LAPORAN TUGAS BESAR
PEMBELAJARAN MESIN LANJUT

Implementasi *Machine Learning Automation Tools* menggunakan Library TPOT dan AutoML Tables pada Google Cloud Platform untuk Prediksi Cuaca Hujan Australia

Laporan

disusun untuk memenuhi Tugas Besar Mata Kuliah Pembelajaran Mesin Lanjut

oleh:

Angel Metanosa Afinda (1301174639)
Muhammad Aziz Pratama (1301180018)

PROGRAM STUDI TEKNIK INFORMATIKA

FAKULTAS INFORMATIKA

UNIVERSITAS TELKOM

BANDUNG

2021

1. PENDAHULUAN

1.1 Pengenalan Dataset

Dataset WeatherAUS adalah sebuah dataset yang memiliki informasi tentang pengamatan cuaca harian dari berbagai lokasi di seluruh Australia yang diperoleh dari Biro Meteorologi Persemakmuran Australia. Kumpulan data weatherAUS diperbarui secara berkala, pembaruan dataset ini biasanya sesuai dengan pembaruan informasi terkini dari situs web Biro Meteorologi. Dataset ini menjadi salah satu sumber yang bertujuan untuk melatih model prediktif dan memprediksi kemungkinan hujan besok berdasarkan pengamatan cuaca hari ini yang ditandai dengan label Yes dan No pada variabel target RainTomorrow.

1.2 Rumusan Masalah

Rumusan masalah yang diangkat pada penelitian ini yaitu:

1. Bagaimana performa TPOT dalam mencari dan memprediksi algoritma, hyperparameter tuning, evaluasi, dan melakukan klasifikasi terbaik pada dataset weatherAUS?
2. Bagaimana performa AutoML Tables GCP dalam mencari dan memprediksi algoritma, hyperparameter tuning, evaluasi, dan melakukan klasifikasi terbaik pada dataset weatherAUS?

1.3 Tujuan

Tujuan yang ingin dicapai pada penelitian ini yaitu:

1. Mengetahui performa TPOT dalam mencari dan memprediksi algoritma, hyperparameter tuning, evaluasi, dan melakukan klasifikasi terbaik pada dataset weatherAUS.
2. Mengetahui performa AutoML Tables GCP dalam mencari dan memprediksi algoritma, hyperparameter tuning, evaluasi, dan melakukan klasifikasi terbaik pada dataset weatherAUS.

2. TINJAUAN PUSTAKA

2.1 *Machine Learning*

Machine Learning adalah aplikasi dari disiplin ilmu kecerdasan buatan (Artificial Intelligence) yang menggunakan teknik statistika untuk menghasilkan suatu model otomatis dari sekumpulan data, dengan tujuan memberikan komputer kemampuan untuk "belajar". Pembelajaran mesin atau machine learning memungkinkan komputer mempelajari sejumlah data (learn from data) sehingga dapat menghasilkan suatu model untuk melakukan proses input-output tanpa menggunakan kode program yang dibuat secara eksplisit. Proses belajar tersebut menggunakan algoritma khusus yang disebut machine learning algorithms. Terdapat banyak algoritma machine learning dengan efisiensi dan spesifikasi kasus yang berbeda-beda.

2.2 AutoML

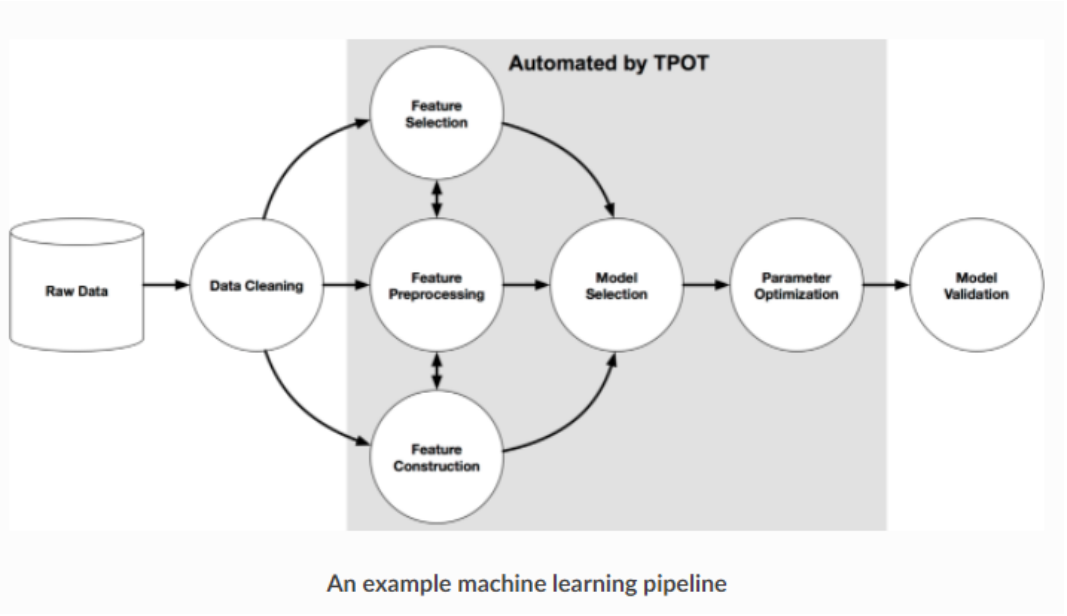
Auto Machine Learning (Auto ML) adalah kecerdasan buatan bisa belajar pemrograman kecerdasan buatan sendiri secara mandiri. Mesin pembelajaran ini telah mencapai keberhasilan yang cukup besar dan semakin banyak disiplin ilmu yang berkaitan dengannya.

Langkah-langkah untuk Auto Machine Learning

1. Persiapan dan konsumsi data (dari data mentah dan format lain-lain)
- Deteksi jenis kolom; misalnya boolean, numerik diskrit, numerik kontinu, atau teks
- Deteksi maksud kolom; misalnya target / label, bidang stratifikasi, fitur numerik, fitur teks kategorikal, atau fitur teks bebas
- Deteksi tugas; misalnya klasifikasi biner, regresi, pengelompokan, atau peringkat
2. Rekayasa fitur
- Pemilihan fitur
- Ekstraksi fitur
- Pembelajaran meta dan pembelajaran transfer
- Deteksi dan penanganan data miring dan / atau nilai yang hilang
3. Pemilihan model
4. Optimalisasi hyperparameter dari algoritma pembelajaran dan fiturisasi
5. Pilihan pipeline di bawah batasan waktu, memori, dan kompleksitas
6. Pemilihan matrik evaluasi dan prosedur validasi
7. Masalah pemeriksaan
- Deteksi kebocoran
- Deteksi kesalahan konfigurasi
8. Analisis hasil yang diperoleh
9. Antarmuka pengguna dan visualisasi untuk pembelajaran mesin otomatis

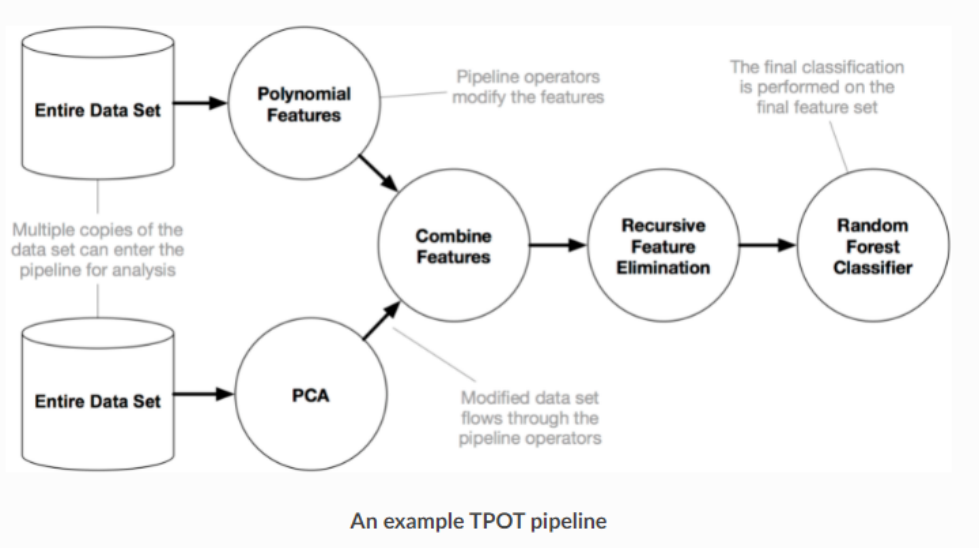
2.3 TPOT

TPOT adalah alat Pembelajaran Mesin Otomatis Python yang mengoptimalkan pipeline pembelajaran mesin menggunakan genetic programming. TPOT akan mengotomatisasikan bagian paling membosankan dari machine learning dengan secara cerdas menjelajahi ribuan kemungkinan pipeline untuk menemukan yang terbaik untuk data kita.



An example machine learning pipeline

Setelah TPOT selesai melakukan fungsinya, maka, TPOT mengeluarkan hasil berupa kode algortima terbaik beserta parameter tuning untuk pipeline terbaik sehingga developer dapat mengetahui dan melakukan eksplorasi selanjutnya dengan hasil tersebut.



An example TPOT pipeline

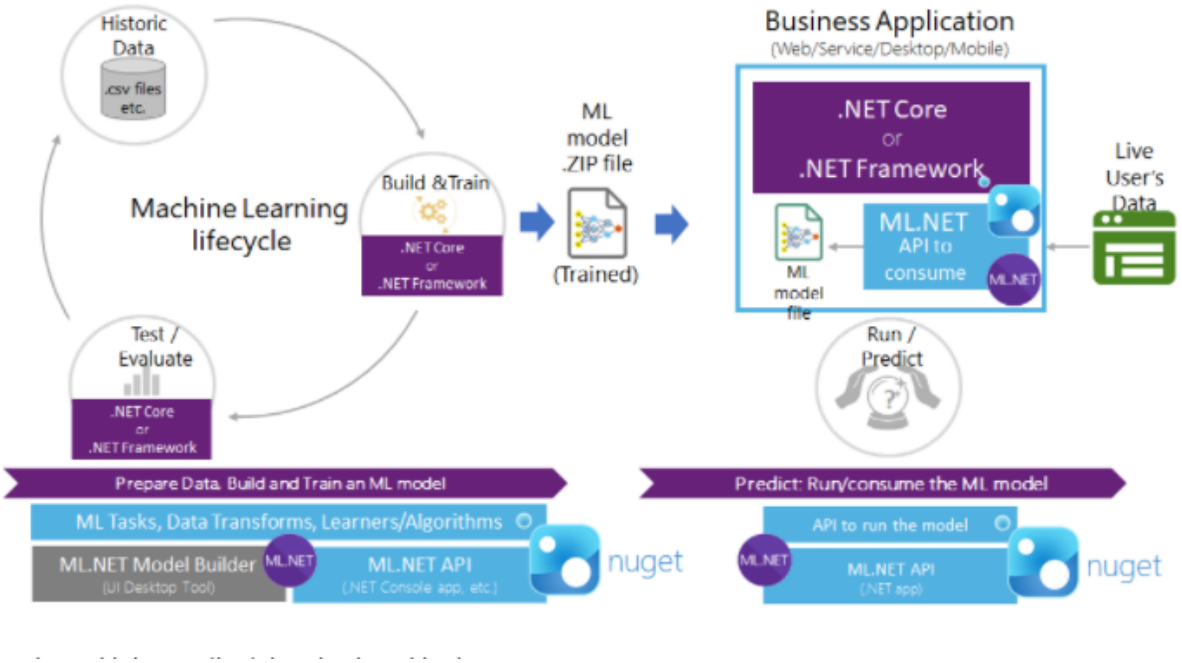
2.4 AutoML Table by Google Cloud Platform

Google Cloud Platform, adalah kumpulan layanan komputasi awan yang ditawarkan oleh Google. GCP berjalan di atas infrastruktur yang sama yang digunakan oleh Google untuk produk internalnya, seperti Google Search, YouTube dan Gmail. AutoML Tables adalah salah satu produk Google Cloud yang memungkinkan data scientist, analis, dan developer membuat dan menerapkan model machine learning tercanggih secara otomatis pada data terstruktur dengan kecepatan dan skala yang ditingkatkan secara masif.

Proyek yang digagas Google ini dapat menjadi solusi dari sedikitnya tenaga ahli artificial intelligence. Sebab banyak perusahaan yang menginginkan implementasi mesin

pembelajaran untuk meningkatkan bisnis mereka. Untuk membuat mesin pembelajaran, programmer harus menyusun algoritma terbaik dan melalui serangkaian percobaan. Proses ini digunakan dan dipelajari oleh Auto Machine Learning.

Proyek Auto ML Google merupakan sebuah pendekatan yang mengotomatisasi desain model pembelajaran mesin. Auto ML dapat merancang jaringan syaraf kecil yang bekerja sama dengan jaringan syaraf tiruan yang dirancang oleh pakar manusia, namun hasil ini dibatasi pada kumpulan data akademis kecil seperti CIFAR-10, dan Penn Treebank.

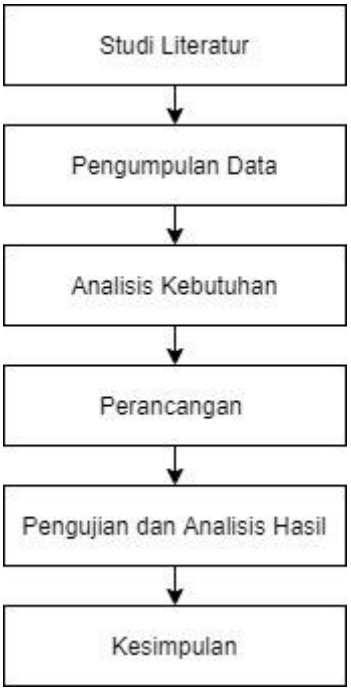


Berikut langkah-langkah pengerjaan AutoML Table pada GCP:

1. Pengumpulan Data
   Tentukan data yang dibutuhkan untuk melatih dan menguji model berdasarkan hasil yang ingin dicapai.
2. Persiapan Data
   Pastikan data Anda diformat dengan benar sebelum dan sesudah impor data
3. Latih
   Tetapkan parameter dan bangun model.
4. Evaluasi
   Meninjau hasil confusion matrix
5. Uji
   Menguji model pada data uji
6. Terapkan dan prediksi
   Jadikan model tersedia untuk digunakan

3. METODE PENELITIAN

Metodologi penelitian menjelaskan langkah-langkah yang akan digunakan serta perancangan dalam melakukan implementasi *machine learning automation tools* menggunakan library TPOT dan AutoML Tables pada Google Cloud Platform untuk prediksi cuaca hujan Australia. Tahapan metodologi penelitian ditunjukkan pada Gambar 1.



3.1 Pengumpulan Dataset

Dataset weatherAUS memiliki 145.460 baris dengan 23 *features/columns* dengan 16 *features* bertipe data float dan 7 *features* bertipe data object, dengan *features* sebagai berikut:

- Date : Tanggal observasi
- Location : Nama lokasi umum tempat stasiun cuaca
- MinTemp : Suhu minimal dalam celcius
- MaxTemp : Suhu maksimal dalam celcius
- Rainfall : Jumlah curah hujan yang tercatat untuk hari itu dalam mm
- Evaporation : Penguapan (mm) dalam 24 jam hingga jam 9 pagi
- Sunshine : Jumlah jam sinar matahari yang cerah dalam sehari.
- WindGustDir : Arah hembusan angin terkuat dalam 24 jam hingga tengah malam
- WindGustSpeed : Kecepatan (km/jam) hembusan angin terkuat dalam 24 jam hingga tengah malam
- WindDir9am : Arah angin pada jam 9 pagi
- WindDir3pm : Arah angin pada jam 3 sore
- WindSpeed9am : Kecepatan angin (km/jam) rata-rata lebih dari 10 menit sebelum jam 9 pagi
- WindSpeed3pm : Kecepatan angin (km/jam) rata-rata lebih dari 10 menit sebelum jam 3 sore
- Humidity9am : Kelembaban (persen) pada jam 9 pagi

- Humidity3pm : Kelembaban (persen) pada jam 3 sore
- Pressure9am : Tekanan atmosfer (hpa) berkurang menjadi rata-rata permukaan laut pada pukul 9 pagi
- Pressure3pm : Tekanan atmosfer (hpa) berkurang menjadi rata-rata permukaan laut pada pukul 3 sore
- Cloud9am : Pecahan langit tertutup awan pada pukul 9 pagi. Ini diukur dalam "okta", yang merupakan satuan delapan. Ini mencatat berapa perdelapan langit yang tertutup awan. Ukuran 0 menunjukkan langit yang benar-benar cerah sementara angka 8 menunjukkan bahwa itu benar-benar mendung.
- Cloud3pm : Bagian langit yang tertutup awan (dalam "oktas": seperdelapan) pada jam 3 sore. Lihat Cload9am untuk deskripsi nilai
- Temp9am : Suhu (derajat C) pada jam 9 pagi
- Temp3pm : Suhu (derajat C) pada jam 3 sore.
- RainToday : Boolean: 1 jika curah hujan (mm) dalam 24 jam hingga 9 pagi melebihi 1 mm, jika tidak 0
- RainTomorrow : Variabel Target. Prediksi kondisi hujan atau tidak pada besok harinya.

3.2 Analisis Kebutuhan

Analisis kebutuhan dilakukan dengan menentukan apa saja yang dibutuhkan dalam pengimplementasian *machine learning automation tools* menggunakan library TPOT dan AutoML Tables pada Google Cloud Platform untuk prediksi cuaca hujan Australia.

3.3 Pengujian dan Analisis

Pengujian yang dilakukan pada penelitian ini adalah pengujian terhadap pengaruh hasil dari AutoML oleh Library TPOT dan AutoML Table dari Google Cloud Platform. Pada pengujian ini, ada 3 tipe dataset yang akan diproses, yaitu:

1. Dataset dengan pre-processing seluruh nilai NaN yang dilakukan drop.
2. Dataset dengan pre-processing seluruh nilai NaN tidak dilakukan drop.
3. Dataset dengan pre-processing yang lengkap.

3.3.1 Hasil Implementasi dan Pengujian pada TPOT

Pada bagian implementasi ini, kami melakukan beberapa tahapan sebagai berikut:

A. Eksplorasi data

Terdapat beberapa tahapan awal untuk mempersiapkan data sebelum masuk ke dalam tahap training model dan eksperimen. Eksplorasi data terlebih dahulu untuk melihat apa saja dan bagaimana data yang terdapat pada dataset dan setelah itu di cleaning sebelum pemodelan dan training TPOT.

1. Read dataset weatherAUS yang telah dimiliki terlebih dahulu

```
df = pd.read_csv('/content/drive/MyDrive/malin lanjut/weatherAUS.csv')
```

2. Melihat info yang terdapat dari dataset weatherAUS

Date	Location	MinTemp	MaxTemp	Rainfall	Evaporation	Sunshine	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm
2008-12-01	Albury	13.4	22.9	0.6	NaN	NaN	W	44.0	W	WNW	20.0	24.0
2008-12-02	Albury	7.4	25.1	0.0	NaN	NaN	WNW	44.0	NNW	WSW	4.0	22.0
2008-12-03	Albury	12.9	25.7	0.0	NaN	NaN	WSW	46.0	W	WSW	19.0	26.0
2008-12-04	Albury	9.2	28.0	0.0	NaN	NaN	NE	24.0	SE	E	11.0	9.0
2008-12-05	Albury	17.5	32.3	1.0	NaN	NaN	W	41.0	ENE	NW	7.0	20.0

Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	No	No
44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	No	No
38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	23.2	No	No
45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	26.5	No	No
82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	29.7	No	No

```
df.info()
```

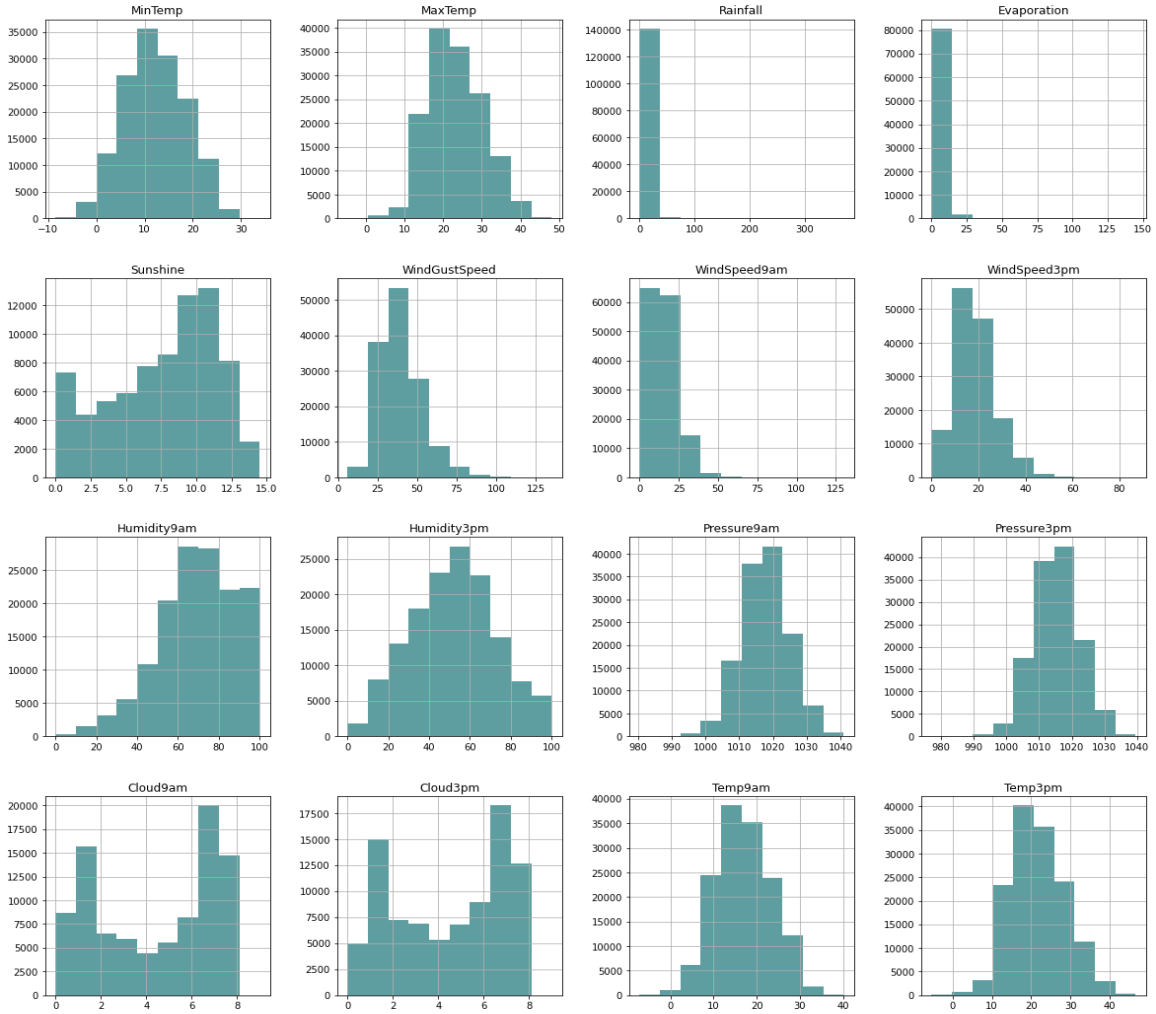
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 145460 entries, 0 to 145459
Data columns (total 23 columns):
 #   Column         Non-Null Count   Dtype
---  ------         --------------   -----
 0   Date           145460 non-null  object
 1   Location       145460 non-null  object
 2   MinTemp        143975 non-null  float64
 3   MaxTemp        144199 non-null  float64
 4   Rainfall       142199 non-null  float64
 5   Evaporation    82670 non-null   float64
 6   Sunshine       75625 non-null   float64
 7   WindGustDir    135134 non-null  object
 8   WindGustSpeed  135197 non-null  float64
 9   WindDir9am     134894 non-null  object
 10  WindDir3pm     141232 non-null  object
 11  WindSpeed9am   143693 non-null  float64
 12  WindSpeed3pm   142398 non-null  float64
 13  Humidity9am    142806 non-null  float64
 14  Humidity3pm    140953 non-null  float64
 15  Pressure9am    130395 non-null  float64
 16  Pressure3pm    130432 non-null  float64
 17  Cloud9am       89572 non-null   float64
 18  Cloud3pm       86102 non-null   float64
 19  Temp9am        143693 non-null  float64
 20  Temp3pm        141851 non-null  float64
 21  RainToday      142199 non-null  object
 22  RainTomorrow   142193 non-null  object
dtypes: float64(16), object(7)
memory usage: 25.5+ MB
```

3. Melihat missing values yang terdapat pada dataset

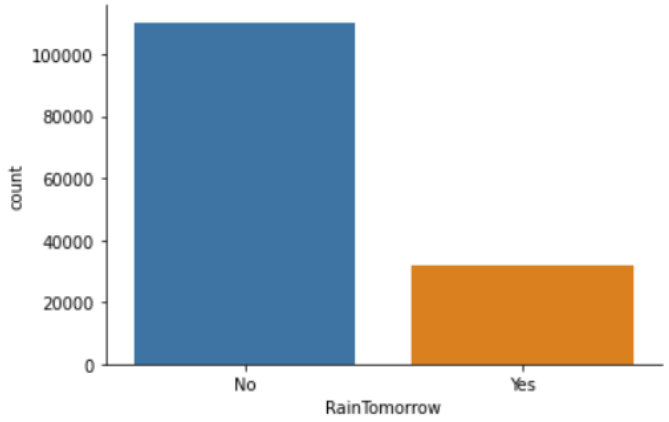
```
#melihat persentase missing values
Sunshine         48.009762
Evaporation      43.166506
Cloud3pm         40.807095
Cloud9am         38.421559
Pressure9am      10.356799
Pressure3pm      10.331363
WindDir9am        7.263853
WindGustDir       7.098859
WindGustSpeed     7.055548
Humidity3pm       3.098446
WindDir3pm        2.906641
Temp3pm           2.481094
RainTomorrow      2.245978
RainToday         2.241853
Rainfall          2.241853
WindSpeed3pm      2.105046
Humidity9am       1.824557
WindSpeed9am      1.214767
Temp9am           1.214767
MinTemp           1.020899
MaxTemp           0.866905
Location          0.000000
Date              0.000000
dtype: float64
```

```
#melihat jumlah missing value
Sunshine         69835
Evaporation      62790
Cloud3pm         59358
Cloud9am         55888
Pressure9am      15065
Pressure3pm      15028
WindDir9am       10566
WindGustDir      10326
WindGustSpeed    10263
Humidity3pm       4507
WindDir3pm        4228
Temp3pm           3609
RainTomorrow      3267
RainToday         3261
Rainfall          3261
WindSpeed3pm      3062
Humidity9am       2654
WindSpeed9am      1767
Temp9am           1767
MinTemp           1485
MaxTemp           1261
Location             0
Date                 0
dtype: int64
```

4. Melihat persebaran yang terdapat pada tiap feature di dataset



5. Melihat persebaran data dari target feature (RainTomorrow)



Pada eksperimen TPOT ini akan menggunakan 3 percobaan dengan tahapan persiapan data yang berbeda, yaitu :

1. Skenario Pertama

- Drop kolom feature yang terdapat banyak missing value nya seperti sunshine, evaporation, cloud3pm, cloud9am, dan date.
- Drop data yang memiliki missing value.

```
data_train = df.drop(columns=['Sunshine'])
data_train = data_train.drop(columns=['Evaporation'])
data_train = data_train.drop(columns=['Cloud3pm'])
data_train = data_train.drop(columns=['Cloud9am'])
data_train = data_train.drop(columns=['Date'])

data_train = data_train.dropna()
```

- Memberi label encoding pada data kategorikal menggunakan LabelEncoder agar menjadi bentuk numerikal.

```
#memberi label pada data kategorikal
le = LabelEncoder()
data_train['RainToday'] = le.fit_transform(data_train['RainToday'])
data_train['RainTomorrow'] = le.fit_transform(data_train['RainTomorrow'])
data_train['Location'] = le.fit_transform(data_train['Location'])
data_train['WindGustDir'] = le.fit_transform(data_train['WindGustDir'])
data_train['WindDir9am'] = le.fit_transform(data_train['WindDir9am'])
data_train['WindDir3pm'] = le.fit_transform(data_train['WindDir3pm'])
```

- Berikut contoh dataset yang telah diolah

Location	MinTemp	MaxTemp	Rainfall	WindGustDir	WindGustSpeed	WindDir9am	WindDir3pm	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
1	13.4	22.9	0.6	13	44.0	13	14	20.0	24.0	71.0	22.0	1007.7	1007.1	16.9	21.8	0	0
1	7.4	25.1	0.0	14	44.0	6	15	4.0	22.0	44.0	25.0	1010.6	1007.8	17.2	24.3	0	0
1	12.9	25.7	0.0	15	46.0	13	15	19.0	26.0	38.0	30.0	1007.6	1008.7	21.0	23.2	0	0
1	9.2	28.0	0.0	4	24.0	9	0	11.0	9.0	45.0	16.0	1017.6	1012.8	18.1	26.5	0	0
1	17.5	32.3	1.0	13	41.0	1	7	7.0	20.0	82.0	33.0	1010.8	1006.0	17.8	29.7	0	0

- Berikut jumlah data berdasarkan target dan sekarang data sudah tidak memiliki missing value.

```
RainTomorrow     0.0
RainToday        0.0
MinTemp          0.0
MaxTemp          0.0
Rainfall         0.0
WindGustDir      0.0
WindGustSpeed    0.0
WindDir9am       0.0
WindDir3pm       0.0
WindSpeed9am     0.0
WindSpeed3pm     0.0
Humidity9am      0.0
Humidity3pm      0.0
Pressure9am      0.0
Pressure3pm      0.0
Temp9am          0.0
Temp3pm          0.0
Location         0.0
```

```
data_train['RainTomorrow'].value_counts()
#jumlah data target
```

```
0    87906
1    25019
```

- Terakhir melakukan split data, X berisi data feature kolom selain target dan y berisi data target yaitu feature RainTomorrow. Setelah didapat X dan y, maka akan dilakukan train test split, agar mendapat data X train, y train dan X test, y test dengan perbandingan data train sebanyak 0.75 dan data test sebanyak 0.25.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

2. **Skenario Kedua**
   - Karena pada TPOT dapat melakukan imputing pada data nan/missing values, kali ini akan membiarkan nilai missing value tetap ada pada data
   - Percobaan ini hanya akan menggunakan kolom feature numerikal saja yaitu MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm', 'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm', 'Temp9am', 'Temp3pm', 'RainToday','RainTomorrow'.

```
col = ['MinTemp', 'MaxTemp', 'Rainfall', 'WindGustSpeed', 'WindSpeed9am', 'WindSpeed3pm',
       'Humidity9am', 'Humidity3pm', 'Pressure9am', 'Pressure3pm', 'Cloud9am', 'Cloud3pm',
       'Temp9am', 'Temp3pm', 'RainToday','RainTomorrow']
data_train = df[col]
```

   - Melakukan label encoding pada kolom feature RainToday dan RainTomorrow karena kolom tersebut masih berisi ‘yes’ atau ‘no’

```
le = LabelEncoder()
data_train['RainToday'] = le.fit_transform(data_train['RainToday'])
data_train['RainTomorrow'] = le.fit_transform(data_train['RainTomorrow'])
```

- Berikut contoh dataset yang telah diolah

MinTemp	MaxTemp	Rainfall	WindGustSpeed	WindSpeed9am	WindSpeed3pm	Humidity9am	Humidity3pm	Pressure9am	Pressure3pm	Cloud9am	Cloud3pm	Temp9am	Temp3pm	RainToday	RainTomorrow
13.4	22.9	0.6	44.0	20.0	24.0	71.0	22.0	1007.7	1007.1	8.0	NaN	16.9	21.8	0	0
7.4	25.1	0.0	44.0	4.0	22.0	44.0	25.0	1010.6	1007.8	NaN	NaN	17.2	24.3	0	0
12.9	25.7	0.0	46.0	19.0	26.0	38.0	30.0	1007.6	1008.7	NaN	2.0	21.0	23.2	0	0
9.2	28.0	0.0	24.0	11.0	9.0	45.0	16.0	1017.6	1012.8	NaN	NaN	18.1	26.5	0	0
17.5	32.3	1.0	41.0	7.0	20.0	82.0	33.0	1010.8	1006.0	7.0	8.0	17.8	29.7	0	0

- Terakhir melakukan split data, X berisi data feature kolom selain target dan y berisi data target yaitu feature RainTomorrow. Setelah didapat X dan y, maka akan dilakukan train test split, agar mendapat data X train, y train dan X test,y test dengan perbandingan data train sebanyak 0.75 dan data test sebanyak 0.25.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

3. Skenario Ketiga

Sebelumnya sudah diketahui bahwa dataset ini timpang atau *imbalanced*. Agar menjadi balance, maka akan dilakukan percobaan menggunakan metode oversample dengan library smote.

- Pertama input data missing values features numerik dengan nilai rata-rata tiap *feature*-nya.

```
# mengisi data missing values pada feature numerikal dengan nilai rata-rata tiap feature
data_train = df.copy()
data_train["MinTemp"]= data_train["MinTemp"].fillna(data_train["MinTemp"].mean())
data_train["MaxTemp"]= data_train["MaxTemp"].fillna(data_train["MaxTemp"].mean())
data_train["Evaporation"]= data_train["Evaporation"].fillna(data_train["Evaporation"].mean())
data_train["Sunshine"]= data_train["Sunshine"].fillna(data_train["Sunshine"].mean())
data_train["WindGustSpeed"]= data_train["WindGustSpeed"].fillna(data_train["WindGustSpeed"].mean())
data_train["Rainfall"]= data_train["Rainfall"].fillna(data_train["Rainfall"].mean())
data_train["WindSpeed9am"]= data_train["WindSpeed9am"].fillna(data_train["WindSpeed9am"].mean())
data_train["WindSpeed3pm"]= data_train["WindSpeed3pm"].fillna(data_train["WindSpeed3pm"].mean())
data_train["Humidity9am"]= data_train["Humidity9am"].fillna(data_train["Humidity9am"].mean())
data_train["Humidity3pm"]= data_train["Humidity3pm"].fillna(data_train["Humidity3pm"].mean())
data_train["Pressure9am"]= data_train["Pressure9am"].fillna(data_train["Pressure9am"].mean())
data_train["Pressure3pm"]= data_train["Pressure3pm"].fillna(data_train["Pressure3pm"].mean())
data_train["Cloud9am"]= data_train["Cloud9am"].fillna(data_train["Cloud9am"].mean())
data_train["Cloud3pm"]= data_train["Cloud3pm"].fillna(data_train["Cloud3pm"].mean())
data_train["Temp9am"]= data_train["Temp9am"].fillna(data_train["Temp9am"].mean())
data_train["Temp3pm"]= data_train["Temp3pm"].fillna(data_train["Temp3pm"].mean())
```

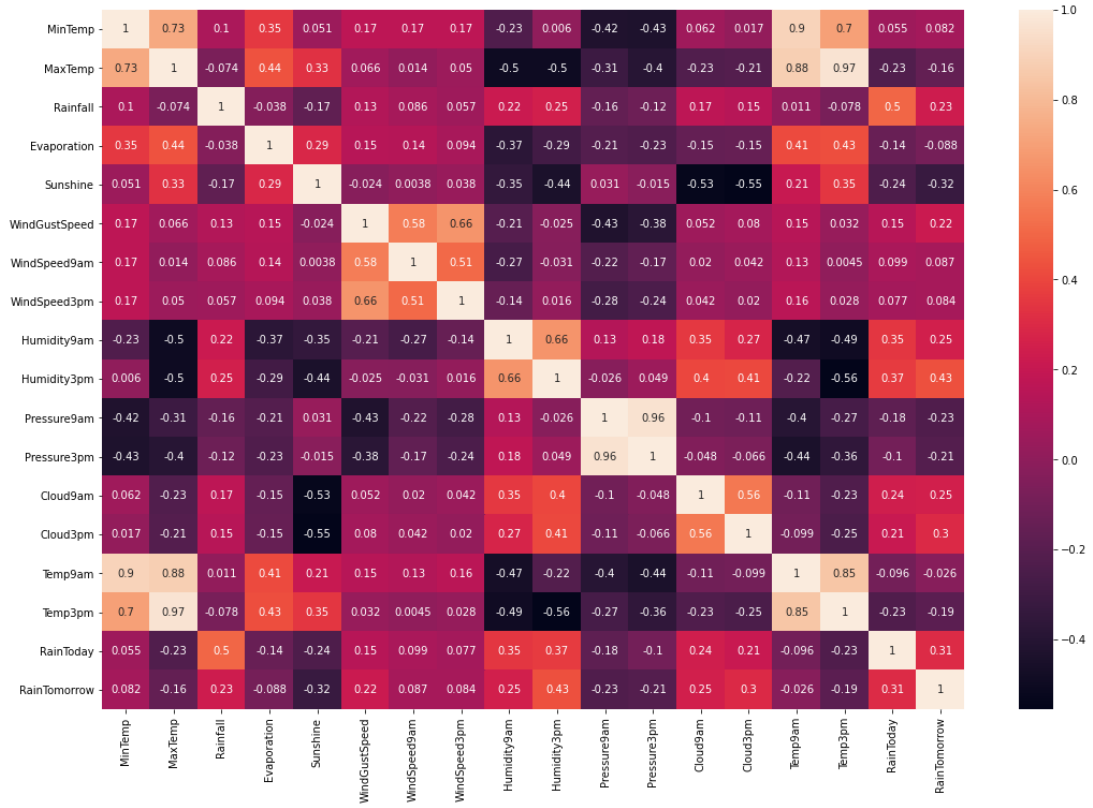
- Memberi label terlebih dahulu pada RainToday (yes = 1, no = 0) dan RainTomorrow (yes = 1, no = 0)

```
data_train['RainTomorrow'] = data_train['RainTomorrow'].map({'Yes': 1, 'No': 0})
data_train['RainToday'] = data_train['RainToday'].map({'Yes': 1, 'No': 0})
```

- Lalu missing values pada feature kategorikal kita input dengan nilai modus tiap feature nya

```
# mengisi data missing values pada feature kategorikal dengan modus tiap feature
data_train['RainToday']=data_train['RainToday'].fillna(data_train['RainToday'].mode()[0])
data_train['RainTomorrow']=data_train['RainTomorrow'].fillna(data_train['RainTomorrow'].mode()[0])
data_train['WindDir9am'] = data_train['WindDir9am'].fillna(data_train['WindDir9am'].mode()[0])
data_train['WindGustDir'] = data_train['WindGustDir'].fillna(data_train['WindGustDir'].mode()[0])
data_train['WindDir3pm'] = data_train['WindDir3pm'].fillna(data_train['WindDir3pm'].mode()[0])
```

- Melihat korelasi tiap feature dengan menggunakan visualisasi heatmap



- Drop data yang sangat berkorelasi (positif), pada kolom temp3pm, temp9am, humidity9am beserta date.

```
data_train=data_train.drop(['Temp3pm','Temp9am','Humidity9am',"Date"],axis=1)
data_train.columns
```

- Melakukan label encoding pada sisa kolom kategorikal yang ada dengan LabelEncoder

```
le = LabelEncoder()
data_train["Location"] = le.fit_transform(data_train["Location"])
data_train["WindDir9am"]= le.fit_transform(data_train["WindDir9am"])
data_train["WindDir3pm"]= le.fit_transform(data_train["WindDir3pm"])
data_train["WindGustDir"] = le.fit_transform(data_train["WindGustDir"])
```

- Melakukan split data, X berisi data feature kolom selain target dan y berisi data target yaitu feature RainTomorrow.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

```
from imblearn.over_sampling import SMOTE
os = SMOTE()
X_train, y_train = os.fit_resample(X_train, y_train)
```

- Setelah didapat X dan y, maka akan dilakukan train test split, agar mendapat data X train, y train dan X test,y test dengan perbandingan data train sebanyak 0.75 dan data test sebanyak 0.25.

```
Counter({0.0: 85079, 1.0: 85079})
```

- Setelah itu kita melakukan oversampling dengan library smote yang dapat memperbanyak data target minoritas (RainTomorrow = 1) menjadi 1:1 jumlahnya sebanding dengan data target mayoritas (RainTomorrow = 0). sekarang data train target sudah sebanding, yaitu sama-sama banyak sekitar 85ribuan.

B. Pemodelan

Pada pemodelan ini menggunakan library TPOT. TPOT merupakan salah satu tools AutoML yang dapat memudahkan kita dalam mencari model machine learning mana yang paling baik untuk dataset yang kita miliki.

Kami memilih memakai TPOT karena pemakaian TPOT sangat mudah dan TPOT dibangun di atas scikit-learn, jadi semua kode yang dihasilkannya akan terlihat familiar jika kita terbiasa dengan *scikit-learn*. Kita hanya perlu melakukan data cleaning dan TPOT dapat mengurus sisanya.

Tahap-tahapan penggunaan TPOT :

- Sebelum memakai TPOT, kita perlu menyiapkan data yang sudah di *cleaning* terlebih dahulu, lalu memisahkan feature mana yang digunakan dan dijadikan target. Setelah itu kita melakukan split data menjadi data train dan data test.
- Kita dapat mengatur seberapa banyak generasi dan populasi yang kita inginkan maupun waktu yang dibutuhkan. Setelah diatur, kita dapat melakukan training model TPOT dengan data train dan TPOT akan melakukan

pemrosesan data dan mencari algoritma machine learning mana yang paling baik untuk data yang kita punya.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

```
tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2, n_jobs=-1)
tpot.fit(X_train, y_train)
```

C. Eksperimen

Pada tahap ini, kami melakukan eksperimen pertama dengan menggunakan data yang telah di cleaning pada skenario pertama yang telah dijelaskan sebelumnya. Dengan memakai data yang telah di cleaning pada skenario pertama, kita memiliki data kelas sebagai berikut :

```
data_train['RainTomorrow'].value_counts()
#jumlah data target
```

```
0    87906
1    25019
```

Setelah data displit dan dijadikan X_train, y_train dan X_test, y_test, kita fitting data X dan y train ke model TPOT.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

```
tpot = TPOTClassifier(generations=5, population_size=50, verbosity=2, n_jobs=-1)
tpot.fit(X_train, y_train)
```

Hasil dari training TPOT :

```
Generation 1 - Current best internal CV score: 0.8574380091577796

Generation 2 - Current best internal CV score: 0.8587604284566173

Generation 3 - Current best internal CV score: 0.8587604284566173

Generation 4 - Current best internal CV score: 0.8587604284566173

Generation 5 - Current best internal CV score: 0.8587604284566173

Best pipeline: LinearSVC(RobustScaler(GradientBoostingClassifier(input_matrix, learning_rate=0.1, max_depth=8, max_features=0.
TPOTClassifier(config_dict=None, crossover_rate=0.1, cv=5,
               disable_update_check=False, early_stop=None, generations=5,
               log_file=None, max_eval_time_mins=5, max_time_mins=None,
               memory=None, mutation_rate=0.9, n_jobs=-1, offspring_size=None,
               periodic_checkpoint_folder=None, population_size=50,
               random_state=None, scoring=None, subsample=1.0, template=None,
               use_dask=False, verbosity=2, warm_start=False)
```

TPOT memberi tahu score Cross-Validation antar data training saat sedang running. TPOT memberikan informasi algoritma machine learning apa yang paling baik beserta hyperparameter untuk data yang kita miliki saat telah selesai running.
Dari hasil diatas, kita memiliki model TPOT yang berisi algoritma LinearSVC dan hyperparameter nya. Setelah itu, kita dapat menguji nya pada data test yang telah kita split sebelumnya.

```
print('accuration for this pipeline : ', tpot.score(X_test, y_test))
```

```
accuration for this pipeline :  0.8571479172570133
```

```
y_pred = tpot.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

           0       0.89      0.93      0.91     21840
           1       0.73      0.59      0.65      6392

    accuracy                           0.86     28232
   macro avg       0.81      0.76      0.78     28232
weighted avg       0.85      0.86      0.85     28232
```

Pada skenario ini didapat hasil akurasi sekitar 0.86 dan mendapatkan f1-score pada RainTomorrow(0) = 0.91 dan f1-score RainTomorrow(1) = 0.65

D. Evaluasi

Setelah melakukan eksperimen pertama, diketahui bahwa setelah generasi kedua tidak terjadi perubahan nilai akurasi cross-validation yang dilakukan oleh TPOT. oleh karena itu, pada 2 skenario selanjutnya, akan dilakukan penurunan generasi maupun populasi.

- **Percobaan model pada data train skenario kedua**

Dengan data train yang telah kita siapkan pada skenario kedua, pada data ini memiliki nilai NaN, dan kita membiarkan TPOT untuk menangani data NaN yang terdapat pada data train. Lalu generasi diturunkan 1, dan menjadi 4 generasi saja, ini dilakukan agar dapat menurunkan waktu running TPOT agar tidak selama pada skenario pertama.

```
X = data_train.drop(columns=['RainTomorrow'])
y = data_train.RainTomorrow
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size = 0.75, test_size = 0.25)
```

```
tpot = TPOTClassifier(generations=4, population_size=50, verbosity=2, n_jobs=-1)
tpot.fit(X_train, y_train)
```

```
Generation 1 - Current best internal CV score: 0.851633677431575

Generation 2 - Current best internal CV score: 0.851633677431575

Generation 3 - Current best internal CV score: 0.8520314423714368

Generation 4 - Current best internal CV score: 0.8527512075007102

Best pipeline: GradientBoostingClassifier(input_matrix, learning_rate=0.1, max_depth=6, max_features=0.5, min_samples_leaf=12,
TPOTClassifier(config_dict=None, crossover_rate=0.1, cv=5,
               disable_update_check=False, early_stop=None, generations=4,
               log_file=None, max_eval_time_mins=5, max_time_mins=None,
               memory=None, mutation_rate=0.9, n_jobs=-1, offspring_size=None,
               periodic_checkpoint_folder=None, population_size=50,
               random_state=None, scoring=None, subsample=1.0, template=None,
               use_dask=False, verbosity=2, warm_start=False)
```

Pada skenario kedua ini, TPOT mendapatkan Gradient Boosting Classifier sebagai algoritma machine learning yang paling baik untuk data ini. Setelah itu, kita dapat menguji nya pada data test yang telah kita split sebelumnya.

```
print('accuration for this pipeline : ', tpot.score(X_test, y_test))
```

```
Imputing missing values in feature set
accuration for this pipeline :  0.8564082166093702
```

```
y_pred = tpot.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
Imputing missing values in feature set
              precision    recall  f1-score   support

           0       0.88      0.95      0.91     27542
           1       0.75      0.52      0.61      7655

    accuracy                           0.86     35197
   macro avg       0.81      0.73      0.76     35197
weighted avg       0.85      0.86      0.85     35197
```

Pada skenario ini didapat hasil akurasi sekitar 0.86 dan mendapatkan f1-score pada RainTomorrow(0) = 0.91, sedangkan f1-score RainTomorrow(1) = 0.61.

- **Percobaan model pada data train skenario ketiga**

Dengan data train yang telah kita siapkan pada skenario ketiga, pada data ini memiliki nilai tidak memiliki nilai NaN dan sudah tidak imbalanced karena di oversampling data kelas minoritas (RainTomorrow = 1). Lalu generasi dibuat menjadi 3 sedangkan populasi nya 30, ini dilakukan agar dapat menurunkan waktu running TPOT agar tidak terlalu lama karena pada skenario ini memiliki jumlah data yang lebih banyak dikarenakan oversampling.

```
tpot = TPOTClassifier(generations=3, population_size=30, verbosity=2, n_jobs=-1)
tpot.fit(X_train, y_train)
```

```
Optimization Progress: 88%  144/? [2:41:51<00:00, 71.43s/pipeline]

Generation 1 - Current best internal CV score: 0.8932348797565405

Generation 2 - Current best internal CV score: 0.8932348797565405

Generation 3 - Current best internal CV score: 0.8949861954524524

Best pipeline: ExtraTreesClassifier(input_matrix, bootstrap=True, criterion=entropy, max_features=0.6500000000000001, min_samples_leaf=1, min_samples_split=10, n_estimators=100)
TPOTClassifier(config_dict=None, crossover_rate=0.1, cv=5,
               disable_update_check=False, early_stop=None, generations=3,
               log_file=None, max_eval_time_mins=5, max_time_mins=None,
               memory=None, mutation_rate=0.9, n_jobs=-1, offspring_size=None,
               periodic_checkpoint_folder=None, population_size=30,
               random_state=None, scoring=None, subsample=1.0, template=None,
               use_dask=False, verbosity=2, warm_start=False)
```

Pada skenario ketiga ini, TPOT mendapatkan Extra Tree Classifier sebagai algoritma machine learning yang paling baik untuk data ini. Setelah itu, kita dapat menguji nya pada data test yang telah kita split sebelumnya.

```
print('accuration for this pipeline : ', tpot.score(X_test, y_test))
```

```
accuration for this pipeline :  0.8476282139419772
```

```
y_pred = tpot.predict(X_test)
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.90      0.90      0.90     28504
         1.0       0.65      0.64      0.65      7861

    accuracy                           0.85     36365
   macro avg       0.78      0.77      0.77     36365
weighted avg       0.85      0.85      0.85     36365
```

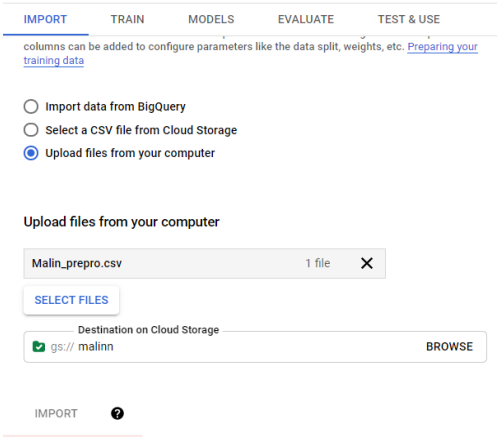
Pada skenario ini didapat hasil akurasi sekitar 0.85 dan mendapatkan f1-score pada RainTomorrow(0) = 0.90, sedangkan f1-score RainTomorrow(1) = 0.65.

3.3.2 Hasil Implementasi dan Pengujian pada AutoML Table GCP

Pada bagian implementasi ini, kami melakukan beberapa tahapan sebagai berikut:

1. **Eksplorasi Dataset**

   Langkah-langkah eksplorasi dan pembersihan dataset yang digunakan pada AutoML Table GCP melakukan langkah-langkah yang sama dengan dataset pada implementasi data pada Library TPOT

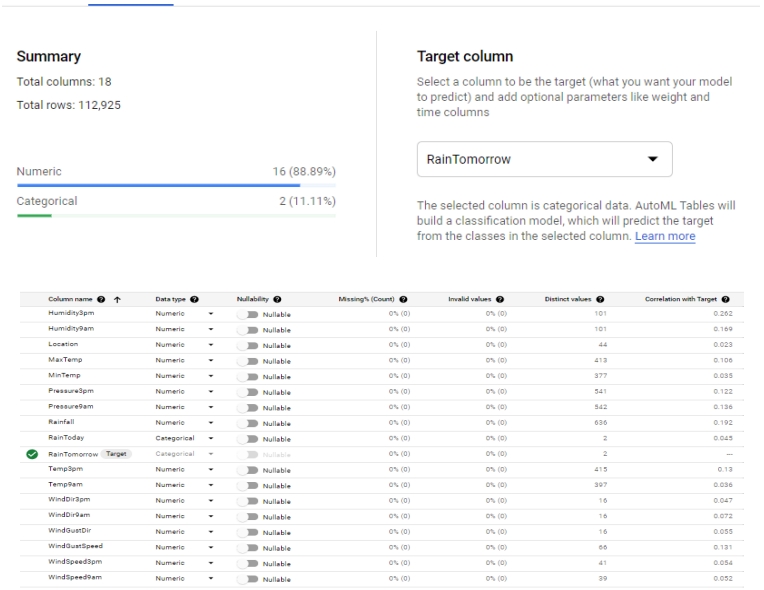


2. **Import Dataset**

Melakukan import dataset yang tersimpan pada device. Dataset ini adalah data akhir yang sudah diekspor dari proses pre-processing. Pada bagian import dataset, ada 3 dataset yang diimport pada lembar kerja yang berbeda.

3. Train Dataset

a. Dataset dengan melakukan drop data NaN

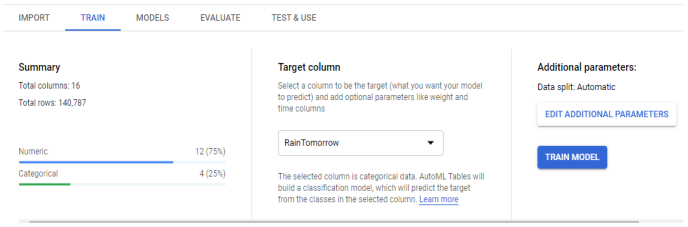


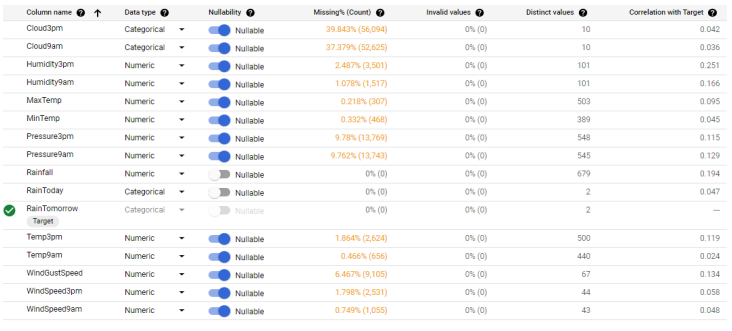
Pada dataset drop NaN, terdapat persebaran dataset sebanyak 88,89% untuk tipe data numeric dan 11,11% untuk tipe data categorical. Kemudian dilakukan train model dengan memilih target kolom yaitu RainTomorrow.

Pada gambar diatas, dapat diketahui bahwa dataset sudah bersih dari nilai null/NaN.

b. Dataset dengan tidak melakukan drop data NaN

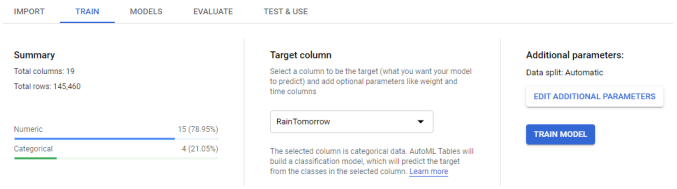
Pada dataset drop NaN, terdapat persebaran dataset sebanyak 75% untuk tipe data numeric dan 25% untuk tipe data categorical. Kemudian dilakukan train model dengan memilih target kolom yaitu RainTomorrow.



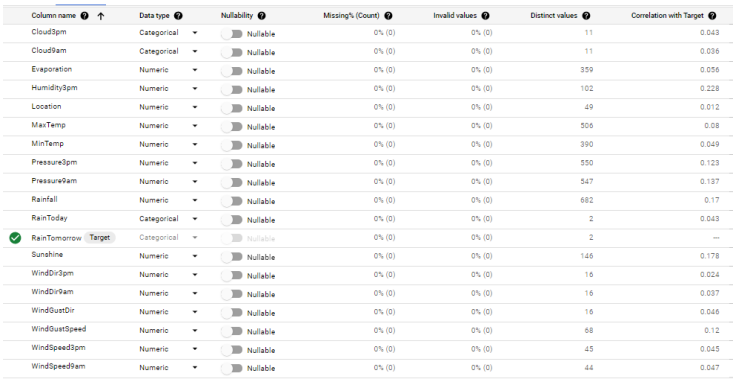


Pada dataset yang tidak melakukan drop nilai NaN dapat diketahui bahwa sebagian besar kolom masih memiliki nilai NaN. Pada AutoML Table ini memiliki kemampuan untuk mendeteksi dan memproses nilai NaN dengan handle NaN yang terbaik.

c. Dataset dengan melakukan preprocessing untuk dataset oversampling.



Pada dataset *oversampling handle*, terdapat persebaran dataset sebanyak 78,95% untuk tipe data numeric dan 21,05% untuk tipe data categorical. Kemudian dilakukan train model dengan memilih target kolom yaitu RainTomorrow.



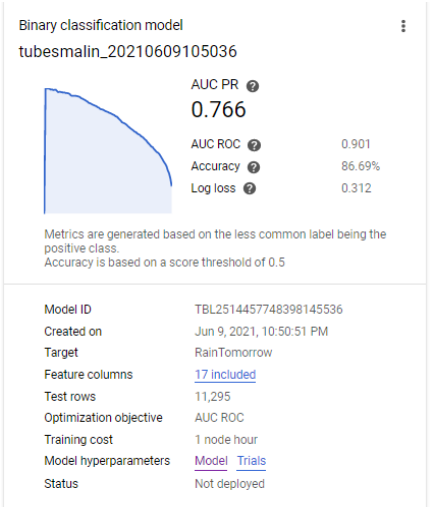
Column name	Data type	Nullability	Missing% (Count)	Invalid values	Distinct values	Correlation with Target
Cloud3pm	Categorical	Nullable	0% (0)	0% (0)	11	0.043
Cloud9am	Categorical	Nullable	0% (0)	0% (0)	11	0.036
Evaporation	Numeric	Nullable	0% (0)	0% (0)	359	0.056
Humidity3pm	Numeric	Nullable	0% (0)	0% (0)	102	0.228
Location	Numeric	Nullable	0% (0)	0% (0)	49	0.012
MaxTemp	Numeric	Nullable	0% (0)	0% (0)	506	0.08
MinTemp	Numeric	Nullable	0% (0)	0% (0)	390	0.049
Pressure3pm	Numeric	Nullable	0% (0)	0% (0)	550	0.123
Pressure9am	Numeric	Nullable	0% (0)	0% (0)	547	0.137
Rainfall	Numeric	Nullable	0% (0)	0% (0)	682	0.17
RainToday	Categorical	Nullable	0% (0)	0% (0)	2	0.043
RainTomorrow Target	Categorical	Nullable	0% (0)	0% (0)	2	--
Sunshine	Numeric	Nullable	0% (0)	0% (0)	146	0.178
WindDir3pm	Numeric	Nullable	0% (0)	0% (0)	16	0.024
WindDir9am	Numeric	Nullable	0% (0)	0% (0)	16	0.037
WindGustDir	Numeric	Nullable	0% (0)	0% (0)	16	0.046
WindGustSpeed	Numeric	Nullable	0% (0)	0% (0)	68	0.12
WindSpeed3pm	Numeric	Nullable	0% (0)	0% (0)	45	0.045
WindSpeed9am	Numeric	Nullable	0% (0)	0% (0)	44	0.047

Pada dataset dengan kasus oversampling ini dapat diketahui bahwa dataset sudah tidak memiliki nilai NaN dan terdapat beberapa kolom baru hasil dari tambahan data setelah melakukan *oversampling handle.*

4. Membangun Pemodelan Klasifikasi

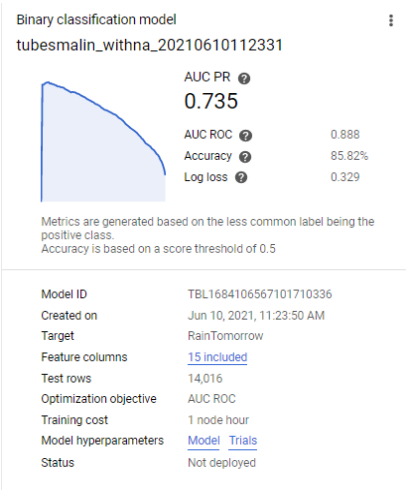
Setelah sistem melakukan train selama beberapa waktu, maka akan muncul hasil model terbaik untuk setiap dataset.

a. Dataset dengan melakukan drop data NaN



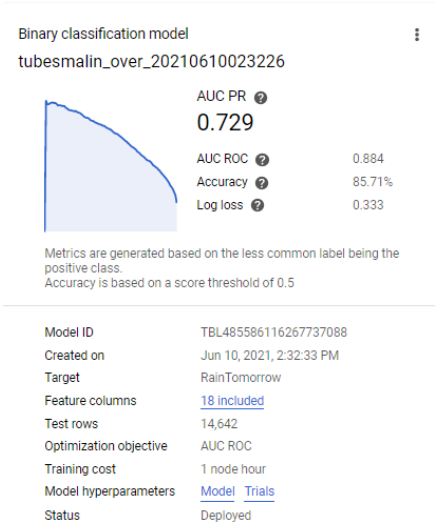
Pada dataset dengan melakukan drop NaN, dapat diketahui bahwa model ini menghasilkan akurasi sebesar 86,69%, nilai AUC ROC 0,0901, log loss 0,312, dan nilai AUC 0,766.

b. Dataset dengan tidak melakukan drop data NaN



Pada dataset dengan melakukan drop NaN, dapat diketahui bahwa model ini menghasilkan akurasi sebesar 85,82%, nilai AUC ROC 0,888, log loss 0,329, dan nilai AUC 0,735.

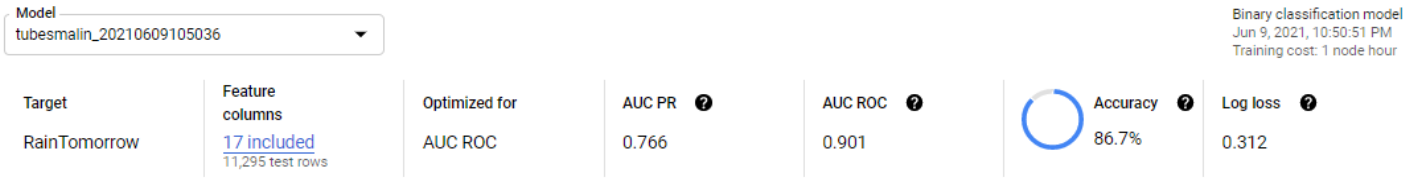
c. Dataset dengan melakukan preprocessing untuk dataset oversampling



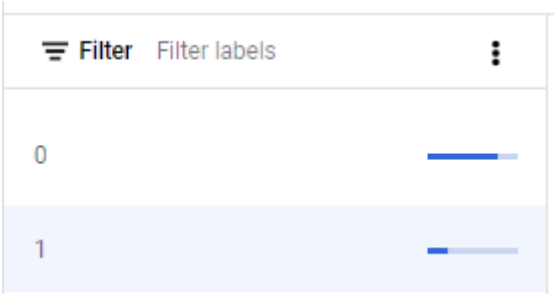
Pada dataset dengan melakukan drop NaN, dapat diketahui bahwa model ini menghasilkan akurasi sebesar 85,71%, nilai AUC ROC 0,884, log loss 0,333, dan nilai AUC 0,729.

5. Mengevaluasi Model

a. Dataset dengan melakukan drop data NaN

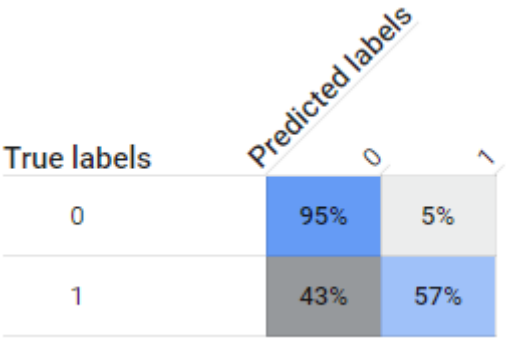


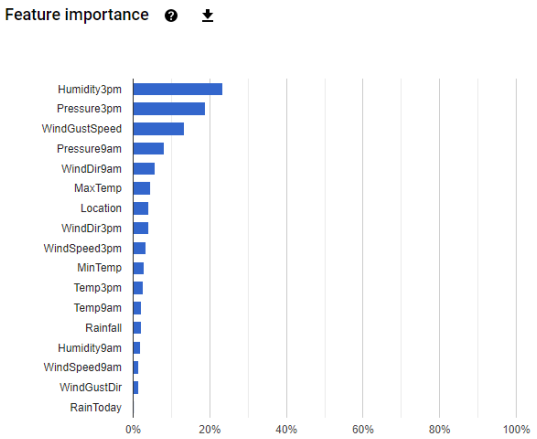
Pada dataset tipe pertama, didapatkan akurasi sebesar 86,7% dengan log loss sebesar 0,312. Selain itu didapatkan pula informasi bahwa dataset yang diproses merupakan tipe data Binary Classifier yang artinya memiliki dua kelas atau label target. Dengan jumlah label kelas 0 lebih banyak dari pada label kelas 1.



F1 score	0.656
Accuracy	86.7% (9,792/11,295)
Precision	77.6% (1,435/1,850)
True positive rate (Recall)	56.9% (1,435/2,523)
False positive rate	0.047 (415/8,772)

Pada dataset pertama, didapatkan juga nilai precision 77,6% dan Recall 56,9%.

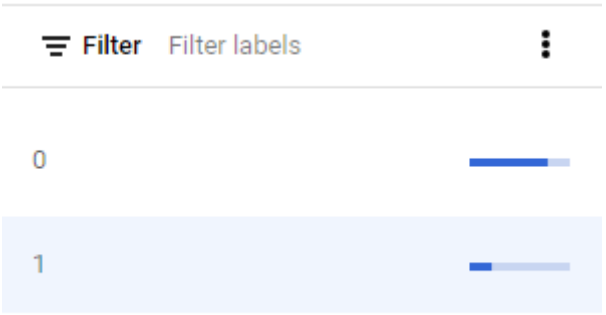




Berikut adalah tabel dari hasil confusion matrix untuk dataset pertama. Selain itu, terdapat juga visualisasi dari persebaran data Feature Importance, yang mana menunjukkan hasil pada kolom Humidity3pm dengan ukuran lebih dari 20%.

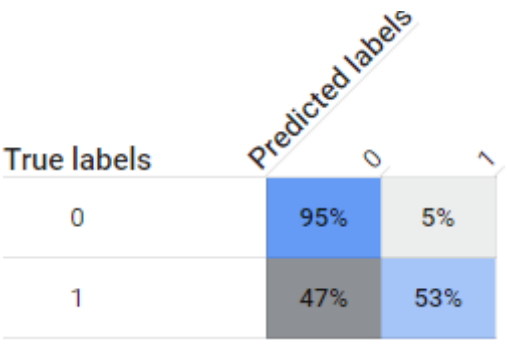
b. Dataset dengan tidak melakukan drop data NaN

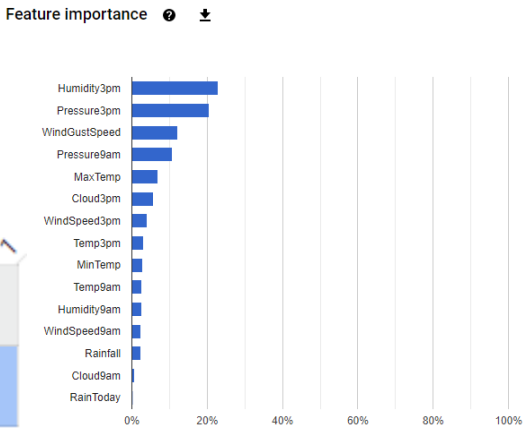
Pada dataset tipe pertama, didapatkan akurasi sebesar 865,8% dengan log loss sebesar 0,329. Selain itu didapatkan pula informasi bahwa dataset yang diproses merupakan tipe data Binary Classifier yang artinya memiliki dua kelas atau label target. Dengan jumlah label kelas 0 lebih banyak dari pada label kelas 1.



F1 score	0.622
Accuracy	85.8% (12,029/14,016)
Precision	75.6% (1,637/2,165)
True positive rate (Recall)	52.9% (1,637/3,096)
False positive rate	0.048 (528/10,920)

Pada dataset pertama, didapatkan juga nilai precision 75,6% dan





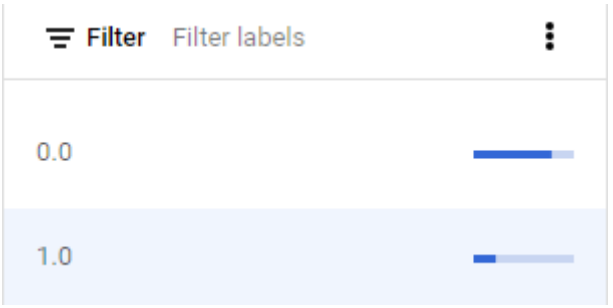
Recall 52,9%.

Berikut adalah tabel dari hasil confusion matrix untuk dataset pertama.

Selain itu, terdapat juga visualisasi dari persebaran data Feature Importance, yang mana menunjukkan hasil pada kolom Humidity3pm dengan ukuran lebih dari 20%.

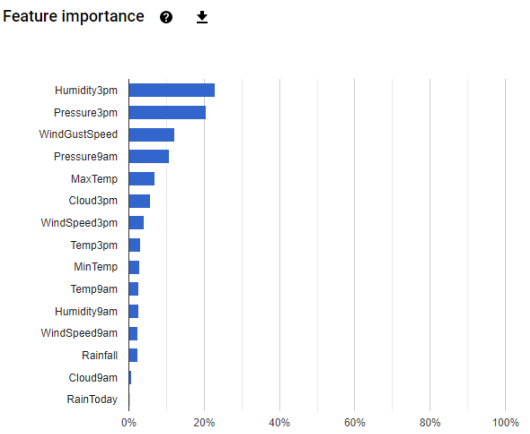
c. Dataset dengan melakukan preprocessing untuk dataset oversampling

Pada dataset tipe pertama, didapatkan akurasi sebesar 85,7% dengan log loss sebesar 0,333. Selain itu didapatkan pula informasi bahwa dataset yang diproses merupakan tipe data Binary Classifier yang artinya memiliki dua kelas atau label target. Dengan jumlah label kelas 0 lebih banyak dari pada label kelas 1.



F1 score	0.622
Accuracy	85.7% (12,549/14,642)
Precision	75.1% (1,722/2,293)
True positive rate (Recall)	53.1% (1,722/3,244)
False positive rate	0.050 (571/11,398)

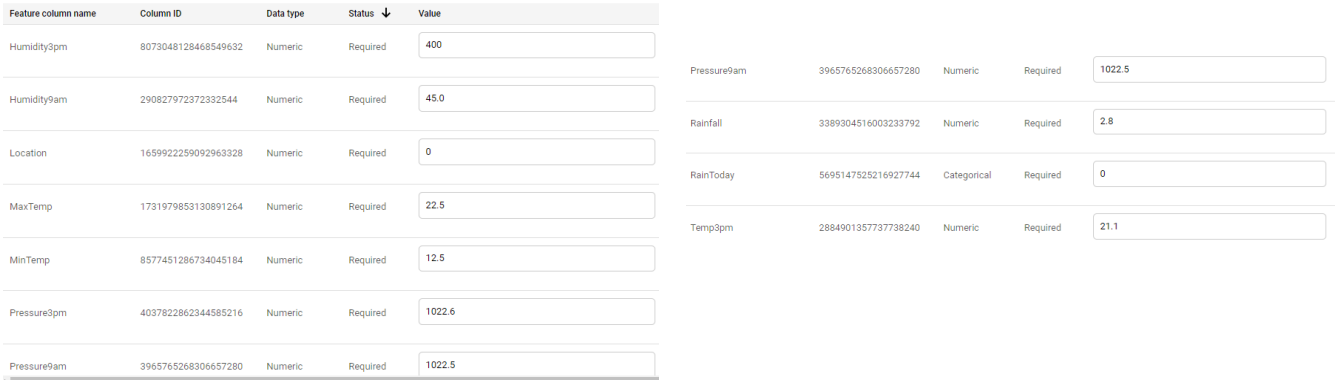
Pada dataset pertama, didapatkan juga nilai precision 75,1% dan Recall 53,1%. Berikut adalah tabel dari hasil confusion matrix untuk dataset pertama. Selain itu, terdapat juga visualisasi dari persebaran data Feature Importance, yang mana menunjukkan hasil pada kolom Humidity3pm dengan ukuran lebih dari 20%.



6. Melakukan Test dan Prediksi Model

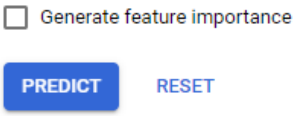
Pada tahapan ini, dilakukan model prediction menggunakan model train yang telah disimpan dan di-deploy sebelumnya. Tahapan penggunaannya adalah sebagai berikut:

a. Inputkan seluruh value secara random pada seluruh feature yang tersedia.



Feature column name	Column ID	Data type	Status	Value
Humidity3pm	8073048128468549632	Numeric	Required	400
Humidity9am	290827972372332544	Numeric	Required	45.0
Location	1659922259092963328	Numeric	Required	0
MaxTemp	1731979853130891264	Numeric	Required	22.5
MinTemp	8577451286734045184	Numeric	Required	12.5
Pressure3pm	4037822862344585216	Numeric	Required	1022.6
Pressure9am	3965765268306657280	Numeric	Required	1022.5
Pressure9am	3965765268306657280	Numeric	Required	1022.5
Rainfall	3389304516003233792	Numeric	Required	2.8
RainToday	5695147525216927744	Categorical	Required	0
Temp3pm	2884901357737738240	Numeric	Required	21.1

b. Setelah mengisi seluruh value yang tersedia, maka tekan button “predict”



c. Maka hasil prediksi dapat dilihat pada tampilan seperti berikut:

Dari hasil prediksi ini dapat diketahui bahkan value yang sebelumnya telah diinputkan akan menghasilkan nilai prediksi 1 yang artinya iya besok akan hujan dengan nilai *confidence score* 0,89 atau 89% akurat.

3.4 Hasil Pengujian dan Analisis

3.4.1 Hasil Pengujian TPOT

Hasil dari ketiga skenario yang telah dilakukan :

	Skenario 1 (Drop NaN)	Skenario 2 (Tidak Drop NaN)	Skenario 3 (Oversampling)
Precision	0.81	0.81	0.78
Recall	0.76	0.73	0.77
F1-Score	0.78	0.73	0.77
Accuracy	0.857	0.856	0.847

3.4.2 Hasil Pengujian AutoML Table

Hasil dari ketiga skenario yang telah dilakukan :

	Skenario 1 (Drop NaN)	Skenario 2 (Tidak Drop NaN)	Skenario 3 (Oversampling)
Precision	0,776	0,756	0.751
Recall	0,569	0,529	0.531
F1-Score	0,656	0,622	0.622
Accuracy	0,867	0,858	0.857

Dari 2 tabel diatas, dapat diketahui bahwa model terbaik diperoleh oleh skenario 1 dengan melakukan drop NaN dan diproses menggunakan AutoML Table dengan nilai akurasi 86,7%. Akurasi ini juga tidak jauh berbeda dengan hasil yang diperoleh dari skenario 1 pada TPOT yang memperoleh hasil akurasi 85,7%, dengan selisih hanya 1% antara hasil dari TPOT dan hasil dari AutoML Table. Maka dari 3 skenario yang sudah dilakukan di 2 tools AutoML, didapatkan hasil paling terbaik pada skenario dengan melakukan drop atau membuang nilai *missing value* atau nilai NaN.

4. KESIMPULAN

4.1 Kesimpulan

Dari pengujian dua tools AutoML yaitu TPOT dan AutoML Table, didapatkan kesimpulan bahwa TPOT dan AutoML Table memiliki beberapa perbedaan baik itu cara penggunaan, kelebihan dan kekurangan, maupun tingkat efisiensi dari hasil yang diperoleh. Perbedaan kedua tools tersebut adalah sebagai berikut:

No	TPOT	AutoML Table
1.	Memberikan informasi algoritma *machine learning* dan parameter tuning terbaik yang menghasilkan hasil paling optimal terhadap dataset diinputkan.	Tidak memberikan informasi algoritma *machine learning* dan parameter tuning terbaik yang menghasilkan hasil paling optimal terhadap dataset diinputkan.
2.	Semakin banyak generasi dan populasi yang kita tentukan pada TPOT, maka akan semakin banyak pula percobaan yang akan dilakukannya dalam mencari algoritma yang paling baik untuk dataset.	Semakin lama waktu pemrosesan AutoML yang kita tentukan pada AutoML yaitu berupa *node hours*, maka akan semakin lama pula percobaan yang akan dilakukannya dalam mencari algoritma yang paling baik untuk dataset.
3.	Tidak dapat melakukan preprocessing data secara otomatis. Akan lebih optimal jika memproses secara manual menggunakan tahapan-tahapan preprocessing dataset terlebih dahulu sendiri, kemudian memproses dataset yang sudah bersih tersebut menggunakan TPOT untuk memberikan gambaran Algoritma terbaik untuk dataset yang kita punya.	Dapat melakukan preprocessing secara otomatis tanpa harus melakukan tahapan-tahapan preprocessing terlebih dahulu. Namun, akan lebih optimal jika melakukan label encoding dahulu terhadap dataset yang akan diproses, karena beberapa kasus menunjukkan jika tidak melakukan label encoding atau merubah dataset kategorik menjadi numerik, maka akan memperlambat AutoML dalam menemukan algoritma terbaiknya.
4.	TPOT tidak dapat melakukan visualisasi hasil persebaran data dan evaluasi secara otomatis, hanya menampilkan algoritma dan parameter tuning terbaiknya saja.	AutoML Table dapat melakukan visualisasi hasil persebaran data dan evaluasi secara lengkap dan jelas, namun tidak dapat menampilkan algoritma dan parameter tuning terbaiknya.

5.	Melakukan model tes validasi dan prediksi secara manual.	Dapat melakukan model tes validasi dan prediksi secara otomatis hanya dengan memasukkan nilai value secara random ke dalam kolom-kolom yang telah disediakan

4.2 Saran

1. Menggunakan tools AutoML sesuai dengan kebutuhan dan rencana penelitian selanjutnya. Apabila ingin mengetahui algoritma dan parameter tuning terbaik guna untuk mendukung penelitian selanjutnya, maka disarankan untuk menggunakan AutoML tools yaitu TPOT. Selain itu, apabila membutuhkan informasi tentang persebaran data, visualisasi evaluasi model, dan juga test model secara otomatis, maka AutoML Table dapat menjadi pilihan yang tepat, karena memiliki kelebihan di sisi visualisasi dan keakuratan informasi dari dataset yang diproses.

2. Melakukan teknik preprocessing terlebih dahulu sebelum menggunakan AutoML Tools, dengan tujuan untuk mempermudah proses pencarian algoritma dan mengurangi kemungkinan kegagalan dalam proses prediksi algoritma dan parameter tuning.

3. Penulis berharap dapat melanjutkan dan memperbaiki teknik-teknik m*achine learning solution* yang lebih baik lagi dari sebelumnya sehingga dapat memberikan data analisis yang lengkap dan akurat.