]]
```

	precision	recall	f1-score	support
0.0	0.88	0.95	0.91	13611
1.0	0.71	0.47	0.57	3455
accuracy			0.85	17066
macro avg	0.79	0.71	0.74	17066
weighted avg	0.84	0.85	0.84	17066

kNeighbors Classifier	Decision Tree Classifier	Naive-Bayes Classifier	Random Forest Classifier
0.8410289464432205	0.7664947849525372	0.8023555607640923	0.8534513066916677

Dari percobaan yang telah dilakukan dengan menggunakan data set dan data train diketahui bahwa menggunakan metode Random Forest Classifier menghasilkan hasil akurasi tertinggi dibandingkan dengan metode kNeighbors Classifier, Decision Tree Classifier, Naive-Bayes Classifier yaitu **0.8534513066916677**.