Tugas Besar Tahap Kedua Machine Learning

oleh:

Hafidz Lazuardi (NIM 1301184200)

Dhimas Hafid Kurniawan (NIM 1301184054)

IF-42-03

Program Studi S1 Informatika
Fakultas Informatika
Universitas Telkom
Bandung

2021

1. Formulasi Masalah

Pada tugas besar tahap kedua ini kami mendapatkan dataset berupa data latih (salju_train.csv) dan data uji (salju_test.csv) nomor dua tentang informasi yang nantinya digunakan untuk memprediksi apakah besok akan turun salju atau tidak. Proses memprediksi tersebut akan dilakukan dengan pengolahan dataset menggunakan model klasifikasi serta dengan melakukan eksperimen atau membandingkan dua kolom yaitu kolom BersaljuHariIni sebagai atribut untuk memprediksi hari ini akan turun salju atau tidak dan kolom BersaljuBesok sebagai atribut untuk memprediksi besok akan turun salju atau tidak. Detail proses akan dijelaskan pada bagian eksperimen dan analisis. Sehingga tujuan dari klasifikasi adalah memprediksi turunnya salju berdasarkan model yang dibuat.

2. Data Pre-Processing

Dataset yang didapatkan berupa data latih (salju_train.csv) dan data uji (salju_test.csv) tidak dapat diolah secara langsung. Sehingga perlu lakukan *data pre-processing* dengan tahapan-tahapan seperti *data cleansing, data splitting, data editing, data reduction* dan lain-lain. *Data pre-processing* akan dilakukan kepada data latih dan data uji sama persis. Sebagai tambahan, data latih digunakan untuk membuat model klasifikasi dan data uji digunakan untuk prediksi dan kalkulasi nilai akurasi. Setiap langkah akan diberikan *screenshot* dan penjelasan.

A. Langkah pertama adalah mengubah nama kolom untuk nantinya dapat divisualisasikan dalam satu grafik.

```
# Mengganti nama kolom "BersaljuBesok" menjadi "Class" pada datatest
datatest.rename(columns={'BersaljuBesok':'Class'}, inplace=True)

# Mengganti nama kolom 'BersaljuBesok' menjadi 'Class' pada datatrain
datatrain.rename(columns={'BersaljuBesok':'Class'}, inplace=True)
```

B. Melakukan drop kolom yang tidak diperlukan dan memilih kolom yang diperlukan untuk proses klasifikasi agar tidak terjadi data yang tidak digunakan atau *miss use data.*

```
#drop kolom yang tidak diperlukan pada datatrain
datatrain = datatrain.drop(columns=['id'])
```

```
#menampilkan jumlah data pada kolom class pada datatest
datatest['Class'].value_counts()
```

```
No     13824
Yes     3939
Name: Class, dtype: int64
```

```
#menampilkan jumlah data pada kolom class pada datatrain
datatrain['Class'].value_counts()
```

```
Tidak    82701
Ya       23963
Name: Class, dtype: int64
```

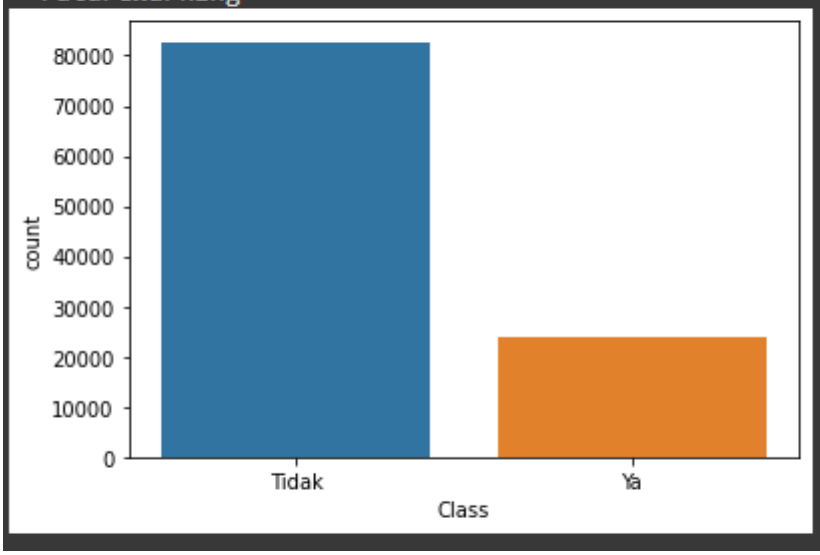
```
test_0 = datatest[datatest['Class'] == 'No']
test_1 = datatest[datatest['Class'] == 'Yes']

print("Class 0 :", test_0.shape)
print("Class 1 :", test_1.shape)
```

```
Class 0 : (13824, 23)
Class 1 : (3939, 23)
```

C. Melakukan visualisasi data pada kolom ‘class’ yang sebelumnya telah digabung dengan beberapa kolom lainnya.

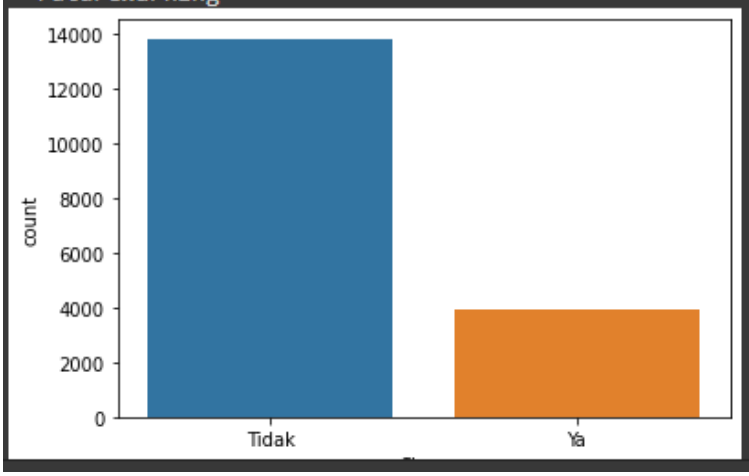
```
# Memvisualisasi data pada kolom 'Class' pada datatrain
train_plot = sns.countplot(datatrain['Class'])
train_plot.set_xticklabels(['Tidak','Ya'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:
  FutureWarning
```



```
# Memvisualisasi data pada kolom 'Class' pada datatest
train_plot = sns.countplot(datatest['Class'])
train_plot.set_xticklabels(['Tidak','Ya'])
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:
  FutureWarning
```



D. Melakukan *data oversampling* untuk menyeimbangkan data yang bertujuan meningkatkan akurasi yang didapat.

```
train_count_0, train_count_1 = datatrain['Class'].value_counts()
train_1_over = train_1.sample(train_count_0, replace = True)

salju_train_sampling = pd.concat([train_1_over, train_0], axis=0)

print("Total Class 1 and 0 :", salju_train_sampling['Class'].value_counts())
```

```
Total Class 1 and 0 : Ya       82701
Tidak    82701
Name: Class, dtype: int64
```

```
test_count_0, test_count_1 = datatest['Class'].value_counts()
test_1_over = test_1.sample(test_count_0, replace = True)

salju_test_sampling = pd.concat([test_1_over, test_0], axis=0)

print("Total Class 1 and 0 :", salju_test_sampling['Class'].value_counts())
```

```
Total Class 1 and 0 : Yes    13824
No     13824
Name: Class, dtype: int64
```

E. Melakukan pengecekan *missing value* agar tidak terjadi pengolahan data dengan nilai kosong namun dengan jumlah komputasi sama dengan pengolah data nilai tidak kosong.

```
salju_train_sampling.isnull().sum()
```

```
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
Class                         0
dtype: int64
```

```
salju_test_sampling.isnull().sum()
```

```
Tanggal                       0
KodeLokasi                    0
SuhuMin                     110
SuhuMax                      66
Hujan                       360
Penguapan                 11878
SinarMatahari             13023
ArahAnginTerkencang        1816
KecepatanAnginTerkencang   1807
ArahAngin9am               1837
ArahAngin3pm                708
KecepatanAngin9am           255
KecepatanAngin3pm           458
Kelembaban9am               386
Kelembaban3pm               656
Tekanan9am                 2621
Tekanan3pm                 2602
Awan9am                   10208
Awan3pm                   10677
Suhu9am                     214
Suhu3pm                     493
BersaljuHariIni             360
Class                         0
dtype: int64
```

F. Mengatasi *missing value* yang telah ditemukan dengan *replace* menggunakan median untuk nilai numerik dan modus untuk nilai string (*data wrangling*).

```
salju_train_sampling.fillna(salju_train_sampling.median(), inplace=True)
# Nilai Numeric diisi dengan nilai median

salju_train_sampling = salju_train_sampling.fillna(salju_train_sampling.mode().iloc[0])
# Nilai String diisi dengan nilai modus
salju_test_sampling.fillna(salju_test_sampling.median(), inplace=True)
# Nilai Numeric diisi dengan nilai median

salju_test_sampling = salju_test_sampling.fillna(salju_test_sampling.mode().iloc[0])
# Nilai String diisi dengan nilai modus
```

```
salju_train_sampling.isnull().sum()
salju_test_sampling.isnull().sum()
```

G. Menentukan *feature categorical* untuk dikonversi atau di-*encoding* tipe nilai data agar dapat dilakukan *modeling*

```
for col in categorical_columns:
    if col in salju_train_sampling.columns:
        enco = LabelEncoder()
        enco.fit(list(salju_train_sampling[col].astype(str).values))
        salju_train_sampling[col] = enco.transform(list(salju_train_sampling[col].astype(str).values))
```

```
salju_train_sampling.head()
```

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang
71296	649	5	2.6	6.5	0.0	4.4	6.9	1
5916	2891	13	14.8	18.8	16.8	4.4	6.9	13
80860	1710	31	20.2	23.7	0.0	5.4	1.7	4
73270	1279	25	11.2	14.6	0.8	1.4	4.6	15
81917	2082	26	7.0	13.1	0.0	4.4	6.9	5

```
numerics = ['int8', 'int16', 'int32', 'int64', 'float16', 'float32', 'float64']
categorical_columns = []
features = salju_test_sampling.columns.values.tolist()
for col in features:
    if salju_test_sampling[col].dtype in numerics: continue
    categorical_columns.append(col)
```

```
for col in categorical_columns:
    if col in salju_test_sampling.columns:
        enco = LabelEncoder()
        enco.fit(list(salju_test_sampling[col].astype(str).values))
        salju_test_sampling[col] = enco.transform(list(salju_test_sampling[col].astype(str).values))
```

```
salju_test_sampling.head()
```

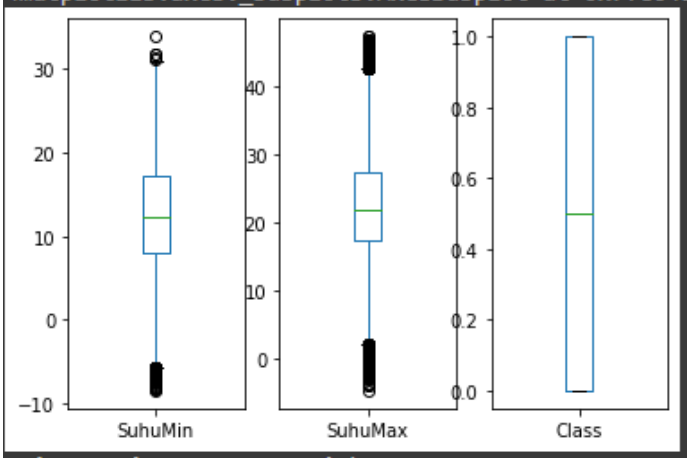
```
salju_test_sampling.head()
```

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMatahari	ArahAnginTerkencang
5234	2136	47	6.7	18.4	0.2	2.0	7.2	14
14799	2876	6	12.5	25.5	0.0	4.4	6.9	3
12951	381	24	5.8	22.4	0.0	2.4	10.2	15
7765	2026	39	8.1	16.4	7.2	4.4	6.9	8
7551	717	11	20.0	26.1	0.0	7.6	4.9	13

H. Melakukan pengecekan *outliers* untuk menentukan batas pengkategorian.

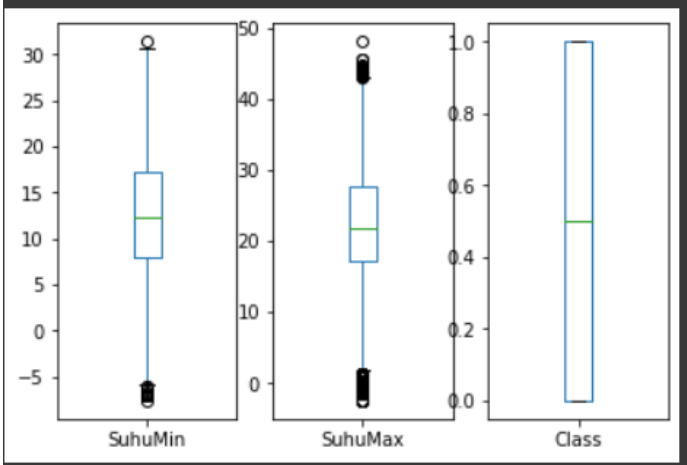
```
f,axes = plt.subplots(1,3)
plt.figure(figsize=(20,10))
salju_train_sampling['SuhuMin'].plot(kind='box',ax=axes[0])
salju_train_sampling['SuhuMax'].plot(kind='box',ax=axes[1])
salju_train_sampling['Class'].plot(kind='box',ax=axes[2])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc64ad6ee50>
```




```
<Figure size 1440x720 with 0 Axes>
```

```
f,axes = plt.subplots(1,3)
plt.figure(figsize=(20,10))
salju_test_sampling['SuhuMin'].plot(kind='box',ax=axes[0])
salju_test_sampling['SuhuMax'].plot(kind='box',ax=axes[1])
salju_test_sampling['Class'].plot(kind='box',ax=axes[2])

plt.show()
```




```
<Figure size 1440x720 with 0 Axes>
```

I. Melakukan perhitungan IQR atau *interquartile range outlier* untuk menentukan batas atas dan batas bawah

```
Q1=salju_train_sampling['SuhuMin'].quantile(0.25)
Q3=salju_train_sampling['SuhuMin'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1-1.5*IQR
Upper_Whisker = Q3+1.5*IQR
print(Lower_Whisker, Upper_Whisker)
```

```
8.0
17.2
9.2
-5.799999999999999 31.0
```

```
Q1=salju_train_sampling['SuhuMax'].quantile(0.25)
Q3=salju_train_sampling['SuhuMax'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1-1.5*IQR
Upper_Whisker = Q3+1.5*IQR
print(Lower_Whisker, Upper_Whisker)
```

```
17.2
27.4
10.2
1.9000000000000004 42.699999999999996
```

```
Q1=salju_test_sampling['SuhuMin'].quantile(0.25)
Q3=salju_test_sampling['SuhuMin'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1-1.5*IQR
Upper_Whisker = Q3+1.5*IQR
print(Lower_Whisker, Upper_Whisker)
```

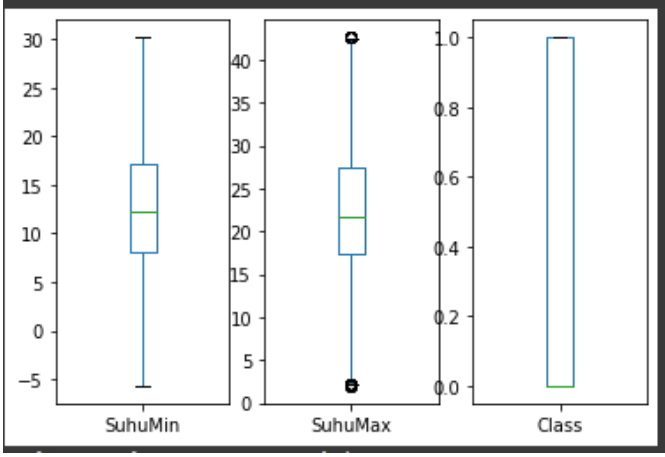
```
8.0
17.3
9.3
-5.950000000000001 31.25
```

```
Q1=salju_test_sampling['SuhuMax'].quantile(0.25)
Q3=salju_test_sampling['SuhuMax'].quantile(0.75)
IQR=Q3-Q1
print(Q1)
print(Q3)
print(IQR)
Lower_Whisker = Q1-1.5*IQR
Upper_Whisker = Q3+1.5*IQR
print(Lower_Whisker, Upper_Whisker)
```

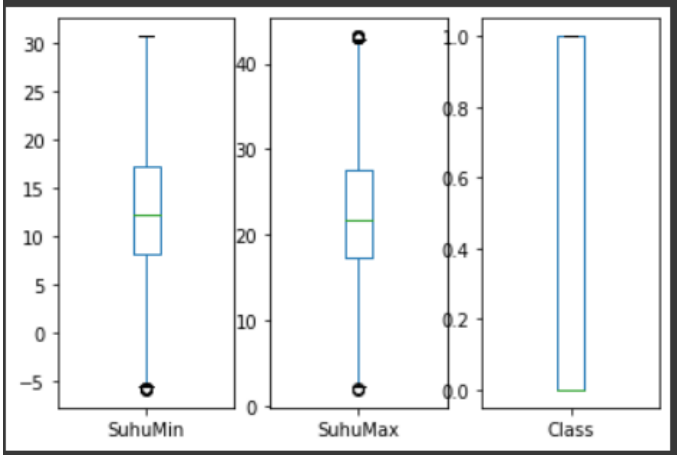
```
17.3
27.7
10.399999999999999
1.7000000000000028 43.3
```

J. Melakukan pengecekan *outliers* menggunakan IQR yang telah didapatkan dengan melakukan visualisasi.

```
f,axes = plt.subplots(1,3)
plt.figure(figsize=(20,10))
salju_train_sampling['SuhuMin'].plot(kind='box',ax=axes[0])
salju_train_sampling['SuhuMax'].plot(kind='box',ax=axes[1])
salju_train_sampling['Class'].plot(kind='box',ax=axes[2])

plt.show()
```




```
f,axes = plt.subplots(1,3)
plt.figure(figsize=(20,10))
salju_test_sampling['SuhuMin'].plot(kind='box',ax=axes[0])
salju_test_sampling['SuhuMax'].plot(kind='box',ax=axes[1])
salju_test_sampling['Class'].plot(kind='box',ax=axes[2])

plt.show()
```



K. Melakukan pengecekan data unik untuk menghindari data double.

```
def check(data):
  for col in data.columns:
    print(col, data[col].unique())

check(salju_train_sampling)
```

```
Tanggal [ 649 2891 1710 ... 3010 1002 3153]
KodeLokasi [ 5 13 31 25 26 37 17 12 30  8 10 48 42 33 24 35  0 40 34 19
 45 22  1 11 18 39 28 47 41 44  9  4 15 20 38  3 46 16 23 29 32  6 36
 43]
SuhuMin [ 2.6 14.8 20.2 11.2  7.   9.9 10.1 19.  17.5 21.4  9.5 17.  13
 23.9 19.4  7.3 14.7 11.8 -1.9 16.9  8.1 14.4  4.   8.5 10.   6.3  7.1
 18.5 16.5 18.7 24.4 20.4 18.  12.9 17.1 16.2 10.8 15.3 16.4 18.3  5.
 11.6 17.3 16.  26.  25.  13.  18.6  8.6 10.5 21.5 11.4 11.5 14.9  5.9
 15.6 17.7  5.4 -0.2 17.9  9.4  5.2 13.4 24.9  9.3 15.5  6.9 12.2 20.6
  5.3  8.2 15.9 13.2 15.8 22.6  8.4 11.9  4.8 15.   3.9  8.  20.9 18.1
 13.1 13.9 25.3 11.  24.5 20.7 19.6 12.3  8.7 19.8 24.7  9.1  9.  19.7
 12.7 19.2  8.9 11.3  0.3 10.3  2.8  7.5  5.5  9.6 11.1 17.2 13.5 13.7
 15.4 22.4  7.8 23.6 10.6  7.9 12.8 20.1 14.3  6.7  4.2  7.4  8.3 -1.1
```

```
def check(data):
  for col in data.columns:
    print(col, data[col].unique())

check(salju_test_sampling)
```

```
Tanggal [2136 2876  381 ...   62 1359  435]
KodeLokasi [47  6 24 39 11 20 22 37 17 31  9 13 10 46  2  7 38  5 45 19
 33 36 27 28  1  0  8 30 32 48 14  4 34 21 26 12 25 29 15 40 41 18 44 42
 43]
SuhuMin [ 6.7 12.5  5.8  8.1 20.   8.   9.6 20.4  6.2 13.3 19.2  5.7  9.
  9.2 19.6 10.4  8.3 11.1 12.  18.8 18.9 14.4  7.  20.8  7.9 23.6 26.
 13.7 20.1 19.7 27.4 22.5 22.6 10.8 14.6 17.   8.4 12.9 14.5 17.7 25.5
 20.7 16.7  9.3 18.3 11.9 11.4 23.8 11.7 16.4 21.6 18.6 16.6 16.2 14.3
  4.5 15.4 13.6 13.4 10.  16.8  9.4  8.9 13.  24.8 25.   6.6 12.2 14.8
 22.  12.3 24.7 23.2  2.9 20.6 11.5 28.1 14.1 11.8 15.8  1.6 14.   9.1
 11.3  6.5 20.3 10.6 11.2  4.2  5.  13.1  4.7 10.7  8.6 15.2  9.8 18.7
 -0.5 17.8 22.4  6.8 17.9 22.1  5.6  7.1 19.3  5.4 11.6 21.8  3.7 12.1
  7.3 26.7 13.8 19.4 10.9  7.6  5.5 10.1 19.  16.1  2.5 15.5 21.5 10.3
 15.7  2.2  8.8  8.7 21.2  7.8 24.6  8.2  6.1  0.5 12.7 17.3  8.5  5.2
 16.3 27.8 27.  14.7 24.4  5.1  1.7 12.6  9.9 10.2 15.1  6.3 25.2  4.4
```

L. Melakukan pengecekan tipe kolom untuk melakukan pengkategorian tipe data pada proses klasifikasi model di machine learning.

```
print(salju_train_sampling.dtypes)
```

```
Tanggal                      int64
KodeLokasi                   int64
SuhuMin                    float64
SuhuMax                    float64
Hujan                      float64
Penguapan                  float64
SinarMatahari              float64
ArahAnginTerkencang          int64
KecepatanAnginTerkencang   float64
ArahAngin9am                 int64
ArahAngin3pm                 int64
KecepatanAngin9am          float64
KecepatanAngin3pm          float64
Kelembaban9am              float64
Kelembaban3pm              float64
Tekanan9am                 float64
Tekanan3pm                 float64
Awan9am                    float64
Awan3pm                    float64
Suhu9am                    float64
Suhu3pm                    float64
BersaljuHariIni              int64
Class                        int64
dtype: object
```

```
print(salju_test_sampling.dtypes)
```

```
Tanggal                      int64
KodeLokasi                   int64
SuhuMin                    float64
SuhuMax                    float64
Hujan                      float64
Penguapan                  float64
SinarMatahari              float64
ArahAnginTerkencang          int64
KecepatanAnginTerkencang   float64
ArahAngin9am                 int64
ArahAngin3pm                 int64
KecepatanAngin9am          float64
KecepatanAngin3pm          float64
Kelembaban9am              float64
Kelembaban3pm              float64
Tekanan9am                 float64
Tekanan3pm                 float64
Awan9am                    float64
Awan3pm                    float64
Suhu9am                    float64
Suhu3pm                    float64
BersaljuHariIni              int64
Class                        int64
dtype: object
```

M. Melakukan replace dataset yang sudah diproses atau diolah (data tanpa feature engineering)

```
#Data train
x_train = salju_train_sampling.drop(['Class'], axis=1)
y_train = salju_train_sampling['Class'].values
```

```
x_train = x_train.values
```

```
#Data test
x_test = salju_test_sampling.drop(['Class'], axis=1)
y_test = salju_test_sampling['Class'].values
```

```
x_test = x_test.values
```

```
len(salju_train_sampling.columns)
```

```
23
```

```
len(salju_train_sampling.columns)
```

```
23
```

N. Melakukan *feature engineering* (*min-max normalization* atau *scaling*) untuk menghandle *outliers* yang telah dibuat.

```
# Normalisasi data train dengan menggunakan min-max normalization
# norm = pembeda variable telah melalui feature engineering
salju_train_sampling_norm = (salju_train_sampling - salju_train_sampling.min(
salju_train_sampling_norm.head(5)
```

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMat
75745	0.122766	0.541667	0.242340	0.371921	0.000000	0.030345	0.4
108371	0.541752	0.708333	0.331476	0.327586	0.011321	0.012414	0.4
79848	0.970993	0.770833	0.328691	0.258621	0.009164	0.008276	0.3
15317	0.524758	0.937500	0.860724	0.721675	0.031806	0.026207	0.2
102437	0.753296	0.437500	0.442897	0.366995	0.026954	0.006897	0.3

```
# Normalisasi data test dengan menggunakan min-max normalization
# norm = pembeda variable telah melalui feature engineering
salju_test_sampling_norm = (salju_test_sampling - salju_test_sampling.min()) /
salju_test_sampling_norm.head(5)
```

	Tanggal	KodeLokasi	SuhuMin	SuhuMax	Hujan	Penguapan	SinarMata
15679	0.213219	0.708333	0.377049	0.239709	0.010518	0.011628	0.47
15634	0.650779	0.937500	0.822404	0.806295	0.000000	0.113372	0.62
9955	0.591357	0.187500	0.631148	0.588378	0.071197	0.058140	0.44
1778	0.021608	0.541667	0.234973	0.334140	0.000000	0.063953	0.47
12983	0.432158	0.729167	0.751366	0.716707	0.009709	0.066860	0.26

O. Melakukan replace dataset yang sudah diproses atau diolah (data dengan feature engineering) untuk disimpan ke variabel yang berbeda dengan data tanpa feature engineering

```
#Data train
# norm = pembeda variable telah melalui feature engineering
x_train_norm = salju_train_sampling_norm.drop(['Class'], axis=1)
y_train_norm = salju_train_sampling_norm['Class'].values

x_train_norm = x_train_norm.values

#Data test
# norm = pembeda variable telah melalui feature engineering
x_test_norm = salju_test_sampling_norm.drop(['Class'], axis=1)
y_test_norm = salju_test_sampling_norm['Class'].values

x_test_norm = x_test_norm.values

len(salju_train_sampling_norm.columns)

23

len(salju_test_sampling_norm.columns)

23
```

3. Proses *Classification*

Pada tahap ini, diperlukan model yang dapat memprediksi permasalahan yang diangkat dimana class label sebagai atribut utama untuk contoh input data yang telah diberikan. Model diharuskan dapat melakukan klasifikasi atau mengelompokkan sesuai tujuan eksperimen. Naive bayes merupakan supervised learning algorithm yang berbasis bayes theorem. Sering digunakan untuk menyelesaikan kasus klasifikasi khususnya pada text classification problems. Digunakan library dalam membuat model untuk klasifikasi.

```
gnb = GaussianNB()
gnb.fit(x_train, y_train.ravel())
y_pred = gnb.predict(x_test)
```

```
gnb = GaussianNB()
gnb.fit(x_train_norm, y_train_norm.ravel())
y_pred_norm = gnb.predict(x_test_norm)
```

Untuk proses klasifikasi dengan algoritma *naive bayes* yang pertama dilakukan adalah membuat model Gaussian *classifier* dengan melakukan memanggil fungsi *naive bayes* berbasis *gaussian* yang telah didefinisikan sebelumnya pada library dan yang terakhir memprediksi respons untuk data test dengan memasukkan data uji dan data latih yang telah diolah yaitu dua jenis dataset. Dataset tersebut adalah dataset yang sudah melalui *feature engineering* dan tanpa melalui *feature engineering*. Sebagai info tambahan, data latih digunakan untuk membuat atau melatih model yang nantinya digunakan sebagai *classifier* dan data uji digunakan untuk memprediksi atau membuat *classifier* dapat merespon tentang mengenai data yang telah diberikan.

4. Eksperimen dan Analisis

Dimasukkan dataset yang sudah diolah ke algoritma naive bayes dan didapatkan akurasi sebagai berikut dengan 0 sebagai nilai numerik dari probabilitas “Tidak Turun Hujan Salju” dan 1 sebagai nilai numerik dari probabilitas “Turun Hujan Salju”

A. Data tanpa *feature engineering*

```
print("Accuracy :",metrics.accuracy_score(y_test, y_pred))
print('')
print(classification_report(y_test,y_pred))
```

```
Accuracy : 0.7426069897551406

              precision    recall  f1-score   support

           0       0.72      0.78      0.75     13779
           1       0.76      0.70      0.73     13747

    accuracy                           0.74     27526
   macro avg       0.74      0.74      0.74     27526
weighted avg       0.74      0.74      0.74     27526
```

Dengan menggunakan algoritma naive bayes dihasilkan akurasi sebesar 74% atau lebih tepatnya 74,26069897551406% . Sesuai dengan nilai precision, kemungkinan tidak terjadi hujan salju memiliki probabilitas 72% sedangkan kemungkinan terjadi hujan salju memiliki probabilitas 76%.

B. Data dengan *feature engineering*

```
print("Accuracy :",metrics.accuracy_score(y_test_norm, y_pred_norm))
print('')
print(classification_report(y_test_norm,y_pred_norm))
```

```
Accuracy : 0.7316896038345619

              precision    recall  f1-score   support

         0.0       0.72      0.77      0.74     13779
         1.0       0.75      0.69      0.72     13760

    accuracy                           0.73     27539
   macro avg       0.73      0.73      0.73     27539
weighted avg       0.73      0.73      0.73     27539
```

Dengan menggunakan algoritma naive bayes dihasilkan akurasi sebesar 73% atau lebih tepatnya 73,1689038345619%. Sesuai dengan nilai precision, kemungkinan tidak terjadi hujan salju memiliki probabilitas 72% sedangkan kemungkinan terjadi hujan salju memiliki probabilitas 75%.

5. Kesimpulan

Ada beberapa kesimpulan yang kami dapatkan setelah melakukan eksperimen dan analisis pada bagian sebelumnya, yaitu:

A. Algoritma *naive bayes* memiliki komputasi lebih cepat dalam *single class prediction* meskipun dapat berjalan juga dalam multi class prediction serta hanya memerlukan data latih yang tidak begitu banyak. Algoritma *naive bayes* berbasis gaussian yang dapat memproses dalam *normal distribution*.
B. Perbedaan komputasi pada data *pre-processing* khususnya jika terjadi *feature engineering* dapat menghasilkan akurasi yang berbeda juga.
C. Data tanpa *feature engineering* menghasilkan precision kemungkinan terjadi hujan salju lebih tinggi dengan angka 76 % dibandingkan data dengan *feature engineering* menghasilkan precision kemungkinan terjadi hujan salju lebih rendah dengan angka 75%.