Laporan Final Project Pembelajaran Mesin

Water Quality Dataset

Kelompok 4 :

Surya Fadli Alamsyah	5025221059
Mohmmad Hanif Furqan Aufa Putra	5025221161
Mochammad Zharif Asyam Marzuqi	5025221163

Dosen :
Dini Adni Navastara, S.Kom, M.Sc.

INSTITUT TEKNOLOGI SEPULUH NOPEMBER SURABAYA

2023/2024

DAFTAR ISI

BAB I
Pendahuluan

1.1. Latar Belakang

Krisis air bersih global adalah masalah serius yang mempengaruhi sekitar 2,2 miliar orang yang tidak memiliki akses ke air minum yang aman. Situasi ini diperparah oleh polusi, perubahan iklim, dan pengelolaan sumber daya air yang tidak memadai, yang menyebabkan masalah kesehatan yang signifikan seperti diare, kolera, dan tifus. Krisis ini diperkirakan akan semakin parah, dengan antara dua hingga tiga miliar orang mengalami kekurangan air setidaknya satu bulan setiap tahun, yang menimbulkan risiko terhadap ketahanan pangan dan akses listrik, terutama di daerah perkotaan (Liu et al., 2024; Jones, 2024).

Penelitian ini bertujuan untuk memprediksi kualitas air dengan menganalisis parameter seperti nitrit, nitrat, amonium, dan fosfat. Dengan menggunakan teknik seperti oversampling dan feature selection, penelitian ini berusaha mengidentifikasi faktor-faktor kunci yang mempengaruhi kualitas air. Ini dapat membantu dalam mengembangkan strategi pengelolaan air yang lebih baik, yang pada gilirannya dapat mengurangi risiko penyakit yang ditularkan melalui air dan meningkatkan kesehatan serta kesejahteraan masyarakat.

Secara global, masalah kelangkaan air tidak hanya terjadi di wilayah seperti Gaza, di mana air mahal dan sering tidak aman, tetapi juga di Afrika sub-Sahara, di mana infrastruktur dan pemerintahan yang buruk, serta perubahan iklim, memperburuk masalah ini. Solusi seperti peningkatan pasokan air, penggunaan kembali air limbah, dan desalinasi sangat penting untuk keamanan air jangka panjang (UNRWA, 2024; WaterAid, 2023).

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah yang telah dijelaskan di atas maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana performa model machine learning dalam mengklasifikasikan kualitas air menggunakan berbagai teknik seperti oversampling, feature selection, dan hyperparameter tuning?
2. Model machine learning mana yang memberikan akurasi terbaik dalam prediksi kualitas air?
3. Bagaimana pengaruh oversampling menggunakan SMOTE terhadap performa model?
4. Sejauh mana teknik feature selection dan hyperparameter tuning dapat meningkatkan akurasi model?

1.3. Batasan Masalah

Beberapa batasan masalah untuk menghindari adanya penyimpangan maupun pelebaran pokok masalah agar penelitian tersebut lebih terarah dalam penelitian ini adalah sebagai berikut:

1. Penelitian ini hanya menggunakan beberapa model machine learning yaitu K-Nearest Neighbors (KNN), Decision Tree, Random Forest, XGBoost, dan Artificial Neural Network (ANN).
2. Teknik oversampling yang digunakan adalah Synthetic Minority Over-sampling Technique (SMOTE).
3. Feature selection dilakukan dengan metode tertentu yang tidak dijelaskan secara spesifik dalam penelitian ini.
4. Hyperparameter tuning dilakukan dengan cara yang tidak dijelaskan secara rinci dalam penelitian ini.
5. Data yang digunakan untuk pengujian adalah data kualitas air yang relevan dengan konteks penelitian ini.

1.4. Tujuan Penelitian

Beberapa tujuan penelitian ini adalah sebagai berikut :

1. Mengukur dan membandingkan performa akurasi model machine learning dalam mengklasifikasikan kualitas air tanpa dan dengan menggunakan teknik oversampling, feature selection, dan hyperparameter tuning.
2. Menentukan model machine learning yang paling optimal untuk prediksi kualitas air dalam berbagai skenario pengujian.
3. Menganalisis pengaruh oversampling, feature selection, dan hyperparameter tuning terhadap performa model machine learning.
4. Memberikan rekomendasi model terbaik dan teknik yang efektif dalam meningkatkan akurasi klasifikasi kualitas air.

1.5. Manfaat Penelitian

Beberapa manfaat yang dapat diperoleh dari dijalankannya penelitian ini adalah sebagai berikut:

1. Memberikan informasi yang bermanfaat bagi peneliti dan praktisi dalam memilih model machine learning yang paling efektif untuk prediksi kualitas air.
2. Membantu dalam memahami pengaruh teknik oversampling, feature selection, dan hyperparameter tuning terhadap performa model machine learning.
3. Memberikan dasar bagi pengembangan lebih lanjut dalam bidang analisis kualitas air menggunakan teknik machine learning.
4. Meningkatkan efisiensi dan akurasi dalam pemantauan dan pengelolaan kualitas air melalui penerapan model dan teknik yang tepat.

BAB II
Studi literatur

2.1. Dasar Theory

2.2. Penelitian Terkait

Kualitas air sangat penting bagi kesehatan masyarakat, dan banyak organisasi telah menetapkan pedoman untuk memastikan air minum yang aman. Organisasi Kesehatan Dunia (WHO) memberikan pedoman komprehensif yang merinci batas yang dapat diterima untuk kontaminan, yang berfungsi sebagai referensi global (WHO, 2022). Di Amerika Serikat, Badan Perlindungan Lingkungan (EPA) menetapkan standar yang ketat melalui Peraturan Air Minum Utama Nasional untuk memastikan air minum yang aman (EPA, 2023). Demikian pula, Petunjuk Air Minum Uni Eropa menguraikan kriteria untuk melindungi kesehatan manusia (EU, 2020), sementara pedoman Kanada mencakup perlindungan air minum dan kehidupan akuatik (CCME, 2024). Australia juga menekankan kualitas air melalui Pedoman Air Minum Australia (ADWG, 2023). Penelitian terbaru telah menggarisbawahi peran model prediksi canggih seperti Jaringan Syaraf Tiruan (JST) dalam pemantauan kualitas air, menyoroti kemampuan mereka untuk menangani data yang kompleks dan meningkatkan penilaian kualitas air (Chen dkk., 2020; Juahir dkk., 2020; Ahmed dkk., 2020).

Penelitian terbaru telah memajukan bidang prediksi kualitas air menggunakan teknik pembelajaran mesin. Duie Tien Bui dkk. (2020) mengembangkan algoritme pembelajaran mesin hibrida baru untuk prediksi Indeks Kualitas Air (Water Quality Index/WQI), yang menunjukkan bahwa algoritme BA-RT sangat efektif (Bui dkk., 2020). Selain itu, Lu dan Ma (2020) mengusulkan model berbasis pohon keputusan hibrida untuk prediksi kualitas air jangka pendek, yang mencapai akurasi yang unggul dengan model CEEMDAN-RF (Lu & Ma, 2020). Ahmed dkk. (2019) menunjukkan bahwa peningkatan gradien dan regresi polinomial dapat secara efisien memprediksi WQI di Pakistan (Ahmed dkk., 2019). Selain itu, Chen dkk. (2019) mengidentifikasi parameter air utama dan menemukan bahwa model seperti pohon keputusan, hutan acak, dan hutan riam dalam memberikan kinerja yang unggul dalam memprediksi tingkat kualitas air di Cina (Chen dkk., 2019). Zhu dkk. (2019) meninjau penerapan pembelajaran mesin dalam evaluasi kualitas air, dengan menekankan keefektifannya dalam memecahkan masalah nonlinier yang kompleks (Zhu dkk., 2019).

Secara kolektif, pedoman dan studi ini menyoroti upaya global untuk mengatur dan memantau kualitas air, memastikan keamanan bagi semua populasi. Pedoman ini memberikan kerangka kerja yang kuat untuk mengevaluasi dan mengelola kualitas air, yang penting untuk mencegah penyakit yang ditularkan melalui air dan meningkatkan kesehatan masyarakat. Studi lebih lanjut oleh Al-Sulttani dkk. (2019) tentang model ensembel untuk prediksi kualitas air permukaan dan Ghazali dan Ali (2022) tentang Analisis Komponen Utama (PCA) untuk pemodelan indeks kualitas air sungai terus menyempurnakan metodologi tersebut (Al-Sulttani dkk., 2019; Ghazali & Ali, 2022). Kemajuan ini menggarisbawahi pentingnya mengintegrasikan teknik pembelajaran mesin yang canggih dalam praktik pengelolaan kualitas air.

BAB III
Metodologi

3.1. Dataset

Dataset diperoleh dari Kaggle, sebuah platform daring terkemuka dalam berbagai kompetisi data dan sumber daya ilmiah. Dataset yang digunakan dalam penelitian ini berisi data tentang kualitas perairan di Eropa yang dapat diakses melalui tautan berikut:

https://www.kaggle.com/datasets/ozgurdogan646/water-quality-dataset/data

Deskripsi Dataset:

- Terdiri dari 20 ribu row dan 29 column data
- Dataset yang digunakan dalam penelitian ini berasal dari dua sumber:
  - Waterbase quality dari EEA
  - What Waste Global Database dari World Bank
- Deskripsi Fitur:
  - parameterWaterBodyCategory: Kategori badan air, sebagaimana didefinisikan dalam daftar kode (sumber: EEA).
  - observedPropertyDeterminandCode: Kode unik dari determinan yang dipantau, sebagaimana didefinisikan dalam daftar kode (sumber: EEA).
  - procedureAnalysedFraction: Spesifikasi fraksi sampel yang dianalisis (sumber: EEA).
  - procedureAnalysedMedia: Jenis media yang dipantau (sumber: EEA).
  - resultUom: Unit pengukuran untuk nilai yang dilaporkan (sumber: EEA).
  - phenomenonTimeReferenceYear: Tahun selama data dikumpulkan (sumber: EEA).
  - parameterSamplingPeriod: Periode tahun selama data digunakan untuk agregasi dikumpulkan (sumber: EEA).
  - resultMeanValue: Nilai rata-rata data yang digunakan untuk agregasi (sumber: EEA).
  - waterBodyIdentifier: Pengidentifikasi internasional unik dari badan air di mana data diperoleh (sumber: EEA).
  - Country: Informasi negara yang dihasilkan menggunakan koordinat.
  - PopulationDensity: Kepadatan penduduk negara.
  - TerraMarineProtected_2016_2018: Rata-rata wilayah laut terlindungi negara antara tahun 2016-2018.
  - TouristMean_1990_2020: Rata-rata jumlah wisatawan negara antara tahun 1990-2020.
  - VenueCount: Jumlah tempat di dekat koordinat yang diberikan.

- netMigration_2011_2018: Rata-rata migrasi negara antara tahun 2011-2018.
- literacyRate_2010_2018: Tingkat literasi negara antara tahun 2010-2018.
- combustibleRenewables_2009_2014: Jumlah energi terbarukan yang dapat terbakar di negara antara tahun 2009-2014.
- droughts_floods_temperature: Data kekeringan, banjir, dan suhu.
- gdp: Produk domestik bruto negara.
- composition_food_organic_waste_percent: Persentase limbah makanan organik.
- composition_glass_percent: Persentase limbah kaca.
- composition_metal_percent: Persentase limbah logam.
- composition_other_percent: Persentase limbah lainnya.
- composition_paper_cardboard_percent: Persentase limbah kertas dan kardus.
- composition_plastic_percent: Persentase limbah plastik.
- composition_rubber_leather_percent: Persentase limbah karet dan kulit.
- composition_wood_percent: Persentase limbah kayu.
- composition_yard_garden_green_waste_percent: Persentase limbah hijau taman dan halaman.
- waste_treatment_recycling_percent: Persentase pengolahan dan daur ulang limbah.

3.2. Data labeling

Kode beserta hasil labeling diadopsi dari EPA dengan batasan regulatori sebagai berikut:

```python
regulatory_limits = {
    'mg{NO2}/L': (0.0, 1.0),        # Nitrite
    'mg{NO3}/L': (0.0, 10.0),       # Nitrate
    'mg{NH4}/L': (0.0, 0.5),        # Ammonium
    'mg{PO4}/L': (0.0, 0.3),        # Phosphate
    'mg{CaCO3}/L': (0.0, 200.0),    # Calcium Carbonate (hardness)
    'mg{P}/L': (0.0, 0.1),          # Phosphorus
    'mg{N}/L': (0.0, 1.0),          # Nitrogen
    'mg{Si}/L': (0.0, 2.0),         # Silicon
    'mg{O2}/L': (5.0, 14.0),         # Dissolved Oxygen
    'uS/cm': (0.0, 1000.0),         # Conductivity
    # 'mg/L': (0.0, 1.0),             # General parameter, assuming generic pollutants
    'Cel': (0.0, 40.0),             # Temperature
    # 'mg{C}/L': (0.0, 2.0),          # Carbon
    'mg{NH3}/L': (0.0, 0.2),        # Ammonia
    '[pH]': (6.5, 8.5),             # pH value
}

# Function to label the data
def label_water_quality(row, limits):
    uom = row['resultUom']
    value = row['resultMeanValue']

    if uom not in limits:
        return 'undefined'
    elif isinstance(limits[uom], tuple):
        lower_bound, upper_bound = limits[uom]
        if value < lower_bound or value > upper_bound:
            return 'dirty'
        else:
            return 'clean'
    elif value > limits[uom]:
        return 'dirty'
    else:
        return 'clean'

# Apply the function to the DataFrame
df['water_quality'] = df.apply(label_water_quality, axis=1, limits=regulatory_limits)

# Print the DataFrame with the new column 'water_quality'
print(df[['resultUom', 'resultMeanValue', 'water_quality']])
```

Gambar 3.1 Labelling Air berdasarkan Uom

```
         resultUom  resultMeanValue water_quality
0         mg{NO2}/L         0.063310         clean
1         mg{NO2}/L         0.046733         clean
2       {massRatio}       132.859000     undefined
3         mg{NO3}/L        11.578376         dirty
4              mmol/L         0.206800     undefined
...               ...              ...           ...
19995     mg{NO2}/L         0.092466         clean
19996             %        89.908300     undefined
19997     mg{NO3}/L        18.901608         dirty
19998   {massRatio}       307.307000     undefined
19999          [pH]         7.954790         clean

[20000 rows x 3 columns]
```

Gambar 3.2 Hasil dari labelling berdasarkan Uom

```python
regulatory_limits = {
    'EEA_3131-01-9' : (90, 110),
    'EEA_3164-08-7' : (0,10000),
    'EEA_3164-07-6' : (0,10000),
    'EEA_3164-01-0' : (0,10),
    'EEA_3133-06-0' : (0,25)
}

# Function to label the data
def label_water_quality(row, limits):
    x = row['observedPropertyDeterminandCode']
    value = row['resultMeanValue']

    if x not in limits:
        return row['water_quality']
    elif isinstance(limits[x], tuple):
        lower_bound, upper_bound = limits[x]
        if value < lower_bound or value > upper_bound:
            return 'dirty'
        else:
            return 'clean'
    elif value > limits[x]:
        return 'dirty'
    else:
        return 'clean'

# Apply the function to the DataFrame
df['water_quality'] = df.apply(label_water_quality, axis=1, limits=regulatory_limits)

# Print the DataFrame with the new column 'water_quality'
print(df[['observedPropertyDeterminandCode', 'resultMeanValue', 'water_quality']])
```

Gambar 3.3 Labelling Air berdasarkan observedPropertyDeterminandCode

```
      observedPropertyDeterminandCode  resultMeanValue water_quality
0                       CAS_14797-65-0         0.063310         clean
1                       CAS_14797-65-0         0.046733         clean
2                        EEA_3164-07-6       132.859000         clean
3                       CAS_14797-55-8        11.578376         dirty
4                        EEA_3151-01-7         0.206800     undefined
...                                ...              ...           ...
19995                   CAS_14797-65-0         0.092466         clean
19996                    EEA_3131-01-9        89.908300         dirty
19997                   CAS_14797-55-8        18.901608         dirty
19998                    EEA_3164-08-7       307.307000         clean
19999                    EEA_3152-01-0         7.954790         clean

[20000 rows x 3 columns]
```

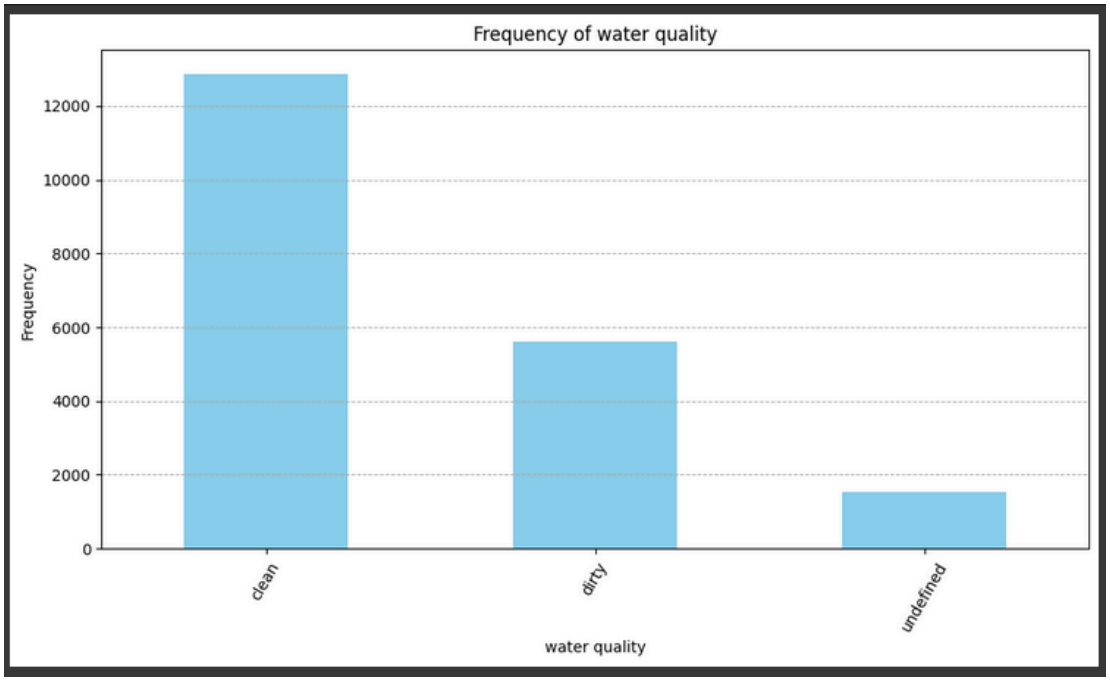
Gambar 3.4 Hasil dari labelling berdasarkan observedPropertyDeterminandCode

```python
df = df[df['water_quality'] != 'undefined'].copy()
```

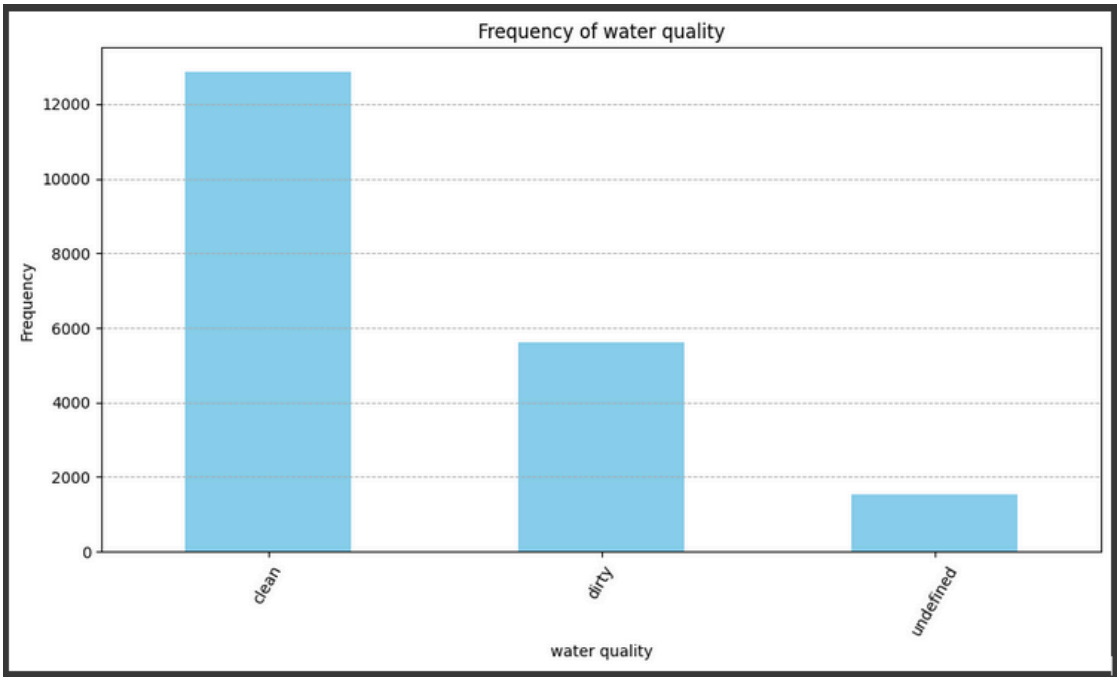
Gambar 3.5 Menghilangkan Air yang tidak bisa di define dari labelling yang berdasarkan Uom

3.3. Exploratory Data Analysis

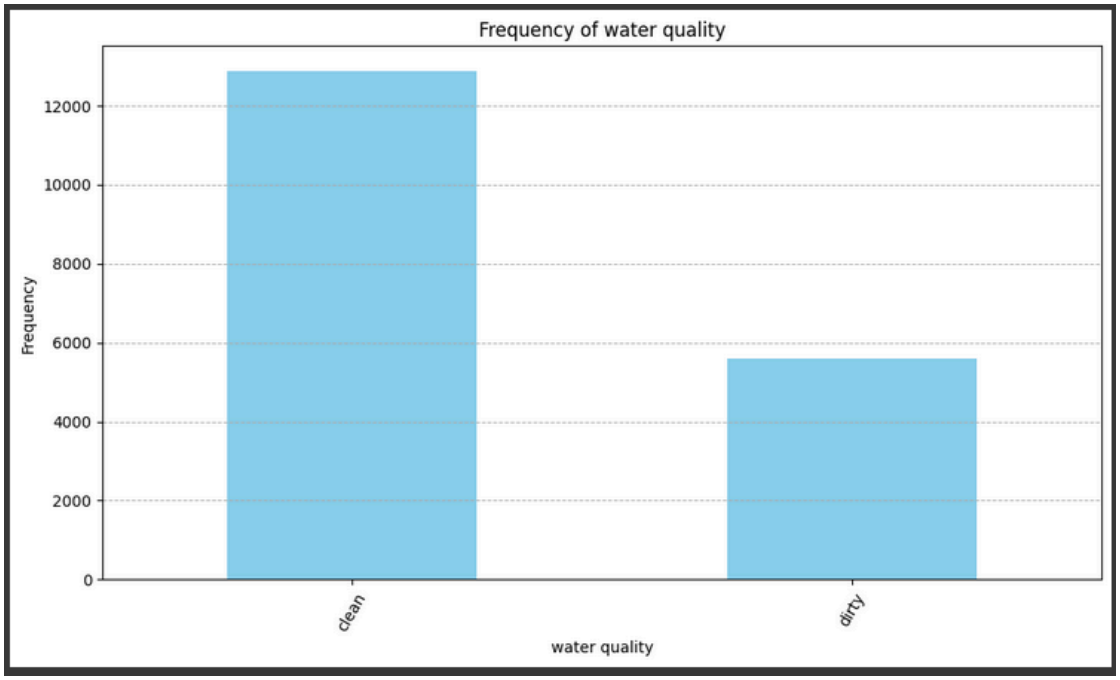
Plot EDA sebagai berikut:



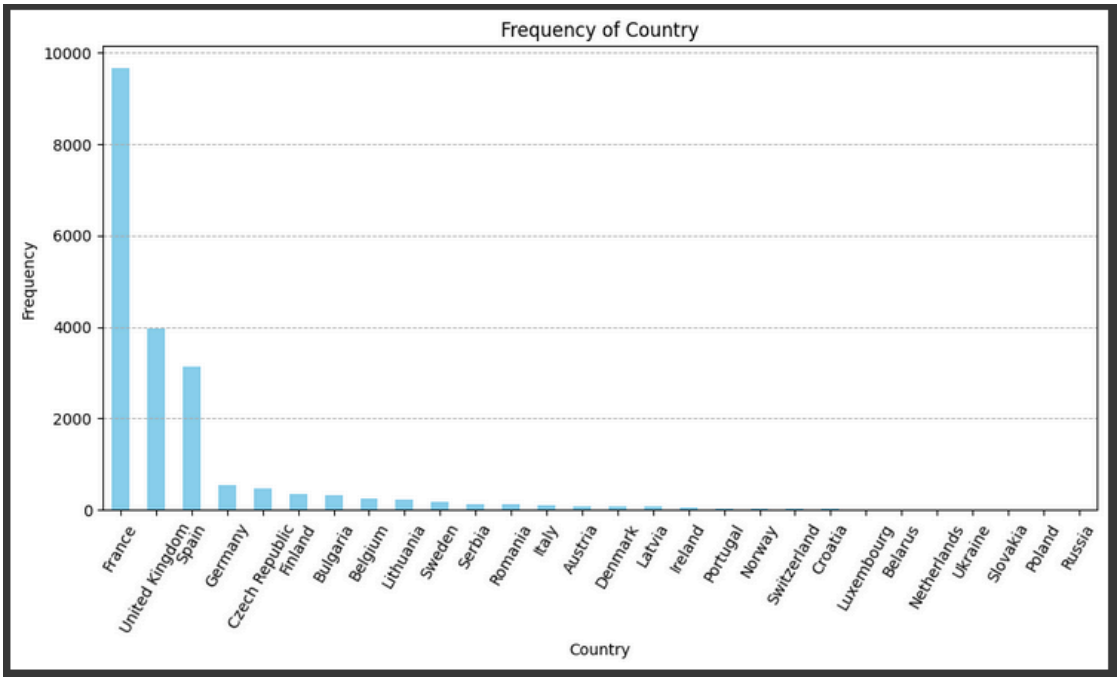
Gambar 3.6 Kualitas Air berdasarkan Uom



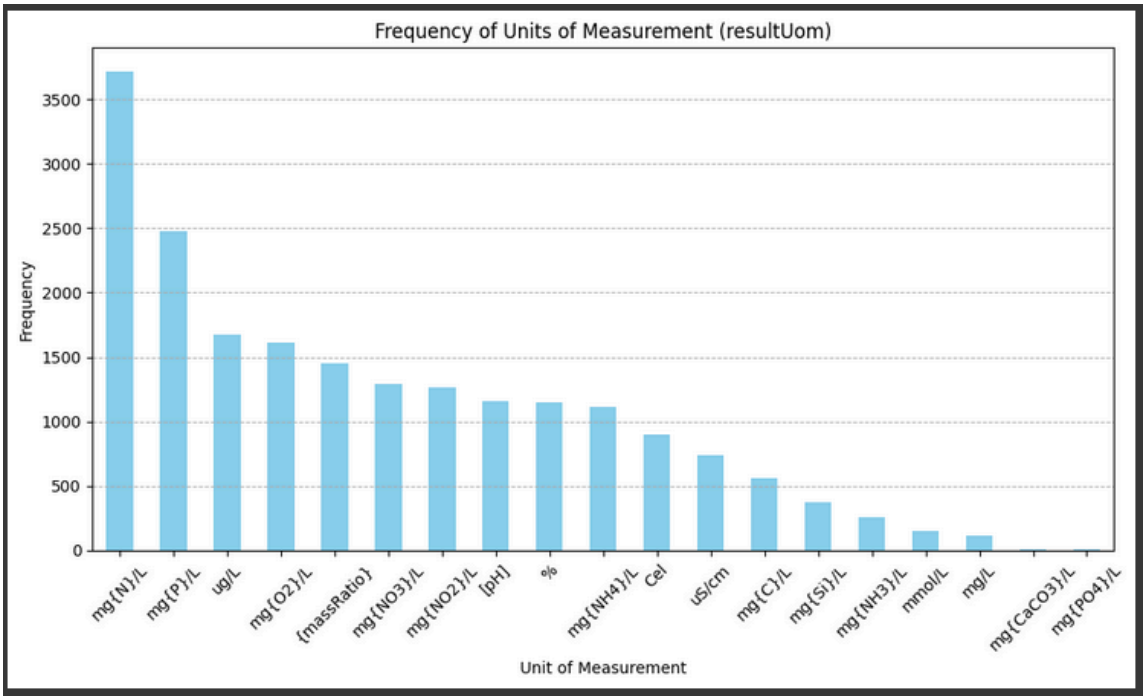
Gambar 3.7 Kualitas Air berdasarkan observedPropertyDeterminandCode



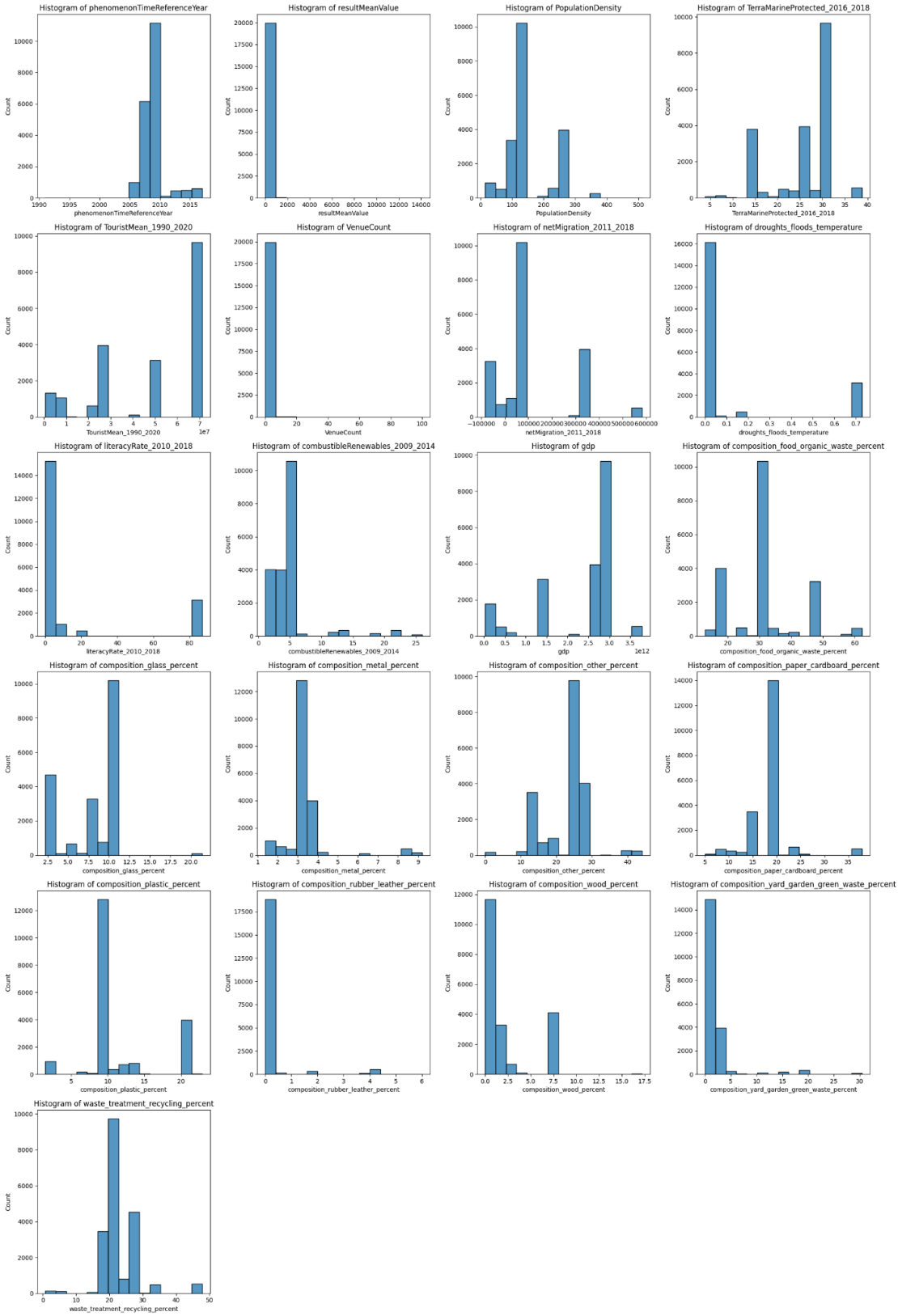
Gambar 3.8 Kualitas Air berdasarkan Uom tanpa Air yang tidak bisa di define



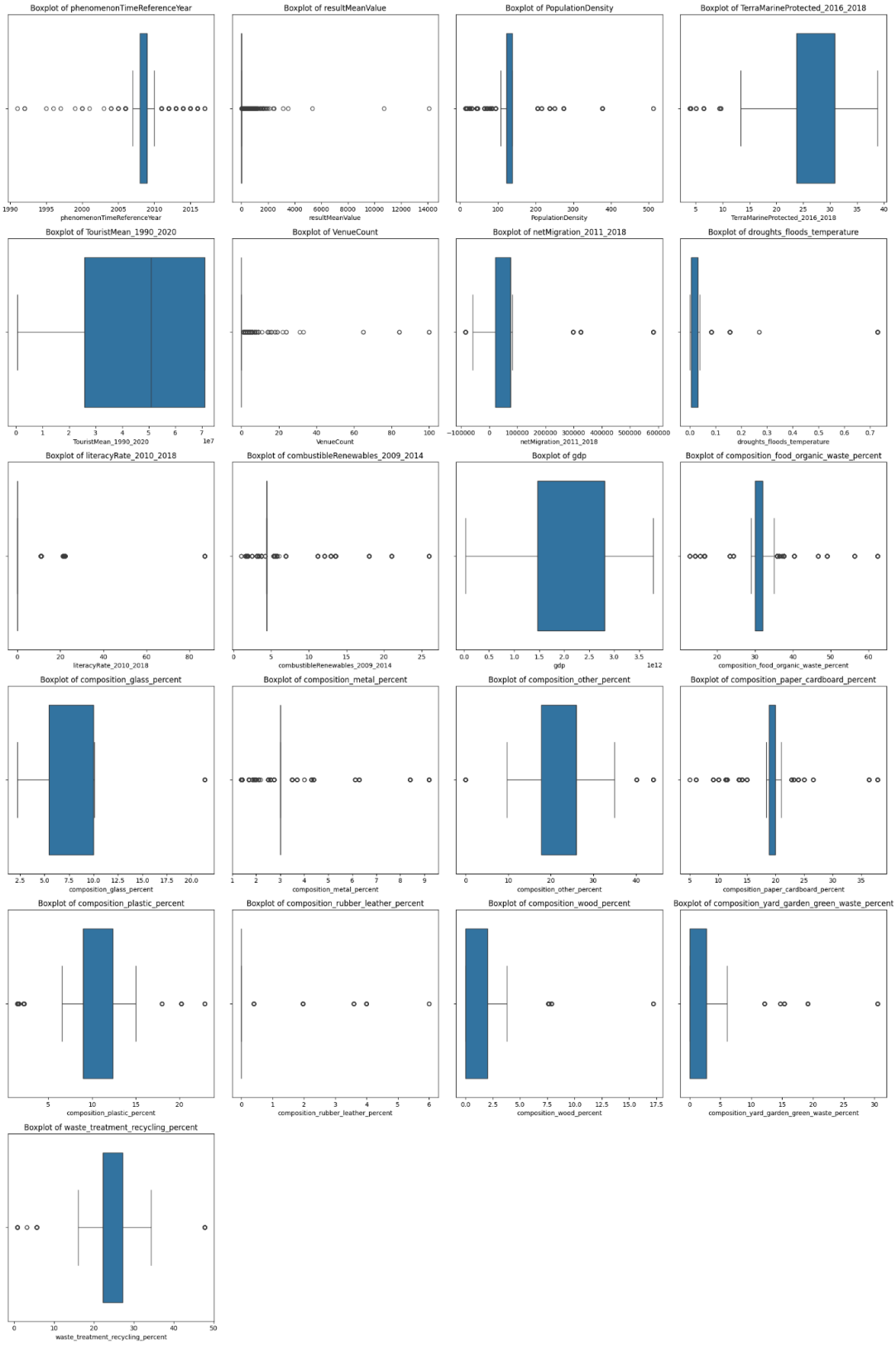
Gambar 3.9 Kualitas Air berdasarkan negara nya



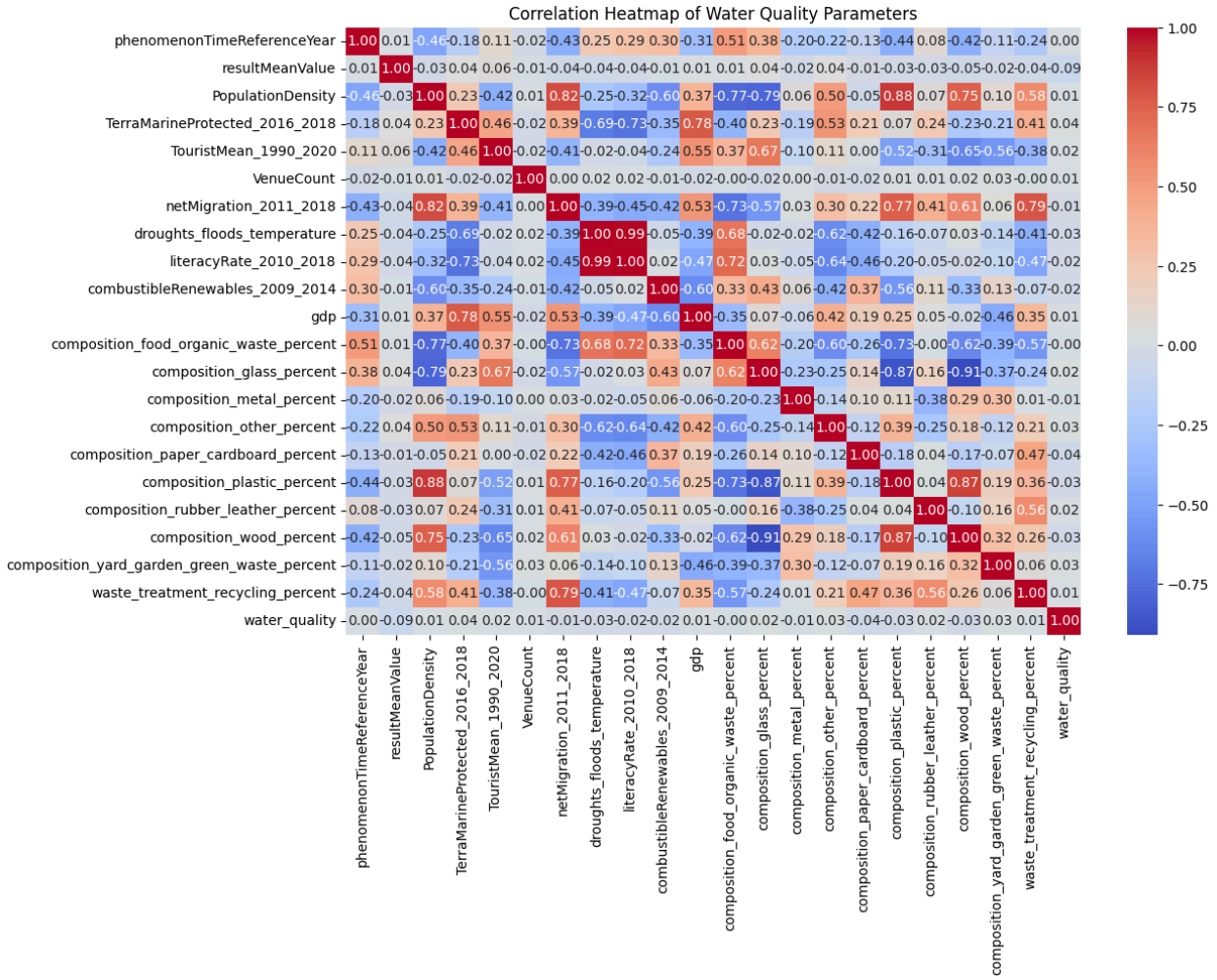
Gambar 3.10 Frekuensi kandungan dalam Air



Gambar 3.11 Bar Plot Data Numerical dalam dataset

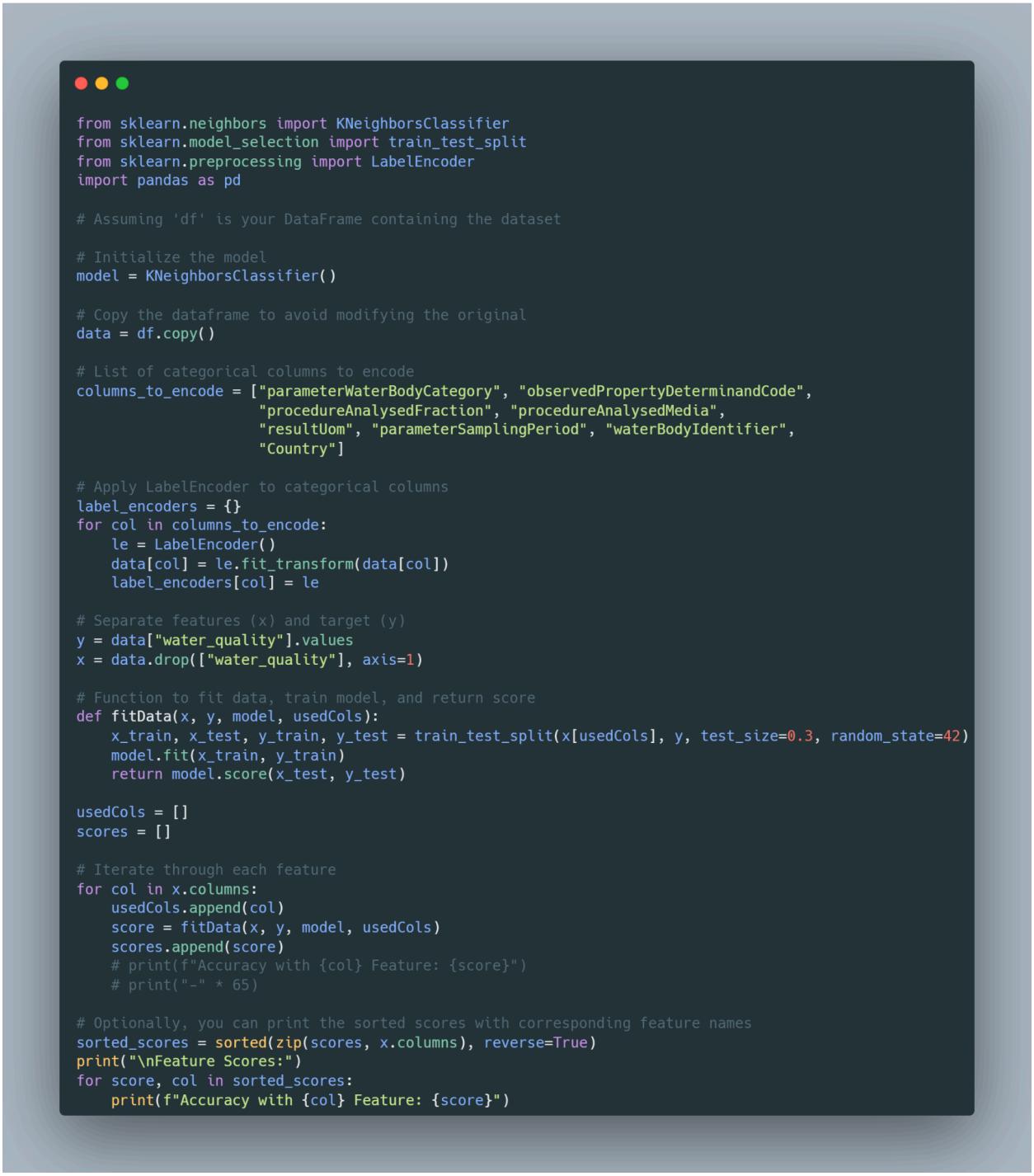


Gambar 3.12 Box Plot Data Numerical dalam dataset



Gambar 3.13 Heatmap Korelasi Data Numerical dalam dataset

3.4. Feature Selection



```python
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
import pandas as pd

# Assuming 'df' is your DataFrame containing the dataset

# Initialize the model
model = KNeighborsClassifier()

# Copy the dataframe to avoid modifying the original
data = df.copy()

# List of categorical columns to encode
columns_to_encode = ["parameterWaterBodyCategory", "observedPropertyDeterminandCode",
                     "procedureAnalysedFraction", "procedureAnalysedMedia",
                     "resultUom", "parameterSamplingPeriod", "waterBodyIdentifier",
                     "Country"]

# Apply LabelEncoder to categorical columns
label_encoders = {}
for col in columns_to_encode:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Separate features (x) and target (y)
y = data["water_quality"].values
x = data.drop(["water_quality"], axis=1)

# Function to fit data, train model, and return score
def fitData(x, y, model, usedCols):
    x_train, x_test, y_train, y_test = train_test_split(x[usedCols], y, test_size=0.3, random_state=42)
    model.fit(x_train, y_train)
    return model.score(x_test, y_test)

usedCols = []
scores = []

# Iterate through each feature
for col in x.columns:
    usedCols.append(col)
    score = fitData(x, y, model, usedCols)
    scores.append(score)
    # print(f"Accuracy with {col} Feature: {score}")
    # print("-" * 65)

# Optionally, you can print the sorted scores with corresponding feature names
sorted_scores = sorted(zip(scores, x.columns), reverse=True)
print("\nFeature Scores:")
for score, col in sorted_scores:
    print(f"Accuracy with {col} Feature: {score}")
```

Gambar 3.14 Proses Feature Selection

Feature Selection dilakukan dengan cara melakukan percobaan tiap parameter yang kemudian dievaluasi skor akurasinya dengan model KNN. Metode feature selection ini dilakukan karena korelasi fitur dengan target pada gambar 3.10 sangat rendah,

sehingga tidak bisa digunakan untuk menentukan fitur mana yang akan digunakan hanya dengan melihat dari korelasi datanya saja

Feature	Accuracy
resultMeanValue	0,986
waterBodyIdentifier	0,866
Country	0,866
TouristMean_1990_2020	0,862
PopulationDensity	0,862
TerraMarineProtected_2016_2018	0,862
netMigration_2011_2018	0,862
VenueCount	0,862
literacyRate_2010_2018	0,860
droughts_floods_temperature	0,860
combustibleRenewables_2009_2014	0,860
resultUom	0,812
procedureAnalysedMedia	0,811
procedureAnalysedFraction	0,811
observedPropertyDeterminandCode	0,811
parameterSamplingPeriod	0,803
phenomenonTimeReferenceYear	0,758
parameterWaterBodyCategory	0,696
waste_treatment_recycling_percent	0,512
gdp	0,512
composition_yard_garden_green_waste_percent	0,512
composition_wood_percent	0,512
composition_rubber_leather_percent	0,512
composition_plastic_percent	0,512
composition_paper_cardboard_percent	0,512
composition_other_percent	0,512
composition_metal_percent	0,512
composition_glass_percent	0,512
composition_food_organic_waste_percent	0,512

Dari tabel hasil evaluasi tiap fitur diatas lalu dilakukan threshold sebesar 0,7 sehingga fitur yang digunakan untuk skenario feature selection adalah resultMeanValue, waterBodyIdentifier, Country, TouristMean_1990_2020, PopulationDensity, TerraMarineProtected_2016_2018, netMigration_2011_2018, VenueCount, literacyRate_2010_2018, droughts_floods_temperature, combustibleRenewables_2009_2014, resultUom, procedureAnalysedMedia, procedureAnalysedFraction, observedPropertyDeterminandCode, parameterSamplingPeriod, phenomenonTimeReferenceYear

3.5. Preprocessing

Setelah melakukan EDA, dilakukan normalisasi data dnegan StandardScaler. Selain itu didapati adanya imbalance, diputuskan untuk melakukan Oversampling Smote pada Dataset untuk menangani imbalance tersebut

3.5.1. Normalisasi Data

Normaslisasi data dilakukan untuk menghindari dominasi oleh feature yang ada

```
# Apply StandardScaler
scaler = StandardScaler()
X = scaler.fit_transform(X)
X
```

Gambar 3.15 Normalisasi Data

3.5.2. Oversampling Smote

Dilakukan Oversampling dikarenakan adanya ketidak seimbangan yang lumayan besar pada data, dimana lebih banyak lebih banyak air yang kotor daripada air yang bersih dengan perbandingan 2.2/1.

```
# Apply SMOTE
smote = SMOTE(random_state=42)
X, y = smote.fit_resample(X, y)
```

Gambar 3.16 Oversampling Smote

```
Resampled class distribution:
 water_quality
0    12789
1    12789
Name: count, dtype: int64
```

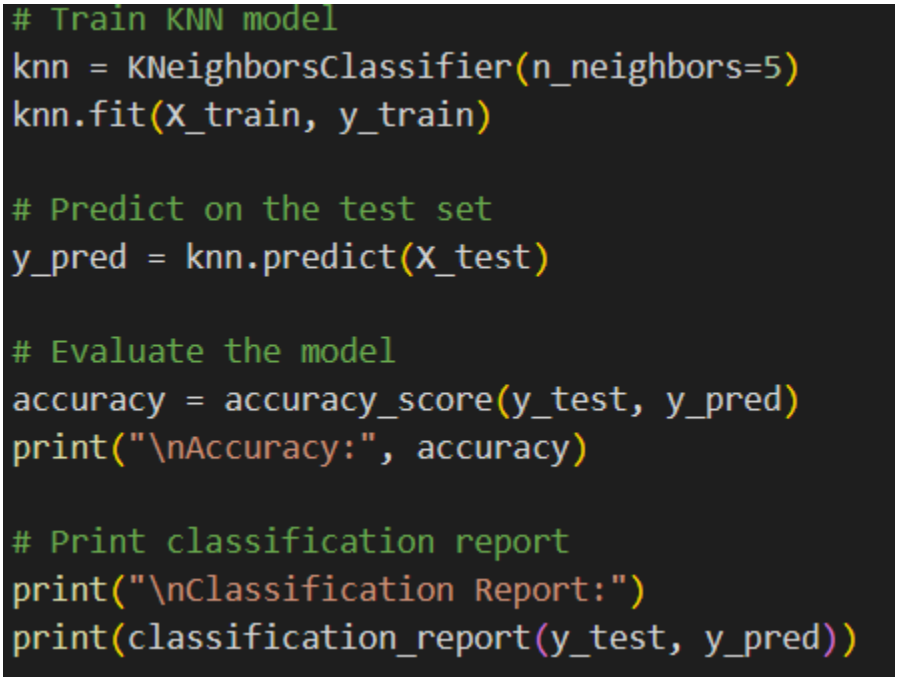
Gambar 3.17 Hasil dari Smote

3.6. Model Training

Kami menggunakan lima model—XGBoost, Artificial Neural Network (ANN), K-Nearest Neighbors (KNN), Random Forest, dan Decision Tree—untuk mengevaluasi performa dan kemampuan masing-masing dalam menangani berbagai jenis data dan kompleksitas.

3.6.1. K-Nearest Neighbors (KNN)

Kami memilih KNN karena kesederhanaannya dan kemampuannya memberikan prediksi yang baik untuk data dengan distribusi yang mirip antara kelas. KNN bekerja dengan mencari k tetangga terdekat dari data baru dan menentukan kelas berdasarkan mayoritas tetangga tersebut.



```
# Train KNN model
knn = KNeighborsClassifier(n_neighbors=5)
knn.fit(X_train, y_train)

# Predict on the test set
y_pred = knn.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Gambar 3.18 KNN model

3.6.2. Decision Tree

Decision Tree digunakan karena mudah dipahami dan diinterpretasikan. Model ini bekerja dengan membagi data berdasarkan fitur yang memberikan informasi gain tertinggi pada setiap node, sehingga menghasilkan keputusan yang jelas.

```
# Train Decision Tree model
decision_tree = DecisionTreeClassifier(random_state=42)
decision_tree.fit(X_train, y_train)

# Predict on the test set
y_pred = decision_tree.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Gambar 3.19 Model Decision Tree

3.6.3. Random Forest

Random Forest dipilih karena kemampuannya menangani overfitting yang umum pada pohon keputusan. Dengan menggabungkan banyak pohon keputusan, model ini memberikan prediksi yang lebih stabil dan akurat.

```
# Train Random Forest model
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
random_forest.fit(X_train, y_train)

# Predict on the test set
y_pred = random_forest.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Gambar 3.20 Model Random Forest

3.6.4. XGBoost

Kami menggunakan XGBoost karena kinerjanya yang tinggi dan kemampuannya menangani data tidak seimbang. Algoritma ini membangun model pohon keputusan secara bertahap dan mengoptimalkan loss function untuk mencapai hasil yang optimal.

```
# Train XGBoost model
xgboost_model = XGBClassifier(random_state=42)
xgboost_model.fit(X_train, y_train)

# Predict on the test set
y_pred = xgboost_model.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Gambar 3.21 Model XGBoost

3.6.5. Artificial Neural Network (ANN)

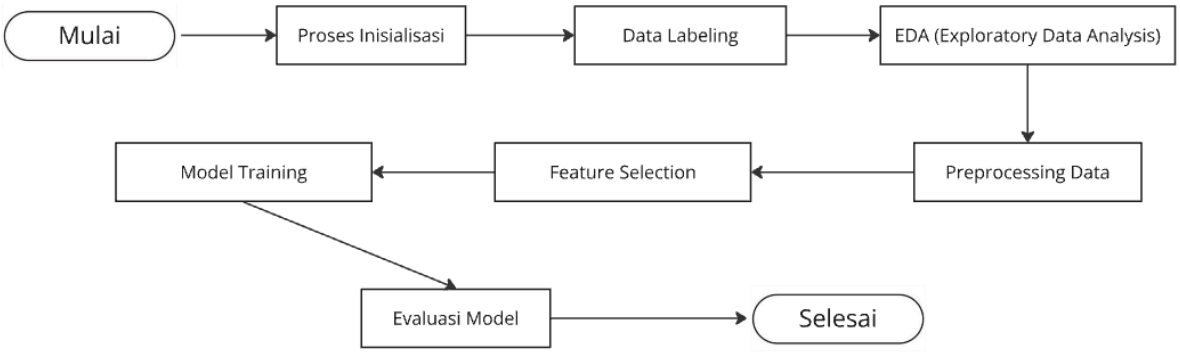
ANN digunakan karena kemampuannya dalam menangani data yang kompleks dan non-linear. Jaringan ini dilatih menggunakan backpropagation untuk meminimalkan loss function dengan mengupdate bobot jaringan secara iteratif.

```
# Build the ANN model
ann_model = Sequential()
ann_model.add(Dense(512, activation='relu', input_shape=(X_train.shape[1],)))
ann_model.add(Dense(128, activation='relu'))
ann_model.add(Dense(64, activation='relu'))
ann_model.add(Dense(32, activation='relu'))
ann_model.add(Dense(1, activation='sigmoid'))
# Compile the model
ann_model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Define the early stopping callback
early_stopping = EarlyStopping(monitor='val_loss', patience=10, restore_best_weights=True)

# Increase the number of epochs
ann_model.fit(X_train, y_train, epochs=100, batch_size=32, validation_split=0.2, callbacks=[early_stopping])

# Predict on the test set
y_pred_prob = ann_model.predict(X_test)

# Adjust threshold for multiclass
y_pred = (y_pred_prob > 0.5).astype(int).flatten()

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
print("\nAccuracy:", accuracy)

# Print classification report
print("\nClassification Report:")
print(classification_report(y_test, y_pred))
```

Gambar 3.22 Model ANN

3.7. Desain Sistem

Gambar 3.23 Desain Sistem



Desain sistem ini mencakup tahapan mulai dari inisialisasi hingga evaluasi model, memastikan proses pengolahan data yang komprehensif dan terstruktur untuk analisis kualitas air.

3.7.1. Proses Inisialisasi

Melakukan proses inisialisasi library yang akan digunakan dan melakukan proses inisialisasi dataset dengan menggunakan Pandas.

3.7.2. Proses Labeling

Proses ini melibatkan pengecekan nilai hasil pengukuran terhadap batasan regulasi yang telah ditentukan untuk berbagai parameter air. Setiap baris data dalam DataFrame akan diberi label 'clean' jika nilai berada dalam batas yang ditetapkan, 'dirty' jika melebihi batas, dan 'undefined' jika unit pengukuran tidak terdaftar dalam batasan regulasi.

3.7.3. EDA (Exploratory Data Analysis)

Pada tahap EDA ini, dilakukan beberapa langkah untuk memahami data dengan lebih baik:

- Visualisasi Data Kualitas Air
  Data divisualisasikan dalam bentuk diagram batang untuk melihat distribusi kualitas air berdasarkan hasil pengukuran.
- Filtering dan Labeling Kualitas Air
  Data yang tidak terdefinisi ('undefined') dihapus, dan dilakukan labeling kualitas air berdasarkan batasan regulasi yang telah ditetapkan.
- Analisis Frekuensi Negara

Dilakukan analisis terhadap frekuensi negara yang tercatat dalam dataset, dengan menggunakan diagram batang untuk visualisasi distribusi data negara.

- Analisis Frekuensi Unit Pengukuran
  Dilakukan analisis terhadap frekuensi unit pengukuran yang digunakan dalam dataset, menggunakan diagram batang untuk memvisualisasikan distribusi unit pengukuran.

3.7.4. Preprocessing

Dengan menggabungkan oversampling untuk keseimbangan kelas dan normalisasi untuk konsistensi skala, data siap digunakan untuk analisis dan pengembangan model lebih lanjut.

3.7.4.1. Oversampling

Oversampling dengan SMOTE digunakan untuk menyeimbangkan jumlah sampel antara kategori minoritas dan mayoritas dalam dataset. Misalnya, jika data awal memiliki lebih banyak sampel untuk kategori 'clean' daripada 'dirty', SMOTE akan membuat sampel sintetis untuk 'dirty' sehingga jumlahnya seimbang.

3.7.4.2. Normalisasi Data

Normalisasi dengan StandardScaler digunakan untuk memastikan bahwa semua fitur dalam dataset memiliki skala yang seragam. Hal ini penting karena beberapa model machine learning sensitif terhadap skala data, sehingga normalisasi memastikan interpretasi yang lebih akurat dari data.

3.7.5. Feature Selection

Feature selection digunakan dalam analisis data untuk memilih fitur-fitur yang paling relevan dalam memprediksi kualitas air. Proses dimulai dengan mengubah fitur-fitur kategorikal menjadi numerik agar dapat digunakan oleh model machine learning. Selanjutnya, fitur-fitur dipisahkan dari target yang ingin diprediksi, yaitu kualitas air. Dengan menggunakan model KNeighborsClassifier, setiap fitur dievaluasi secara berurutan, mengukur akurasi model saat setiap fitur ditambahkan. Fitur-fitur yang memberikan akurasi tinggi, seperti 'resultMeanValue', 'waterBodyIdentifier', dan 'Country', dianggap penting karena kontribusi positif mereka dalam prediksi kualitas air. Dengan menghilangkan fitur-fitur yang kurang relevan, diharapkan dapat memperbaiki efisiensi dan performa model dalam memprediksi target dengan lebih baik.

3.7.6. Pembuatan Model

Melakukan deklarasi model pembelajaran mesin dan membuat arsitektur model deep learning, yakni ANN.

Gambar 3.24 KNN Declaration

```
knn = KNeighborsClassifier(n_neighbors=5)
```

Gambar 3.25 Decision Tree Declaration

```
decision_tree = DecisionTreeClassifier(random_state=42)
```

Gambar 3.26 Random Forest Declaration

```
random_forest = RandomForestClassifier(n_estimators=100, random_state=42)
```

Gambar 3.27 XGBoost Declaration

```
xgboost_model = XGBClassifier(random_state=42)
```

Gambar 3.24 Arsitektur ANN

Model: "sequential_3"

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 512)	15,360
dense_1 (Dense)	(None, 128)	65,664
dense_2 (Dense)	(None, 64)	8,256
dense_3 (Dense)	(None, 32)	2,080
dense_4 (Dense)	(None, 1)	33

Total params: 91,393 (357.00 KB)
Trainable params: 91,393 (357.00 KB)
Non-trainable params: 0 (0.00 B)

3.7.7. Tanpa Oversampling, Feature Selection, Hyperparameter Tuning

Analisis dilakukan menggunakan data asli tanpa dilakukan oversampling, feature selection, atau hyperparameter tuning.

Gambar 3.25 Skenario tanpa Oversampling

```python
# Copy the dataframe to avoid modifying the original
data = df.copy()

# List of categorical columns to encode
columns_to_encode = ["parameterWaterBodyCategory", "observedPropertyDeterminandCode",
                     "procedureAnalysedFraction", "procedureAnalysedMedia",
                     "resultUom", "parameterSamplingPeriod", "waterBodyIdentifier",
                     "Country"]

# Apply LabelEncoder to categorical columns
label_encoders = {}
for col in columns_to_encode:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Separate features (x) and target (y)
y = data["water_quality"]
X = data.drop(["water_quality"], axis=1)
# Check the balance of the classes
print("Original class distribution:\n", y.value_counts())
```

Gambar 3.26 Data normal

```
Original class distribution:
 water_quality
0    12789
1     5578
Name: count, dtype: int64
```

3.7.8. Smote Oversampling

Skenario kedua menggunakan Smote Oversampling untuk mencoba menangani Imbalance pada data.

Gambar 3.27 Kode Data sebelum Smote dilakukan

```python
# Separate features (x) and target (y)
y = data["water_quality"]
X = data.drop(["water_quality"], axis=1)
# Check the balance of the classes
print("Original class distribution:\n", y.value_counts())
```

Gambar 3.28 Smote Oversampling

```
23 # Apply SMOTE
24 smote = SMOTE(random_state=42)
25 X, y = smote.fit_resample(X, y)
26
27 # Check the balance of the classes after applying SMOTE
28 print("\nResampled class distribution:\n", pd.Series(y).value_counts())
```

Gambar 3.29 Sebelum dan sesudah dilakukan Smote Oversampling

```
Original class distribution:
 water_quality
0    12789
1     5578
Name: count, dtype: int64

Resampled class distribution:
 water_quality
0    12789
1    12789
Name: count, dtype: int64
```

Disini bisa kita lihat bahwa setelah dilakukan Smote Oversampling, imbalance pada dataset telah menghilang.

3.7.9. Skenario Pengujian 3

Pada skenario ketiga kami, kami menerapkan Feature Selection dengan mempertimbangkan hanya fitur-fitur yang dipilih, yaitu: resultMeanValue, waterBodyIdentifier, Country, TouristMean_1990_2020, PopulationDensity, TerraMarineProtected_2016_2018, netMigration_2011_2018, VenueCount, literacyRate_2010_2018, droughts_floods_temperature, combustibleRenewables_2009_2014, resultUom, procedureAnalysedMedia, procedureAnalysedFraction, observedPropertyDeterminandCode, parameterSamplingPeriod, dan phenomenonTimeReferenceYear.

Gambar 3.30 Skenario Feature Selection

```
data = df.loc[:, ['resultMeanValue', 'waterBodyIdentifier', 'Country', 'TouristMean_1990_2020', 'PopulationDensity',
                  'TerraMarineProtected_2016_2018', 'netMigration_2011_2018', 'VenueCount', 'literacyRate_2010_2018',
                  'droughts_floods_temperature', 'combustibleRenewables_2009_2014', 'resultUom', 'procedureAnalysedMedia',
                  'procedureAnalysedFraction', 'observedPropertyDeterminandCode', 'parameterSamplingPeriod', 'phenomenonTimeReferenceYear', 'water_quality']]

# List of categorical columns to encode
columns_to_encode = ["observedPropertyDeterminandCode",
                     "procedureAnalysedFraction", "procedureAnalysedMedia",
                     "resultUom", "parameterSamplingPeriod", "waterBodyIdentifier",
                     "Country"]

# Apply LabelEncoder to categorical columns
label_encoders = {}
for col in columns_to_encode:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Separate features (x) and target (y)
y = data["water_quality"]
X = data.drop(["water_quality"], axis=1)
# Check the balance of the classes
data
```

Gambar 3.31 Data setelah dilakukan Feature Selection bagian kiri

	resultMeanValue	waterBodyIdentifier	Country	TouristMean_1990_2020	PopulationDensity	TerraMarineProtected_2016_2018	netMigration_2011_2018	VenueCount	literacyRate_2010_2018	droughts_floods_temperature
0	0.063310	1227	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
1	0.046733	1385	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
2	132.859000	1385	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
3	11.578376	1439	8	71176346.0	122.299437	30.831906	75808.375	2.0	0.000000	0.005718
5	2.477792	311	21	50941692.0	93.677197	15.047884	-40055.250	4.0	87.158924	0.729194
...	...	...	...	...	...	...	...	...	...	...
19995	0.092466	1935	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
19996	89.908300	1935	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
19997	18.901608	1937	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
19998	307.307000	1937	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718
19999	7.954790	1937	8	71176346.0	122.299437	30.831906	75808.375	0.0	0.000000	0.005718

18367 rows × 18 columns

Gambar 3.32 Data setelah dilakukan Feature Selection bagian kanan

combustibleRenewables_2009_2014	resultUom	procedureAnalysedMedia	procedureAnalysedFraction	observedPropertyDeterminandCode	parameterSamplingPeriod	phenomenonTimeReferenceYear	water_quality
4.457840	8	0	1	2	12	2009	0
4.457840	8	0	1	2	12	2009	0
4.457840	17	0	1	24	12	2009	0
4.457840	9	0	1	1	12	2009	1
4.363288	9	0	1	1	12	2009	0
...	...	...	...	...	...	...	...
4.457840	8	0	1	2	12	2009	0
4.457840	0	0	1	7	12	2009	1
4.457840	9	0	1	1	12	2009	1
4.457840	17	0	1	25	12	2009	0
4.457840	2	0	1	15	12	2009	0

3.7.10. Skenario Pengujian 4

Skenario keempat kami melibatkan penggunaan Hyperparameter Tuning untuk mengurangi risiko overfitting, serta untuk meningkatkan efisiensi komputasi dan mempercepat proses pelatihan model.

Gambar 3.33 Skenario Hyperparameter Tuning

```
# Copy the dataframe to avoid modifying the original
data = df.copy()

# List of categorical columns to encode
columns_to_encode = ["parameterWaterBodyCategory", "observedPropertyDeterminandCode",
                     "procedureAnalysedFraction", "procedureAnalysedMedia",
                     "resultUom", "parameterSamplingPeriod", "waterBodyIdentifier",
                     "Country"]

# Apply LabelEncoder to categorical columns
label_encoders = {}
for col in columns_to_encode:
    le = LabelEncoder()
    data[col] = le.fit_transform(data[col])
    label_encoders[col] = le

# Separate features (x) and target (y)
y = data["water_quality"]
X = data.drop(["water_quality"], axis=1)
# Check the balance of the classes
data
```

Gambar 3.34 Trial Parameter untuk KNN

Parameter	Range/Choices	Type
n_neighbors	[1, 30]	int
weights	['uniform', 'distance']	categorical
p	[1, 2]	int

Gambar 3.35 Trial Parameter untuk Decision Tree

Parameter	Range/Choices	Type
max_depth	[1, 32]	int
min_samples_split	[2, 20]	int
min_samples_leaf	[1, 20]	int

Gambar 3.36 Trial Parameter untuk Random Forest

Parameter	Range/Choices	Type
n_estimators	[50, 300]	int
max_depth	[2, 32]	int
min_samples_split	[2, 20]	int
min_samples_leaf	[1, 20]	int
max_features	['sqrt', 'log2']	categorical

Gambar 3.37 Trial Parameter untuk XGBoost

Parameter	Range/Choices	Type
max_depth	[3, 10]	int
learning_rate	[0.01, 0.3]	float
n_estimators	[100, 1000]	int
gamma	[0, 5]	float
min_child_weight	[1, 10]	int
subsample	[0.5, 1.0]	float
colsample_bytree	[0.5, 1.0]	float
lambda	[1e-8, 1.0] (log scale)	float
alpha	[1e-8, 1.0] (log scale)	float

Gambar 3.34 Trial Parameter untuk ANN

Parameter	Range/Choices	Type
n_layers	[1, 3]	int
activation	['relu', 'sigmoid']	categorical
optimizer	['Adam', 'RMSprop']	categorical
learning_rate	[1e-5, 1e-1] (log scale)	float
dropout_rate	[0.0, 0.5]	float
batch_size	[16, 128]	int
num_epochs	[10, 100]	int
n_units_l{i}	[4, 128]	int

3.7.11. Evaluasi Model

Memilih model terbaik setelah keempat skenario pengujian dilakukan dan melakukan classification report pada data hasil prediksi model tersebut terhadap data test.

BAB IV
Hasil dan Pembahasan

4.1. Hasil Pengujian

Pengujian tiap model dan skenario menggunakan metrik accuracy, didapatkan hasil seperti Tabel 4.1

Model	Baseline	Oversampling SMOTE	Feature Selection	Hyperparameter Tuning
KNN	0,8941	0,9158	0,8952	0,9167
Decision Tree	0,9989	0,9949	0,9992	0,9981
Random Forest	0,9924	0,9932	0,9951	0,9924
XGBoost	0,9976	0,9951	0,9973	0,9976
ANN	0,9159	0,9322	0,9197	0,9077

Tabel 4.1 Hasil Pengujian tiap skenario

4.2. Analisis Hasil

4.2.1. Tanpa Oversampling, Feature Selection, dan Hyperparameter Tuning

Dalam skenario tanpa oversampling, feature selection, dan hyperparameter tuning, performa akurasi dari beberapa model machine learning diukur dan dianalisis. Model **K-Nearest Neighbors (KNN)** menunjukkan akurasi sebesar 0,8941, yang menunjukkan performa yang cukup baik namun tidak terbaik dibandingkan model lainnya. **Decision Tree** mencapai akurasi yang sangat tinggi sebesar 0,9989, hampir mendekati kesempurnaan dalam klasifikasi. Model **Random Forest** juga memiliki akurasi yang sangat tinggi yaitu 0,9924, sedikit di bawah Decision Tree namun masih dalam kategori performa yang sangat baik. **XGBoost** menunjukkan akurasi yang hampir sama tinggi dengan Decision Tree, yaitu 0,9976, menegaskan kekuatan model ini dalam menangani berbagai jenis data. Terakhir, **Artificial Neural Network (ANN)** memiliki akurasi sebesar 0,9159, yang lebih baik daripada KNN namun masih di bawah model-model ensemble seperti Random Forest dan XGBoost. Hasil ini menunjukkan bahwa dalam kondisi tanpa teknik peningkatan data dan tuning parameter, model-model ensemble dan boosting cenderung memberikan hasil yang lebih superior dibandingkan dengan model yang lebih sederhana seperti KNN dan ANN.

4.2.2. Oversampling

Dalam skenario dengan oversampling menggunakan SMOTE, performa akurasi dari beberapa model machine learning diukur dan dianalisis. Model K-Nearest Neighbors (KNN) menunjukkan akurasi sebesar 0,9158, yang mengalami peningkatan dibandingkan dengan tanpa oversampling, menunjukkan bahwa SMOTE membantu KNN menangani kelas minoritas dengan lebih baik. Decision Tree mencapai akurasi sebesar 0,9949, sedikit menurun dibandingkan dengan tanpa oversampling tetapi masih tetap sangat tinggi. Random Forest memiliki akurasi sebesar 0,9932, yang juga menunjukkan sedikit peningkatan dan mempertahankan performa yang sangat baik. XGBoost menunjukkan akurasi sebesar 0,9951, sedikit meningkat dari skenario tanpa oversampling, menegaskan kekuatan model ini dalam menangani data yang seimbang. Terakhir, Artificial Neural Network (ANN) memiliki akurasi sebesar 0,9322, mengalami peningkatan signifikan dibandingkan dengan tanpa oversampling, menunjukkan bahwa SMOTE sangat efektif dalam membantu ANN menangani data yang tidak seimbang. Hasil ini menunjukkan bahwa oversampling dengan SMOTE dapat meningkatkan akurasi model, terutama pada model yang lebih sederhana seperti KNN dan ANN, sementara model-model ensemble seperti Random Forest dan XGBoost tetap menunjukkan performa yang sangat baik.

4.2.3. Feature Selection

Dalam skenario dengan feature selection, performa akurasi dari beberapa model machine learning diukur dan dianalisis. Model K-Nearest Neighbors (KNN) menunjukkan akurasi sebesar 0,8952, sedikit meningkat dibandingkan dengan tanpa feature selection, menunjukkan bahwa pemilihan fitur membantu KNN dalam menangani data lebih efisien. Decision Tree mencapai akurasi sebesar 0,9992, sedikit meningkat dan hampir mendekati kesempurnaan dalam klasifikasi. Random Forest memiliki akurasi sebesar 0,9951, juga meningkat dari skenario tanpa feature selection dan menunjukkan performa yang sangat baik. XGBoost menunjukkan akurasi sebesar 0,9973, sedikit menurun tetapi tetap sangat tinggi, menegaskan kekuatan model ini dalam menangani data dengan fitur yang lebih relevan. Terakhir, Artificial Neural Network (ANN) memiliki akurasi sebesar 0,9197, sedikit meningkat dibandingkan dengan tanpa feature selection, menunjukkan bahwa pemilihan fitur membantu ANN dalam meningkatkan performa. Hasil ini menunjukkan bahwa feature selection dapat meningkatkan akurasi model dengan menghilangkan fitur yang tidak relevan dan membantu model dalam fokus pada informasi yang lebih penting, terutama pada model yang lebih sederhana seperti KNN dan ANN, sementara model-model ensemble seperti Random Forest dan XGBoost tetap menunjukkan performa yang sangat baik.

4.2.4. Hyperparameter Tuning

Dalam skenario dengan hyperparameter tuning, performa akurasi dari beberapa model machine learning diukur dan dianalisis. Model K-Nearest Neighbors (KNN) menunjukkan akurasi sebesar 0,9167, yang mengalami peningkatan signifikan dibandingkan dengan tanpa tuning, menunjukkan bahwa pemilihan hyperparameter yang optimal dapat meningkatkan kinerja KNN secara substansial. Decision Tree mencapai akurasi sebesar 0,9981, sedikit menurun dibandingkan dengan tanpa tuning tetapi tetap sangat tinggi, menunjukkan stabilitas performa model ini. Random Forest memiliki akurasi sebesar 0,9924, sama dengan skenario tanpa tuning, menunjukkan bahwa model ini sudah bekerja dengan optimal bahkan tanpa tuning lebih lanjut. XGBoost menunjukkan akurasi sebesar 0,9976, sama dengan skenario tanpa tuning, menegaskan kekuatan model ini dalam performa konsisten. Terakhir, Artificial Neural Network (ANN) memiliki akurasi sebesar 0,9077, sedikit menurun dibandingkan dengan tanpa tuning, menunjukkan bahwa tuning tidak selalu memberikan peningkatan performa untuk semua model. Hasil ini menunjukkan bahwa hyperparameter tuning dapat meningkatkan akurasi model, terutama pada model yang lebih sederhana seperti KNN, sementara model-model ensemble seperti Random Forest dan XGBoost tetap menunjukkan performa yang sangat baik bahkan tanpa tuning lebih lanjut.

4.3. Perbandingan Skenario

Analisis perbandingan skenario metode pengujian menunjukkan bahwa model K-Nearest Neighbors (KNN) mengalami peningkatan akurasi paling signifikan dengan oversampling SMOTE dan hyperparameter tuning, dari 0,8941 menjadi 0,9158 dan 0,9167. Decision Tree tetap sangat optimal di berbagai skenario, dengan akurasi tertinggi 0,9992 dengan feature selection. Random Forest dan XGBoost menunjukkan stabilitas tinggi, dengan sedikit peningkatan akurasi menggunakan feature selection, masing-masing mencapai 0,9951 dan 0,9973. Artificial Neural Network (ANN) menunjukkan peningkatan akurasi dari 0,9159 menjadi 0,9322 dengan SMOTE, namun menurun dengan hyperparameter tuning menjadi 0,9077. Secara keseluruhan, metode oversampling dan tuning paling efektif untuk KNN dan ANN, sementara model ensemble seperti Decision Tree, Random Forest, dan XGBoost menunjukkan performa stabil dengan perubahan minimal.

BAB V
Kesimpulan

5.1. Kesimpulan

Berdasarkan hasil pengujian pada empat skenario berbeda—tanpa oversampling, dengan oversampling, feature selection, dan hyperparameter tuning—terlihat bahwa model Decision Tree dengan skenario feature selection memberikan hasil terbaik dalam memprediksi kualitas air. Hasil ini menunjukkan bahwa pemilihan fitur yang tepat mampu meningkatkan akurasi model secara signifikan. Dengan akurasi hampir sempurna (0,9992), model Decision Tree tidak hanya mudah diinterpretasikan tetapi juga sangat efektif dalam menangani kompleksitas data kualitas air. Penerapan feature selection memungkinkan pengurangan dimensionalitas data tanpa mengorbankan informasi penting, sehingga memperbaiki efisiensi dan performa model dalam klasifikasi kualitas air.

5.2. Saran

Pada penelitian selanjutnya, masih ada banyak teknik resampling dan metode pembelajaran mesin lain yang dapat diimplementasikan untuk mengatasi ketidakseimbangan kelas dalam memprediksi kualitas air. Metode resampling seperti ADASYN atau penggabungan beberapa teknik lain dapat diuji untuk mencari pendekatan yang paling optimal. Selain itu, melakukan hyperparameter tuning yang lebih mendalam pada model seperti KNN dan ANN untuk mendapatkan hasil prediksi lebih akurat dan mendekati model lain.

Daftar Pustaka

1. Environmental Protection Agency (EPA). (2023). National Primary Drinking Water Regulations. Retrieved from EPA Drinking Water Standards.
2. World Health Organization (WHO). (2022). WHO Guidelines for Drinking-water Quality. Retrieved from WHO Drinking Water Quality Guidelines.
3. European Union (EU). (2020). EU Drinking Water Directive. Retrieved from EU Drinking Water Directive.
4. Canadian Council of Ministers of the Environment (CCME). (2024). Water Quality Guidelines. Retrieved from CCME Guidelines.
5. Australian Drinking Water Guidelines (ADWG). (2023). Australian Drinking Water Guidelines. Retrieved from ADWG Guidelines.
6. Chen, Y., Song, L., Liu, Y., Yang, L., & Li, D. (2020). A Review of the Artificial Neural Network Models for Water Quality Prediction. Precision Agricultural Technology Integration Research Base, Ministry of Agriculture and Rural Affairs, China Agricultural University, Beijing.
7. Juahir, H., Zain, M. S., Toriman, M. E., Mokhtar, M., & Che Man, H. (2020). Application of Artificial Neural Network Models for Predicting Water Quality Index. Department of Chemistry, Faculty of Science, Universiti Malaya, Kuala Lumpur.
8. Ahmed, A. N., Othman, F. B., Afan, H. A., Ibrahim, R. K., Faif, C. M., Hossain, M. S., Ehteram, M., & Elshafie, A. (2020). Machine Learning Methods for Better Water Quality Prediction. Institute of Energy Infrastructure, Universiti Tenaga Nasional, Selangor.
9. Bui, D. T., Khosravi, K., Tiefenbacher, J., Nguyen, H., & Kazakis, N. (2020). Improving prediction of water quality indices using novel hybrid machine-learning algorithms. Science of The Total Environment, 137612.
10. Lu, H., & Ma, X. (2020). Hybrid decision tree-based machine learning models for short-term water quality prediction. Journal of Cleaner Production, 248, 119260.
11. Ahmed, U., Mumtaz, R., Anwar, H., Shah, A. A., Irfan, R., & García-Nieto, J. (2019). Efficient Water Quality Prediction Using Supervised Machine Learning. Water, 11(11), 2210.
12. Chen, K., Chen, H., Zhou, C., Huang, Y., Qi, X., Shen, R., Liu, F., Zuo, M., Zou, X., Wang, J., Zhang, Y., Chen, D., Chen, X., & Ren, H. (2019). Comparative analysis of surface water quality prediction performance and identification of key water parameters using different machine learning models based on big data. Journal of Cleaner Production, 248, 119260.
13. Zhu, M., Wang, J., Yang, X., Zhang, Y., Zhang, L., Ren, H., Wu, B., & Ye, L. (2019). A review of the application of machine learning in water quality evaluation. Journal of Cleaner Production, 248, 119260.
14. Al-Sulttani, A. O., Al-Mukhtar, M., Roomi, A. B., Farooque, A. A., Khedher, K. M., & Yaseen, Z. M. (2019). Proposition of new ensemble data-intelligence models for surface water quality prediction. Journal of Cleaner Production, 248, 119260.
15. Ghazali, N., & Ali, Z. M. (2022). Principal Component Analysis Approach in Klang River Water Quality Index Modelling. Environment and Ecology Research, 11(1), 165-182.
16. Liu, Y., & Zhang, X. (2024). Global water scarcity and pollution: Impacts and mitigation strategies. *Nature Climate Change*, 14(3), 102-115.

17. Jones, M. (2024). The global clean water crisis looms large: Study finds water quality is underrepresented in assessments. *Phys.org*. Retrieved from https://phys.org/news/2024-05-global-crisis-looms-large-quality.html
18. UNRWA. (2024). Water crisis in Gaza: Causes and solutions. *UN News*. Retrieved from https://news.un.org
19. WaterAid. (2023). Water security in sub-Saharan Africa: Challenges and opportunities. Retrieved from https://www.wateraid.org