Perbandingan Algoritma Klasifikasi Random Forest dan Extreme Gradient Boosting pada Dataset Cuaca Provinsi DKI Jakarta Tahun 2018

1st Ahmadien Hafizh Yusufi
S1 Informatika
Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran
ahmadienhy@upnvj.ac.id

2nd Fransisco Ready Permana
S1 Informatika
Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran
fransiscorp@upnvj.ac.id

3rd Raihan Kemmy Rachmansyah
S1 Informatika
Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran
raikemmy@upnvj.ac.id

4th Agung Hot Iman
S1 Informatika
Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran
agunghi@upnvj.ac.id

5th Gito Putro Wardana
S1 Informatika
Fakultas Ilmu Komputer
Universitas Pembangunan Nasional Veteran
gitopw@upnvj.ac.id

Abstrak. Perkembangan teknologi semakin berkembang pesat yang menyebabkan seluruh aktivitas kehidupan manusia tidak terlepas dari penggunakan teknologi. Peradaban yang semakin maju membuat banyak pihak ingin memanfaatkan ilmu teknologi untuk mempermudah aktivitas sehari-hari. Teknologi machine learning *merupakan teknologi yang mengembangkan kecerdasan buatan agar dapat belajar dengan sendirinya dengan menerapkan ilmu seperti statistika, matematika dan data* mining*. Pada penelitian ini penulis ingin membandingkan performa algoritma* Random Forest *dan* Extreme Gradient Boosting *pada data cuaca Provinsi DKI Jakarta tahun 2018. Setelah melakukan tahap pra-proses,* exploratory data analysis*,* resampling *pada label, dan melatih data untuk mendapatkan hasil akurasi yang nantinya digunakan sebagai pembanding algoritma* Extreme Gradient Boosting *dengan* Random Forest*. Berdasarkan hasil penelitian, penulis menyimpulkan bahwa algoritma* Random Forest *lebih baik daripada* Extreme Gradient Boosting *jika digunakan pada data cuaca Provinsi DKI Jakarta tahun 2018 karena waktu pemrosesan yang lebih cepat dan akurasi yang cukup tinggi yaitu 68%.*

Kata Kunci: Machine Learning, *Klasifikasi*, Random Forest, Extreme Gradient Boosting, *Cuaca*

I. PENDAHULUAN

Saat ini perkembembangan ilmu teknologi dan informasi menjadi semakin cepat. Hal ini mendorong berbagai pihak untuk membuat sebuah sistem cerdas yang dapat membantu kegiatan manusia. Salah penerapan dalam membuat sebuah sistem yang cerdas dengan menerapkan *machine learning*. *Machine learning* merupakan teknologi yang bertujuan untuk mengembangkan mesin agar dapat belajar dengan sendirinya. *Machine learning* menerapkan beberapa ilmu seperti statistika, matematika serta data *mining* untuk mengembangkan mesin agar dapat menganalisa suatu data tanpa harus di program secara spesifik.

Machine learning memiliki dua pendekatan yaitu *supervised learning* dan *unsupervised learning* yang keduanya bertujuan untuk menghasilkan wawasan dari sebuah data berdasarkan kecerdasan buatan yang telah di program. Salah satu pendekatan yang dapat ditangani oleh *supervised learning* adalah klasifikasi. Klasifikasi merupakan metode yang bertujuan untuk mengelompokan suatu data berdasarkan data yang telah dilatih sebelumnya. Beberapa contoh algoritma pada klasifikasi diantaranya adalah *Random Forest, Support Vector Machine, Artificial Neural Network, Classification Tree, K-Nearest Neighbor, Extreme Gradient Boosting* dan lain-lain. Selain itu, terdapat istilah *ensemble learning* pada *machine learning* yang menggunakan kombinasi dari beberapa algoritma untuk mendapatkan hasil yang lebih baik dengan tiga metode yaitu *bagging, boosting* dan *stacking*.

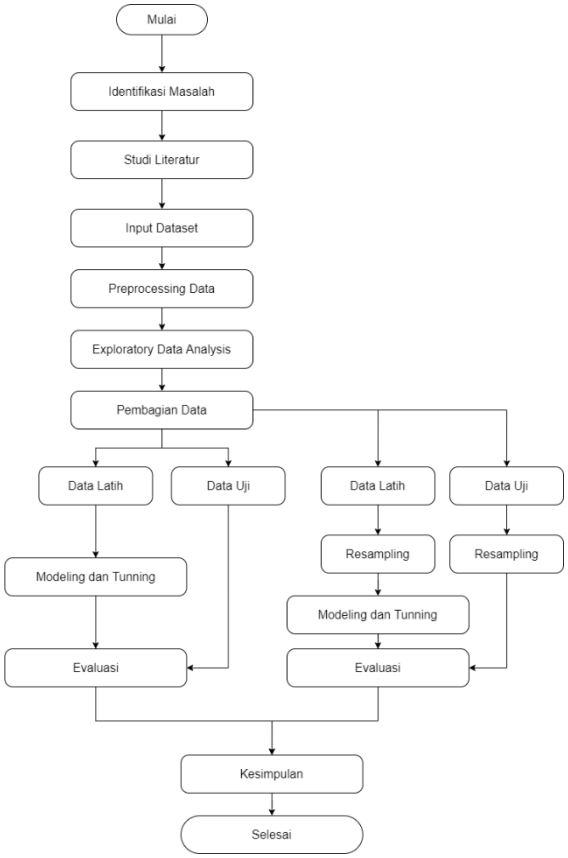
Salah satu algoritma yang menerapkan *ensemble learning* adalah *Random Forest*. Algoritma ini merupakan salah satu dari metode klasifikasi yang bertujuan untuk mengelompokkan data dengan jumlah yang besar. *Random Forest* menerapkan salah satu metode dari *ensemble learning* yang dibangun dari banyak pohon keputusan (*decision tree*) dengan metode *bagging* yang menggabungkan banyak nilai dugaan menjadi satu nilai dengan melakukan pendugaan gabungan berdasarkan k buah *tree*.

Selain *Random Forest*, algoritma *Extreme Gradient Boosting* juga menerapkan *ensemble learning* dan salah satu algoritma klasifikasi yang bertujuan untuk mengelompokkan data dengan jumlah yang besar. *Extreme Gradient Boosting* atau XGB dibangun dengan memanfaatkan konsep *ensemble learning* berdasarkan pohon keputusan (*decision tree*) dengan menerapkan metode *boosting*. Metode *boosting* merupakan sebuah metode yang digunakan untuk meningkatkan performa dari suatu algoritma *weak learner* yang bertujuan untuk mengurangi bias dengan cara membuat pohon yang lemah secara berurutan sehingga setiap pohon baru bisa berfokus kelemahan pohon sebelumnya.

Pada penelitian kali ini peneliti ingin membandingkan hasil akurasi algoritma *Random Forest* dan *Extreme Gradient Boosting* berdasarkan klasifikasi cuaca Provinsi DKI Jakarta pada tahun 2018. Peneliti mengambil penelitian ini karena cuaca pada provinsi tersebut tidak menentu yang mengakibatkan banyak kegiatan terganggu akibat perubahan cuaca secara mendadak. Peneliti ingin membuat

sebuah model serta menemukan algoritma yang lebih baik diantara *Random Forest* dengan *Extreme Gradient Boosting* pada data cuaca Provinsi DKI Jakarta tahun 2018.

II. METODE PENELITIAN



Gambar. 1. Tahapan Metode Penelitian

2.1 Identifikasi Masalah

Pada tahapan identifikasi masalah, peneliti membandingkan hasil akurasi algoritma *random forest* dan *extreme gradient boosting* berdasarkan cuaca pada Provinsi DKI Jakarta tahun 2018. Peneliti menggunakan *random forest* dan *extreme gradient boosting* karena algoritma tersebut sesuai untuk digunakan pada data yang besar dan memiliki persebaran data secara acak. Peneliti menemukan bahwa algoritma *random forest* dan *extreme gradient boosting* sedang ramai dibicarakan pada komunitas karena performanya yang baik serta menghasilkan akurasi yang tinggi di sebagian besar dataset. Oleh karena itu, peneliti ingin membuktikan kebenaranya dengan menggunakan sebuah data aktual yaitu data cuaca provinsi DKI Jakarta pada tahun 2018

2.2 Studi Literatur

Pada tahapan ini peneliti melakukan studi literatur untuk mendapatkan teori-teori yang valid terkait dengan penelitian yang dilakukan sebelumnya. Sumber literatur yang digunakan untuk dijadikan referensi antara lain jurnal, artikel, dan situs internet.

2.3 Input Dataset

Data yang digunakan untuk penelitian ini diambil dari open dataset Indonesia pada website https://katalog.data.go.id/dataset/data-prakiraan-cuaca-wilayah-provinsi-dki-jakarta-tahun-2018. Data tersebut diambil pada setiap kota administrasi yang ada di DKI Jakarta setiap 4 kali dalam sehari selama satu tahun berdasarkan kelembaban udara serta suhu kota tersebut pada waktu pengambilan data. Data ini memiliki jumlah 6 kolom dan 8535 baris.

2.4 *Preprocessing* Data

Tahapan selanjutnya, peneliti melakukan pra-proses pada data yang telah peneliti peroleh. Tahapan pra-proses yang peneliti lakukan disini bertujuan untuk membersihkan data dari derau (*noise)*, melakukan *polynomial feature* pada data, serta melakukan integrasi pada data. Berikut adalah beberapa hal yang peneliti lakukan saat melakukan tahapan *preprocessing*, yaitu:

2.4.1 Memproses *Missing Value*

Salah satu noise yang terdapat pada data ini adalah missing value. Melihat apakah terdapat data yang kosong dan menangani data kosong tersebut dengan cara menghapus data tersebut menggunakan fungsi dropna(). Setelah dilakukan penghapusan data pada missing value, dapat di cek kembali untuk melihat apakah masih ada missing value atau tidak.

2.4.2 Memproses Kolom Cuaca dan Waktu

Tahapan ini bertujuan untuk membersihkan dan mengelompokkan data pada kolom cuaca dan waktu agar data lebih mudah untuk diproses pada tahap selanjutnya. Pada kolom cuaca akan mengelompokkan data menjadi 3 yaitu Cerah, Hujan dan Berawan sedangkan pada kolom waktu mengelompokkan data menjadi 4 yaitu Pagi, Siang, Malam dan Dini Hari. Proses ini dilakukan menggunakan fungsi *Regular Expression*.

2.4.3 Memproses Kolom kelembaban_persen dan suhu_derajat_celcius

Tahapan ini bertujuan untuk membagi data pada kolom kelembaban_persen dan suhu_derajat_celcius menjadi 2 bagian yaitu data minimal dan maksimal. Proses ini menggunakan fungsi split() untuk membagi data lalu disimpan kedalam variabel kelembaban_min dan suhu_min untuk data minimal serta kelembaban_max dan suhu_max untuk data bernilai maksimal berdasarkan kolom kelembaban_persen dan suhu_derajat_celcius.

2.4.4 Memperbaiki Tipe Data

Selanjutnya, kolom dataset yang masih memiliki tipe data objek tetapi memiliki nilai numerik atau angka akan diubah

menjadi bertipe data integer, untuk memudahkan dalam melakukan proses modelling. Fitur-fitur yang diubah menjadi integer yaitu, kelembaban_min, suhu_min, kelembaban_max, dan suhu_max.

2.4.5 Menghapus Kolom

Tahapan terakhir pada pra-proses ini adalah memilih fitur yang akan digunakan dengan cara menghapus kolom yang tidak mempengaruhi proses klasifikasi menggunakan fungsi .drop(). Beberapa kolom yang dihapus dan tidak digunakan yaitu tanggal, kelembaban_persen, dan suhu_derajat_celcius.

2.5 *Exploratory Data Analysis*

Setelah dilakukan pra-proses dilanjutkan dengan melihat serta menganalisis data yang telah diperoleh untuk membuat model yang maksimal. Ada beberapa hal yang dilakukan pada tahapan ini yaitu:

2.5.1 Melihat Jumlah Total Baris dan Kolom

Melihat jumlah baris dan kolom dari dataset menggunakan fungsi .shape yang menghasilkan jumlah 6 kolom dan 8535 baris dengan tujuan untuk melihat seberapa besar data yang dipunya.

2.5.2 Melihat Tipe Data

Selanjutnya dilakukan kembali pengecekan terhadap tipe data pada setiap kolom yang belum sesuai menggunakan fungsi .dtypes() agar data lebih mudah diproses pada tahap berikutnya.

2.5.3 Melihat Statistik Deskriptif

Tahapan berikutnya adalah dengan melihat statistik deskriptif pada setiap kolom untuk melihat ringkasan dari data secara keseluruhan menggunakan fungsi .describe(). Fungsi ini dapat memberikan informasi mengenai nilai rata-rata, standar deviasi dan interquartile pada setiap kolom.

2.5.4 Melihat Bentuk Persebaran Data

Tahap selanjutnya adalah melihat grafis visualisasi data yang bertujuan untuk menganalisis data agar lebih mudah dipahami. Proses visualisasi data yang peneliti lakukan menggunakan package Seaborn lalu memanggil fungsi .pairplot() yang nantinya akan menghasilkan grafis visualisasi dari data cuaca Provinsi DKI Jakarta tahun 2018.

2.6 Pembagian Data

Tahapan setelah pra-proses dan *exploratory data analysis* adalah pembagian data menjadi dua yaitu data latih dan data uji dengan perbandingan 80% data latih dan 20% data uji. Selanjutnya akan dipisahkan antara kolom label dan kolom fitur. Pada kolom label dilakukan pembagian secara stratifikasi atau pembagian data secara seimbang karena label yang digunakan bertipe kategorikal.

2.7 Resampling Data Menggunakan Teknik SMOTE

Setelah tahapan pembagian data, peneliti mengecek jumlah data pada kolom label dan menemukan bahwa label pada dataset yang peneliti gunakan tidak seimbang dengan perbandingan imbalance ratio sebesar 45% sehingga diperlukan proses untuk menyeimbangkan jumlah label terlebih dahulu menggunakan metode resampling dengan teknik oversampling yaitu SMOTE. Resampling adalah teknik untuk memanipulasi kelas distribusi dengan memperbaiki data latih sehingga didapatkan kelas yang seimbang [1]. *Oversampling* adalah salah satu teknik resampling yang bertujuan untuk memanipulasi data dengan membuat data baru pada data minoritas sehingga jumlahnya akan seimbang dengan jumlah data mayoritas [2]. Metode yang populer diterapkan untuk menangani permasalahan keseimbangan data adalah SMOTE. SMOTE atau Synthetic Minority Over-sampling Technique adalah metode pembangkitan data minoritas sebanyak data mayoritas dengan SMOTE yang bekerja dengan cara mengambil secara acak tetangga terdekat sebanyak k dari setiap instance dalam kelas minoritas kemudian membuat instance baru (sintetis) antara instance tersebut dengan tetangga terdekat k yang dipilih secara acak. Dengan pendekatan SMOTE maka dapat dipastikan tidak terjadi masalah duplikasi data sehingga lebih kebal terhadap masalah overfitting [1].

2.8 *Preprocessing*, *Modeling,* dan *Tuning*

2.8.1 Preprocessing

Sebelum tahapan selanjutnya, peneliti mempersiapkan data terlebih dahulu agar lebih siap untuk di proses. Peneliti melakukan tahapan pra-proses kembali setelah tahap pembagian data. Hal ini peneliti lakukan untuk menghindari kebocoran data sehingga akan mengurangi tingkat overfitting ataupun underfitting. Pada tahapan pra-proses kali ini, peneliti menggunakan *ColumnTransformer* dengan membagi kolom menjadi kolom numerik dan kolom kategorik. Untuk kolom numerik dilakukan pra-proses berupa PolynomialFeature untuk sedangkan kolom kategorik dilakukan pra-proses berupa OneHotEncoder dengan tujuan untuk membangun model yang sederhana yang dapat digunakan untuk pemahaman dan melakukan prediksi.

2.8.1.1 Column Transformer

ColumnTransformer adalah sebuah package dari scikitlearn yang berperan untuk membagi dan melakukan pra-proses pada kolom numerik dan kategorik. Peneliti melakukan tahap pra-proses kembali setelah pembagian data dengan tujuan untuk menghindari kebocoran data sehingga dapat mengurangi tingkat overfitting ataupun underfitting.

2.8.1.2 Polynomial Feature

PolynomialFeature adalah sebuah teknik untuk mengatasi data yang tidak linier dengan menambahkan kekuatan pada setiap fitur sebagai fitur yang baru. PolyomialFeature akan melatih model linier pada kumpulan fitur yang telah diperluas [3]. Teknik ini akan membuat fitur baru menjadi bentuk eksponensial sebagai contoh misalnya kita memiliki sebuah fitur dengan nilai x, maka polynomial feature akan membuat fitur baru yang bernilai x2

2.8.1.3 OneHotEncoder
One-Hot Encoding adalah salah satu metode encoding yang merepresentasikan data bertipe kategori sebagai vektor biner dengan nilai integer, 0 dan 1, dimana semua elemen akan bernilai 0 kecuali satu elemen yang bernilai 1, yaitu elemen yang memiliki nilai kategori tersebut.

2.8.2 Modeling

Setelah membuat ColumnTransformer, peneliti melatih data dengan algoritma yang ingin peneliti bandingkan yaitu Random Forest dan Extreme Gradient Boosting. Kemudian peneliti membuat Pipeline untuk menggabungkan ColumnTransformer yang telah dibuat sebelumnya dengan jenis algoritma yang akan peneliti gunakan untuk proses tuning nantinya.

2.8.2.1 Pipeline

Pipeline adalah sebuah metode yang dapat mengatur aliran data *input* dan *output* dari sebuah model atau kumpulan beberapa model *Machine Learning*. Pipeline dapat memproses data mulai dari *input* data mentah, fitur, output, model *Machine Learning* dan parameter model, dan output prediksi.

2.8.2.2 *Random Forest*

Random forest merupakan algoritma yang baik digunakan untuk melakukan klasifikasi data dalam jumlah yang besar. Random forest akan melakukan suatu penggabungan data dengan decision tree serta untuk melakukan proses seleksi sehingga pada saat penentuan klasifikasi pada decision tree, algoritma ini akan melakukan pemecahan terhadap jenis fitur yang ada dalam dataset. Decision tree akan melakukan prediksi secara acak dengan tujuan untuk menghasilkan akurasi yang maksimal.

2.8.2.3 Extreme Gradient Boosting

Extreme Gradient Boosting adalah metode boosting dengan cara menggabungkan kumpulan pohon keputusan yang akan digunakan untuk pembangunan pohon selanjutnya. Model ini dibangun dengan cara membuat model yang baru berdasarkan model yang lama dengan tujuan untuk menghasilkan kesalahan prediksi (error) yang lebih kecil dari model sebelumnya [4].

2.8.3 *Tuning*

Pembuatan *Column Transformer* dan Pipeline bertujuan untuk mempermudah proses dan memprediksi data baru dengan model telah dibuat serta memungkinkan kita dalam melakukan tuning hyperparameter berdasarkan algoritma yang ada di dalam pipeline. Tuning hyperparameter sendiri merupakan proses untuk menemukan nilai hyperparameter terbaik yang sesuai dengan data yang dimiliki sehingga dapat menghasilkan model dengan performa yang terbaik [5].

2.9 Evaluasi

Setelah semua tahapan selesai, selanjutnya akan dilakukan tahapan evaluasi performa dari model Random Forest dan Extreme Gradient Boosting menggunakan data uji. Evaluasi model bertujuan untuk membuat estimasi generalisasi error dari model yang dipilih, dan menguji seberapa baik kinerja model tersebut pada data baru yaitu data uji. Model machine learning yang baik adalah model yang tidak hanya bekerja dengan baik pada data latih, tapi juga pada data uji. Peneliti menggunakan *confusion matrix* untuk mengetahui efisiensi kinerja model agar dapat diketahui seberapa banyak data yang salah diklasifikasi pada data uji dan *classification report* untuk mengukur tingkat akurasi, *presisi*, dan *recall*.

III. HASIL DAN PEMBAHASAN

Penelitian ini menggunakan data cuaca Provinsi DKI Jakarta pada tahun 2018 yang berasal dari situs katalog.data.go.id. Pada awalnya, data yang peneliti dapatkan masih terpisah berdasarkan bulan kemudian digabungkan menjadi satu file. Dataset yang peneliti gunakan memiliki jumlah 8535 baris dan 6 kolom. Penjelasan lebih lanjut mengenai atribut dalam dataset cuaca Provinsi DKI Jakarta Tahun 2018 dapat dilihat pada tabel 1.

Tabel. 1. Atribut variabel dataset.

tanggal	Tanggal pengambilan data pada Provinsi DKI Jakarta tahun 2018
waktu	Waktu pengambilan data pada saat Pagi, Siang, Malam dan lain sebagainya
cuaca	Cuaca hasil pengamatan Provinsi DKI Jakarta tahun 2018 pada tanggal tersebut
kelembaban_persen	Tingkat kelembapan udara pada wilayah tersebut ketika pengamatan data cuaca dilakukan
suhu_derajat_celcius	Suhu udara pada wilayah tersebut ketika pengamatan data cuaca dilakukan

3.1 Pre-processing Data

Tahap selanjutnya peneliti melakukan pra-proses pada data yang telah peneliti peroleh. Tahapan pra-proses yang peneliti lakukan bertujuan untuk membersihkan data (*data cleaning*). Berikut merupakan beberapa hal yang peneliti lakukan saat melakukan pra-proses, yaitu:

3.1.1 Memproses Missing Value

Tahap pertama, peneliti mengecek apakah terdapat data yang kosong atau tidak, pada tahap ini ditemukan bahwa data yang peneliti gunakan terdapat noise atau derau berupa missing value yang jumlahnya tidak terlalu banyak. Peneliti memutuskan untuk menghapus baris yang memiliki missing value tersebut dengan fungsi dropna(). Setelah itu peneliti melakukan pengecekan kembali apakah masih terdapat missing value menggunakan fungsi .isna().sum().

3.1.2 Memproses Kolom cuaca dan waktu

Setelah menghapus missing value, peneliti menemukan banyak nilai dari kolom cuaca dan waktu yang memiliki kesalahan penulisan. Dalam menangani hal tersebut, peneliti memakai fungsi *regular expression* untuk memproses permasalahan pada kolom cuaca dan waktu. Selain kesalahan penulisan, kolom cuaca dan waktu juga memiliki nilai yang sangat luas dan beragam.

Pada kolom cuaca terdiri dari 'Hujan Lokal', 'Hujan Ringan', 'Berawan', 'Cerah Berawan', 'Cerah', 'Berawan Tebal', 'Hujan Sedang', 'Cerah Berawn', 'Cerang Berawan', 'Beawan', 'Berawan ', 'Hujan Petir', 'Hujan Lokal ','Cerah Berawan ', 'Cerah ', 'Cerah Berawah', 'Cerah ', ' Berawan', 'Hujan', 'Hujan Petir ', 'Hujan Sedang ', 'Cerah berawan', 'Hujan Ringanl', 'Berawa', 'Hujang Sedang', 'Hujan Loka', dan 'Hujan Ringan ' sedangkan pada kolom waktu terdiri dari ‘Siang’, ‘Pagi’, ‘Dini Hari’, ‘Malam’, ‘siang’, ‘pagi’, ‘malam’ dan ‘dini hari’. Dari data yang sangat luas dan beragam ini peneliti mengelompokkan data kolom cuaca menjadi 3 yaitu Cerah, Hujan, dan Berawan dan data pada kolom waktu menjadi 4 yaitu Siang, Pagi, Malam, dan Dini Hari menggunakan bantuan fungsi *regular expression*.

3.1.3 Memproses kolom kelembaban_persen dan suhu_derajat_celcius

Peneliti juga melakukan pemrosesan terhadap kolom kelembaban_persen dan suhu_derajat_celcius. Peneliti melakukan splitting terhadap tanda (-) dan membagi data berdasarkan nilai sebelum dan sesudah tanda (-). Kemudian membuat kolom baru yaitu kolom minimal untuk menampung nilai sebelum tanda (-) dan kolom maksimal untuk menampung nilai setelah tanda (-) sehingga terbentuk 4 kolom baru yaitu suhu_max, suhu_min, kelembaban_max, dan kelembaban_min berdasarkan kolom kelembaban_persen dan suhu_derajat_celcius.

3.1.3 Memperbaiki Tipe Data

Tahap selanjutnya peneliti memperbaiki tipe data dari setiap kolom yang belum sesuai. Peneliti menemukan bahwa semua kolom bertipe data objek termasuk kolom numerik yang akan diproses seperti suhu_max, suhu_min, kelembaban_max, dan kelembaban_min sehingga diperlukan perbaikan tipe data menjadi integer agar kolom tersebut dapat diolah dan diproses untuk tahap selanjutnya. Fungsi yang digunakan untuk melihat tipe data setiap kolom adalah fungsi .dtypes() dan untuk mengubah tipe data dari setiap kolom yang tidak sesuai peneliti menggunakan fungsi .astype(). Berikut adalah hasil yang peneliti dapatkan dari fungsi .dtypes() sebelum dan setelah dilakukan pengubahan tipe data:

```
tanggal                object
wilayah                object
waktu                  object
cuaca                  object
kelembaban_persen      object
suhu_derajat_celcius   object
kelembaban_min          int64
kelembaban_max          int64
suhu_min                int64
suhu_max                int64
dtype: object
```

Gambar 2. Sebelum perbaikan tipe data

```
tanggal                object
wilayah                object
waktu                  object
cuaca                  object
kelembaban_persen      object
suhu_derajat_celcius   object
kelembaban_min         object
kelembaban_max         object
suhu_min               object
suhu_max               object
dtype: object
```

Gambar 3. Sesudah perbaikan tipe data

3.1.3 Menghapus Kolom

Tahapan terakhir pada tahap pra-proses adalah memakai fungsi .drop() untuk menyaring serta membuang kolom yang tidak digunakan untuk tahapan selanjutnya. Kolom yang dibuang pada tahap ini adalah kolom tanggal, kelembaban_persen, dan suhu_derajat_celcius.

3.2 Exploratory Data Analysis

Setelah tahapan pra proses, dilanjutkan tahapan *Exploratory Data Analysis* dengan tujuan melihat data, menganalisis data, dan menghasilkan performa model yang maksimal. Ada beberapa hal yang peneliti lakukan pada tahapan ini yaitu:

3.2.1 Melihat Jumlah Total Baris dan Kolom

Langkah pertama yang peneliti lakukan pada tahapan EDA adalah melakukan pengecekan terhadap jumlah total baris dan kolom. Hal ini peneliti lakukan untuk mengetahui seberapa banyak data yang dimiliki oleh dataset cuaca

Provinsi DKI Jakarta pada tahun 2018 yang akan dilatih setelah dilakukannya tahapan pra-proses. Untuk mengetahui jumlah baris dan kolom peneliti menggunakan fungsi .shape(). Berdasarkan fungsi tersebut, didapatkan jumlah awal data peneliti sebanyak 8538 baris dan 6 kolom sedangkan setelah dilakukan tahap pra-proses berjumlah 8390 baris dan 6 kolom.

3.2.2 Melihat Statistik Deskriptif

Tahap selanjutnya adalah melihat statistik seperti nilai maksimum, nilai minimum, jumlah, standar deviasi, median, kuartil pertama pada kolom numerik dan nilai unique serta frekuensi dari kolom kategorik. Selain untuk melihat statistik dari data, tahapan ini juga bertujuan untuk melihat apakah ada anomali dari data yang perlu untuk diperbaiki. Untuk melakukan tahapan ini, peneliti menggunakan fungsi .describe() dan berikut adalah hasil dari proses yang peneliti dapatkan:

	cuaca	waktu	wilayah	tanggal
count	8398	8398	8398	8398
unique	3	4	6	348
top	Cerah	Siang	Jakarta Utara	2018-01-02
freq	3796	2100	1400	48

Gambar. 4. Statistik pada data kategorik

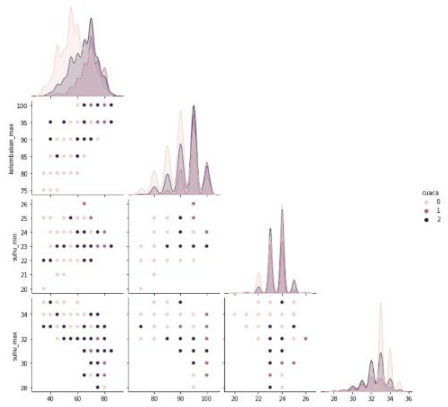
	kelembaban_min	kelembaban_max	suhu_min	suhu_max
count	8398.000000	8398.000000	8398.000000	8398.000000
mean	62.254704	91.727792	23.548464	32.608597
std	10.592668	5.562658	0.775947	1.246190
min	35.000000	75.000000	20.000000	28.000000
25%	55.000000	90.000000	23.000000	32.000000
50%	65.000000	95.000000	24.000000	33.000000
75%	70.000000	95.000000	24.000000	33.000000
max	85.000000	100.000000	26.000000	35.000000

Gambar. 5. Statistik pada data numerik

3.2.3 Melihat Bentuk Persebaran Data

Tahapan selanjutnya pada *Exploratory Data Analysis* adalah membuat visualisasi persebaran data. Tujuan dibuat visualisasi persebaran data tidak hanya untuk melihat bentuk persebaran dari data yang dimiliki, tetapi juga untuk melihat korelasi dari setiap kolom yang ada pada data tersebut. Selain itu, dengan dibuatnya visualisasi persebaran data, dapat memberikan wawasan kepada peneliti mengenai algoritma apa yang sesuai untuk digunakan pada data yang peneliti miliki.

Peneliti menggunakan bantuin fungsi .pairplot() dari *package seaborn* dalam pembuatan visualisasi persebaran data. Berdasarkan hasil dari visualisasi tersebut, peneliti menemukan bahwa data peneliti memiliki persebaran yang cukup acak. Oleh karena itu, penggunaan algoritma *Extreme Gradient Boosting* dan *Random Forest* adalah pilihan yang baik untuk menangani kasus tersebut. Berikut merupakan persebaran dari data yang peneliti:



Gambar. 6. Bentuk Persebaran Data serta korelasi antar tiap kolom

3.3 Pembagian Data

Data yang telah dilakukan tahap pra-proses dan *Exploratory Data Analysis* kemudian akan dibagi menjadi data latih dan juga data uji dengan perbandingan 80% data latih dan 20% data uji atau sebanyak 6718 baris data latih dan 1688 baris menjadi data uji. Peneliti memilih kolom cuaca sebagai label yang akan diklasifikasi serta menerapkan pembagian secara stratified atau pembagian data dengan seimbang kepada kolom cuaca.

3.4 Resampling Data Menggunakan Teknik SMOTE

Setelah melakukan pembagian data, peneliti menemukan bahwa data yang peneliti miliki tidak seimbang. Berdasarkan fungsi .value_counts() didapatkan jumlah data pada label Cerah 3796, Berawan 2896 dan Hujan 1706. Secara matematis, imbalanced data dapat dihitung menggunakan rumus

imbalance_ratio = data minority / data majority x100%
imbalance_ratio = 1706 / 3796 x 100%
imbalance_ratio = 45%

Berdasarkan perhitungan diatas, label Cerah merupakan data majority dan label Hujan merupakan data minority sehingga didapatkan perbandingan imbalance ratio sebesar 45%. Hasil tersebut cukup rendah sebagai data yang imbalance sehingga dapat dikategorikan sebagai mild pada derajat imbalance data.

Untuk menangani hal tersebut, peneliti mencoba menyeimbangkan data menggunakan metode resampling dengan teknik oversampling memakai SMOTE. Teknik resampling bertujuan untuk menyeimbangkan data berdasarkan data majority yaitu Cuaca, hal ini menyebabkan bertambahnya data sehingga jumlah data menjadi 9108 baris

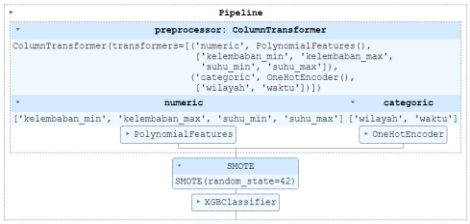
untuk data latih dan 2280 baris untuk data uji. Dalam penelitian ini, peneliti ingin mengetahui perbedaan performa suatu algoritma dengan ataupun tanpa metode resampling. Oleh karena itu, peneliti melakukan modeling untuk melatih dan menguji dengan ataupun tanpa teknik oversampling SMOTE.

3.5 Modeling

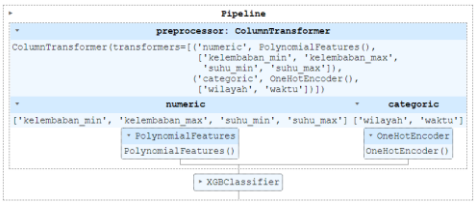
Modeling merupakan tahapan terakhir dalam memproses sebuah data. Pada tahap ini, peneliti membuat sebuah Pipeline yang didalamnya terdapat ColumnTransformer, teknik SMOTE, dan algoritma yang akan dibandingkan yaitu *Extreme Gradient Boosting*, dan *Random Forest* untuk mempermudah dalam memproses serta memprediksi data baru dengan model telah dibuat. Sebelum membuat sebuah Pipeline, peneliti mempersiapkan *ColumnTransformer* terlebih dahulu untuk memproses data numerik dan kategorik. Berikut tahapan yang peneliti lakukan menggunakan *ColumnTransformer* yaitu:

1. Tahap pertama, peneliti membagi kolom menjadi numerik dan kategorik, lalu menyimpannya ke dalam variabel numerik untuk data numerik dan kategorik untuk data kategorikal.
2. Tahap kedua, memanggil metode *ColumnTransformer* dari package sklearn.
3. Tahap ketiga, menerapkan *PolynomialFeature* pada kolom numerik untuk mengatasi data yang persebarannya tidak linier.
4. Tahap terakhir yaitu menerapkan *OneHotEncoding* pada kolom kategorik sebagai vektor biner yang bernilai integer 0 dan 1.

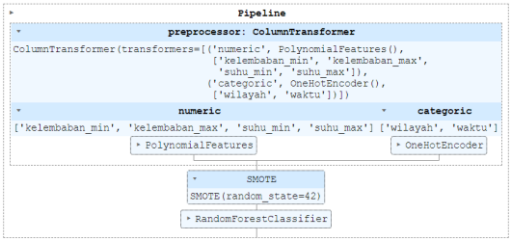
Setelah membuat *ColumnTransformer* dilanjutkan dengan menerapkan teknik SMOTE. Dalam penerapan teknik SMOTE peneliti menyiapkan dua skenario yaitu dengan ataupun tanpa SMOTE untuk membandingkan hasil yang akan didapat. Berikut merupakan isi dari Pipeline dengan ataupun tanpa menerapkan teknik SMOTE:



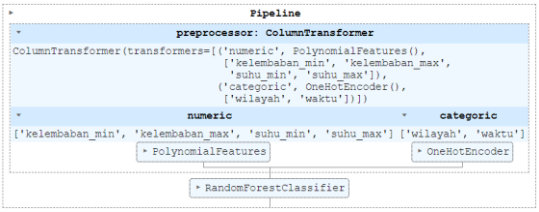
Gambar. 7. Pipeline *Extreme Gradient Boosting* dengan SMOTE



Gambar. 8. Pipeline *Extreme Gradient Boosting* tanpa SMOTE



Gambar. 9. Pipeline *Random Forest* dengan SMOTE



Gambar. 11. Pipeline *Random Forest* tanpa SMOTE

Selain memudahkan dalam memprediksi data, penggunaan Pipeline juga dapat membantu dalam proses *tuning hyperparameter* untuk memaksimalkan akurasi dari algoritma yang peneliti gunakan. Peneliti menggunakan salah satu algoritma *tuning hyperparameter* yaitu RandomSearchCV dengan mengatur nilai cv sebanyak 3, n_iter sebanyak 500, Pipeline, dan *hyperparameter* yang akan di *tuning*. Nilai dari *hyperparameter* yang akan di *tuning* adalah sebagai berikut:

```
parameters = {
    'algo__max_depth': Integer(low=1, high=10),
    'algo__learning_rate': Real(low=-2, high=0, prior='log-uniform'),
    'algo__n_estimators': Integer(low=100, high=200),
    'algo__subsample': Real(low=0.3, high=0.8, prior='uniform'),
    'algo__gamma': Integer(low=1, high=10),
    'algo__colsample_bytree': Real(low=0.1, high=1, prior='uniform'),
    'algo__reg_alpha': Real(low=-3, high=1, prior='log-uniform'),
    'algo__reg_lambda': Real(low=-3, high=1, prior='log-uniform'),
    'prep__numeric__degree': Integer(low=1, high=3),
    'prep__numeric__interaction_only': [True, False]
}
```

Gambar. 12. Parameter XGBoost tanpa SMOTE

```
from jcopml.tuning.space import Integer, Real

parameter_smote = {
    'algo__max_depth': Integer(low=1, high=10),
    'algo__learning_rate': Real(low=-2, high=0, prior='log-uniform'),
    'algo__n_estimators': Integer(low=100, high=200),
    'algo__subsample': Real(low=0.3, high=0.8, prior='uniform'),
    'algo__gamma': Integer(low=1, high=10),
    'algo__colsample_bytree': Real(low=0.1, high=1, prior='uniform'),
    'algo__reg_alpha': Real(low=-3, high=1, prior='log-uniform'),
    'algo__reg_lambda': Real(low=-3, high=1, prior='log-uniform'),
    'prep__numeric__degree': Integer(low=1, high=3),
    'prep__numeric__interaction_only': [True, False],
    'smote__k_neighbors': [3, 5, 7],
    'smote__sampling_strategy': ['minority', 'majority', 'not minority', 'not majority'],
}
```

Gambar. 13. Parameter XGBoost dengan SMOTE

```
from jcopml.tuning.space import Integer, Real

parameter_smote = {
    'algo__n_estimators': Integer(low=100, high=200),
    'algo__max_depth': Integer(low=20, high=80),
    'algo__max_features': Real(low=0.1, high=1, prior='uniform'),
    'algo__min_samples_leaf': Integer(low=1, high=20),
    'prep__numeric__degree': [1, 2, 3],
    'prep__numeric__interaction_only': [True, False],
    'smote__k_neighbors': [3, 5, 7],
    'smote__sampling_strategy': ['minority', 'majority', 'not minority', 'not majority'],
}
```

Gambar. 14. Parameter RF tanpa SMOTE

```
from jcopml.tuning.space import Integer, Real

parameters = {
    'algo__n_estimators': Integer(low=100, high=200),
    'algo__max_depth': Integer(low=20, high=80),
    'algo__max_features': Real(low=0.1, high=1, prior='uniform'),
    'algo__min_samples_leaf': Integer(low=1, high=20),
    'prep__numeric__degree': [1, 2, 3],
    'prep__numeric__interaction_only': [True, False]
}
```

Gambar. 15. Parameter RF dengan SMOTE

3.6 Evaluasi

Setelah tahap mencari parameter yang terbaik, dilanjutkan dengan mengevaluasi model akhir pada data uji untuk menilai kinerja dari model yang telah dibuat menggunakan data baru. Tahapan ini akan menghasilkan parameter terbaik berdasarkan tuning hyperparameter, nilai confusion matrix, akurasi, presisi serta recall dari data uji.

3.6.1 *Extreme Gradient Boosting* tanpa SMOTE

Setelah dilakukan *modeling* dan *tuning,* parameter terbaik yang didapatkan dari algoritma *Extreme Gradient Boosting* tanpa SMOTE adalah sebagai berikut:

Tabel. 2. Nilai hypermeter *Extreme Gradient Boosting* tanpa SMOTE terbaik yang didapat setelah *tuning*

Tuning Hyperparameter Algoritma Extreme Gradient Boosting		**Tuning Hyperparameter Metode PolynomialFeature**	
colsample_bytree	0.95547794732758	degree	2
gamma	2	interaction_only	True
learning_rate	0.088266166600734		
max_depth	9		
n_estimators	100		
reg_alpha	0.011907916572279		
reg_lambda	0.042450971980559		
subsample	0.745740007811175		

Selain mendapatkan parameter terbaik, pada tahap evaluasi peneliti juga mendapatkan nilai akurasi dari algoritma Extreme Gradient Boosting tanpa SMOTE sebesar 69% pada data uji dan 76% pada data latih. Berikut merupakan hasil dari classification report yang peneliti dapatkan:

	precision	recall	f1-score	support
0	0.80	0.84	0.82	3036
1	0.78	0.71	0.74	1365
2	0.71	0.70	0.70	2317
accuracy			0.76	6718
macro avg	0.76	0.75	0.75	6718
weighted avg	0.76	0.76	0.76	6718

	precision	recall	f1-score	support
0	0.76	0.79	0.77	760
1	0.66	0.55	0.60	341
2	0.62	0.65	0.64	579
accuracy			0.69	1680
macro avg	0.68	0.66	0.67	1680
weighted avg	0.69	0.69	0.69	1680

Gambar. 16. Performa *Extreme Gradient Boosting* tanpa SMOTE

3.6.2 *Extreme Gradient Boosting* dengan SMOTE

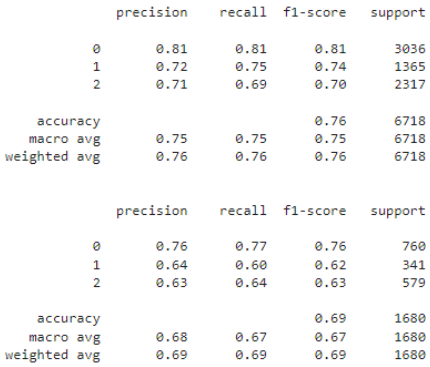
Setelah dilakukan *modeling* dan *tuning,* parameter terbaik yang didapatkan dari algoritma *Extreme Gradient Boosting* menggunakan SMOTE adalah sebagai berikut:

Tabel. 3. Nilai hypermeter *Extreme Gradient Boosting* dengan SMOTE terbaik yang didapat setelah *tuning*

Tuning Hyperparameter Algoritma Extreme Gradient Boosting		**Tuning Hyperparameter Metode Polynomial Feature**		**Tuning Hyperparameter Metode SMOTE**	
colsample_bytree	0.5274558687373737	degree	2	k_neighbors	7
gamma	2	interaction_only	True	sampling_strategy	not majority
learning_rate	0.034444852940 51				
max_depth	10				
n_estimators	131				
reg_alpha	0.004676967546 81				

reg_lambda	0.003882417663 71
subsample	0.7865383989561 4

Berdasarkan classification report, peneliti mendapatkan nilai akurasi dari algoritma Extreme Gradient Boosting menggunakan SMOTE sebesar 69% pada data uji dan 76% pada data latih. Berikut merupakan hasil dari classification report yang peneliti dapatkan:



	precision	recall	f1-score	support
0	0.81	0.81	0.81	3036
1	0.72	0.75	0.74	1365
2	0.71	0.69	0.70	2317
accuracy			0.76	6718
macro avg	0.75	0.75	0.75	6718
weighted avg	0.76	0.76	0.76	6718

	precision	recall	f1-score	support
0	0.76	0.77	0.76	760
1	0.64	0.60	0.62	341
2	0.63	0.64	0.63	579
accuracy			0.69	1680
macro avg	0.68	0.67	0.67	1680
weighted avg	0.69	0.69	0.69	1680

Gambar. 17. Performa *Extreme Gradient Boosting* dengan SMOTE

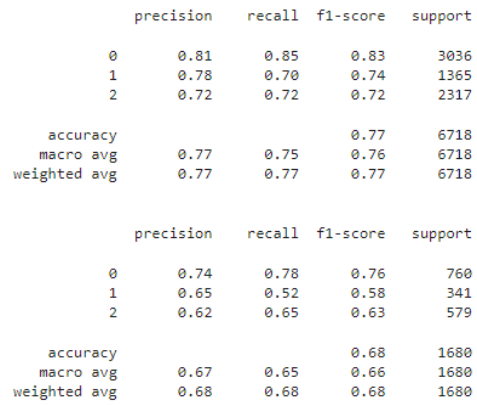
3.6.3 *Random Forest* Tanpa SMOTE

Setelah dilakukan modeling dan tuning, parameter terbaik yang didapatkan dari algoritma Random Forest tanpa SMOTE adalah sebagai berikut:

Tabel. 4. Nilai hypermeter *Extreme Gradient Boosting* tanpa SMOTE terbaik yang didapat setelah *tuning*

Tuning Hyperparameter Algoritma Random Forest		**Tuning Hyperparameter Metode PolynomialFeature**	
max_depth	39	degree	3
max_features	0.4578349581 6651	interaction_only	True
min_samples_leaf	5		
n_estimators	102		

Berdasarkan classification report, peneliti mendapatkan nilai akurasi dari algoritma Random Forest tanpa SMOTE sebesar 68% pada data uji dan 77% pada data latih. Berikut merupakan hasil dari classification report yang peneliti dapatkan:



	precision	recall	f1-score	support
0	0.81	0.85	0.83	3036
1	0.78	0.70	0.74	1365
2	0.72	0.72	0.72	2317
accuracy			0.77	6718
macro avg	0.77	0.75	0.76	6718
weighted avg	0.77	0.77	0.77	6718

	precision	recall	f1-score	support
0	0.74	0.78	0.76	760
1	0.65	0.52	0.58	341
2	0.62	0.65	0.63	579
accuracy			0.68	1680
macro avg	0.67	0.65	0.66	1680
weighted avg	0.68	0.68	0.68	1680

Gambar. 18. Performa *Random Forest* tanpa SMOTE

3.6.4 *Random Forest* dengan SMOTE

Setelah dilakukan modeling dan tuning, parameter terbaik yang didapatkan dari algoritma Random Forest menggunakan SMOTE adalah sebagai berikut:

Tabel. 5. Nilai hypermeter *Extreme Gradient Boosting* tanpa SMOTE terbaik yang didapat setelah *tuning*

Tuning Hyperparameter Algoritma Random Forest		**Tuning Hyperparameter Metode Polynomial Feature**		**Tuning Hyperparameter Metode SMOTE**	
max_depth	68	degree	3	k_neighbor	3
max_features	0.3384059025 1306	interaction_only	True	sampling_strategy	not minority
min_samples_leaf	5				
n_estimators	196				

Berdasarkan classification report, peneliti mendapatkan nilai akurasi dari algoritma Random Forest menggunakan SMOTE sebesar 67% pada data uji dan 77% pada data latih. Berikut merupakan hasil dari classification report yang peneliti dapatkan:

	precision	recall	f1-score	support
0	0.83	0.82	0.82	3036
1	0.79	0.67	0.73	1365
2	0.69	0.77	0.73	2317
accuracy			0.77	6718
macro avg	0.77	0.75	0.76	6718
weighted avg	0.78	0.77	0.77	6718

	precision	recall	f1-score	support
0	0.77	0.73	0.75	760
1	0.65	0.50	0.57	341
2	0.58	0.69	0.63	579
accuracy			0.67	1680
macro avg	0.67	0.64	0.65	1680
weighted avg	0.68	0.67	0.67	1680

Gambar. 19. Performa *Random Forest* dengan SMOTE

3.6 Perbandingan Metode Klasifikasi

Berdasarkan hasil evaluasi dari keempat metode klasifikasi yaitu *Extreme Gradient Boosting* Tanpa SMOTE, *Extreme Gradient Boosting* dengan SMOTE, Random Forest dengan SMOTE, dan Random Forest tanpa SMOTE didapatkan hasil sebagai berikut:

Tabel. 6. Hasil akurasi dari setiap model

	***Extreme Gradient Boosting* Tanpa SMOTE**	***Extreme Gradient Boosting* dengan SMOTE**	**Random Forest tanpa SMOTE**	**Random Forest dengan SMOTE**
Akurasi data uji	69.226%	69.047%	68.035%	67.261%
Akurasi data latih	76.495%	75.796%	77.284%	77.225%

Berdasarkan penelitian yang telah dilakukan, peneliti mendapatkan bahwa algoritma Extreme Gradient Boosting memiliki performa yang lebih baik daripada Random Forest jika digunakan pada data cuaca Provinsi DKI Jakarta tahun 2018. Algoritma Extreme Gradient Boosting menghasilkan nilai akurasi sebesar 69% sedangkan Random Forest sebesar 68%. Berdasarkan penelitian peneliti, jika menggunakan teknik resampling SMOTE menghasilkan data yang lebih seimbang namun tidak memberikan pengaruh yang signifikan terhadap performa model karena tingkat *imbalance* dari data yang peneliti miliki tergolong sangat rendah yaitu 45%. Pada algoritma Extreme Gradient Boosting, penggunaan teknik resampling tidak menambah tingkat akurasi pada model tetapi membuat model menjadi lebih seimbang dan tidak overfitting. Sedangkan pada algoritma Random Forest penggunaan teknik resampling bahkan menurunkan tingkat akurasi model dari 68% menjadi 67%,

Selain itu, melalui penelitian ini terdapat hal yang cukup penting yaitu waktu yang diperlukan pada saat melakukan pelatihan pada data cuaca Provinsi DKI Jakarta tahun 2018. Waktu yang dibutuhkan oleh algoritma Extreme Gradient Boosting untuk melakukan pelatihan data jauh lebih lama dibandingkan dengan waktu yang dibutuhkan oleh Random Forest. Berdasarkan penelitian peneliti, algoritma Extreme Gradient Boosting membutuhkan waktu kurang lebih 15 menit sedangkan Random Forest tidak lebih dari 10 menit.

IV. HASIL DAN PEMBAHASAN

4.1 KESIMPULAN

Dengan mempertimbangkan kelebihan dan kekurangan pada masing-masing algoritma, dapat disimpulkan bahwa algoritma Random Forest lebih baik daripada Extreme Gradient Boosting karena waktu yang dibutuhkan Random Forest lebih cepat dan juga akurasi yang dihasilkan Random Forest tidak berbeda jauh dengan Extreme Gradient Boosting yaitu sekitar 1 - 2% saja. Hal ini membuat peneliti memilih Random Forest sebagai algoritma yang lebih efektif digunakan pada data cuaca Provinsi DKI Jakarta tahun 2018. Harapannya penelitian berikutnya dapat menemukan algoritma yang lebih efektif serta lebih baik dari Random Forest jika digunakan pada data cuaca Provinsi DKI Jakarta tahun 2018 dari segala hal.

4.2 Saran

Saran dari peneliti adalah karena tingkat akurasi dari model yang peneliti buat masih di angka 67% sampai 69% maka, perlu dilakukannya peninjauan kembali mengenai *feature engineering* maupun teknik pra proses lainnya terhadap data yang peneliti miliki sehingga bisa mendapatkan tingkat akurasi yang lebih tinggi daripada yang telah peneliti lakukan.

Referensi

[1] Y. A. Sir and A. H. H. Soepranoto, "Pendekatan Resampling Data Untuk Menangani Masalah Ketidakseimbangan Kelas," *Jurnal Komputer dan Informatika*, vol. 10, no. 1, pp. 31–38, Mar. 2022, doi: 10.35508/jicon.v10i1.6554.

[2] R. Mohammed, J. Rawashdeh, and M. Abdullah, "Machine Learning with Oversampling and Undersampling Techniques: Overview Study and Experimental Results," in *2020 11th International Conference on Information and Communication Systems, ICICS 2020*, Apr. 2020, pp. 243–248. doi: 10.1109/ICICS49469.2020.239556.

[3] R. Manorathna, "Polynomial Regression with a Machine Learning Pipeline," 2020. [Online]. Available: https://www.researchgate.net/publication/344616388

[4] G. A. Mursianto, "Perbandingan Metode Klasifikasi Random Forest dan XGBoost Serta Implementasi Teknik SMOTE pada Kasus Prediksi Hujan," 2021.

[5] M. Radhi, S. Hamonangan Sinurat, D. Ryan Hamonangan Sitompul, E. Indra, and S. Informasi, "PREDIKSI WATER QUALITY INDEX (WQI) MENGGUNAKAN ALGORITMA REGRESSI DENGAN HYPER-PARAMETER TUNING," *Jurnal Sistem Informasi dan Ilmu Komputer Prima)*, vol. 5, no. 1, 2021.