Prediksi Keberadaan Spesies berdasarkan Lokasi menggunakan Algoritma Random Forest

syohana907@gmail.com[1] , desristenatal@gmail.com[2] , lennahutapea18@gmail.com[3]

INTISARI — Keanekaragaman hayati memiliki peran penting dalam menjaga keseimbangan alam, dan untuk melestarikannya, diperlukan pemahaman tentang distribusi spesies tumbuhan di berbagai wilayah. Penelitian ini bertujuan mengembangkan model prediksi keberadaan spesies tumbuhan menggunakan algoritma Random Forest dengan memanfaatkan data lingkungan seperti lokasi dan karakteristik tanah. Algoritma Random Forest dipilih karena kemampuannya dalam menangani data besar dan kompleks serta menghasilkan prediksi yang akurat. Hasil evaluasi menggunakan metrik F1-score berbobot menunjukkan model ini dapat memprediksi distribusi spesies dengan baik, meskipun masih ada tantangan terkait kesulitan dalam menjelaskan bagaimana model membuat prediksi dan pencocokan spesies. Meskipun demikian, model yang dihasilkan dapat mendukung kebijakan pelestarian keanekaragaman hayati berbasis data. Proses deployment juga telah dilakukan, namun masih memerlukan penyempurnaan untuk hasil yang lebih optimal.

KATA KUNCI — **Prediksi Spesies, Algoritma Random Forest, Metrik F1-Score**

I. PENDAHULUAN

Keanekaragaman hayati sangat penting untuk menjaga keseimbangan alam. Tumbuhan memiliki peran dalam menjaga iklim, air, dan kesuburan tanah. Untuk menjaga keanekaragaman ini, penting untuk mengetahui jenis-jenis tumbuhan yang hidup di suatu tempat. Namun, mencari tahu spesies apa saja yang ada di suatu wilayah tidaklah mudah. Dengan bantuan teknologi, seperti data P0 train, data PA train, data PA train soilgrids, dan data P0 train soilgrids, proses ini dapat dilakukan dengan lebih cepat dan akurat. Salah satu cara yang dapat digunakan untuk memprediksi keberadaan spesies adalah dengan algoritma Random Forest.

Random Forest adalah algoritma pembelajaran mesin yang bekerja dengan membentuk banyak pohon keputusan, lalu menggabungkan hasil dari masing-masing pohon untuk mendapatkan prediksi yang lebih akurat. Metode ini dikenal kuat dalam menangani data besar, mampu mengenali pola yang kompleks, serta cukup stabil terhadap variasi data. Dalam konteks prediksi spesies, Random Forest dapat digunakan untuk mengidentifikasi keterkaitan antara faktor lingkungan dan kehadiran spesies tertentu.

Penelitian ini bertujuan untuk mengembangkan model prediksi keberadaan spesies berdasarkan lokasi geografis menggunakan algoritma Random Forest. Hasil dari penelitian ini diharapkan dapat membantu memberikan informasi tambahan dalam kegiatan pemantauan spesies serta mendukung kebijakan pelestarian keanekaragaman hayati secara lebih efisien dan berbasis data.

II. TUJUAN

Tujuan dari proyek ini adalah :

a. Membangun model prediksi distribusi spesies tumbuhan menggunakan data lingkungan dan algoritma pembelajaran mesin, seperti Random Forest atau model lain yang relevan, untuk meningkatkan pemahaman mengenai keberadaan spesies tumbuhan di berbagai lokasi.
b. Mengeksplorasi hubungan antara faktor lingkungan, seperti data iklim, citra satelit, dan karakteristik tanah, untuk mengidentifikasi pola distribusi spesies tumbuhan dan meningkatkan akurasi prediksi.
c. Menilai potensi penggunaan data lingkungan dalam perencanaan pelestarian keanekaragaman hayati, untuk membantu dalam pengelolaan dan pemantauan spesies yang terancam punah atau invasive di berbagai wilayah.

III. RUANG LINGKUP

Studi ini berfokus pada penerapan algoritma pembelajaran mesin untuk memprediksi kehadiran spesies tumbuhan berdasarkan data lingkungan yang tersedia. Data yang digunakan mencakup informasi seperti Presence-Absence (PA), Presence-Only (PO), serta data lingkungan terkait seperti citra satelit, data iklim, dan karakteristik tanah yang diperoleh dari dataset GLC25-PA-train-soilgrids dan GLC25-PO-train-soilgrids Proses analisis akan melibatkan tahap pengolahan data, pelatihan model, evaluasi performa menggunakan metrik micro F1-score, dan interpretasi hasil prediksi. Fokus utama dari analisis ini adalah untuk memahami hubungan antara faktor-faktor lingkungan dan distribusi spesies tumbuhan, sehingga hasil prediksi dapat digunakan untuk mendukung perencanaan konservasi, pengelolaan keanekaragaman hayati, dan pengambilan kebijakan berbasis data yang lebih efektif.

IV. STUDI LITERATUR

1. Prediksi Keberadaan Spesies

Prediksi keberadaan spesies sangat penting dalam ekologi dan konservasi karena dapat membantu memperkirakan lokasi spesies berdasarkan faktor lingkungan yang ada. Salah satu metode yang digunakan untuk memodelkan distribusi spesies adalah algoritma berbasis entropi maksimum, yang membandingkan distribusi probabilitas berdasarkan data keberadaan spesies dan data latar belakang lingkungan. Metode ini bekerja dengan meminimalkan perbedaan antara kedua distribusi tersebut untuk menghasilkan model distribusi yang lebih akurat. Kelebihan utama dari metode ini adalah kemampuannya untuk bekerja dengan data keberadaan spesies saja, yang sering kali terbatas dalam penelitian [1].

Namun, meskipun efektif, metode ini memiliki beberapa kekurangan. Salah satunya adalah ketidakmampuan untuk menentukan dengan pasti seberapa banyak spesies tersebut ada di seluruh area yang dipelajari, atau prevalensinya. Selain itu, bias dalam pengumpulan data, seperti pemilihan lokasi yang tidak merata, dapat mempengaruhi hasil model. Oleh karena itu, penting untuk menggunakan data latar belakang yang representatif dan memilih fitur lingkungan yang tepat agar model yang dihasilkan lebih akurat dan dapat diandalkan [2].

2. Algoritma Random Forest

Algoritma Random Forest (RF) adalah salah satu metode yang sering digunakan untuk mengolah data citra satelit, terutama dalam memetakan tutupan hutan dan jenis-jenis hutan. RF bekerja dengan membuat banyak pohon keputusan (decision tree) dan menggabungkan hasilnya agar lebih akurat. Dalam penelitian oleh Waśniewski dan tim (2020), algoritma ini digunakan bersama citra Sentinel-2 dan data ketinggian permukaan tanah (DEM) untuk memetakan lima jenis hutan di Gabon. Hasilnya sangat baik, dengan akurasi mencapai lebih dari 97% untuk beberapa tipe hutan.

Kelebihan Random Forest adalah kemampuannya memproses banyak data sekaligus dan tetap bekerja dengan baik meskipun data latih tidak sempurna. Algoritma ini juga dapat menunjukkan fitur mana yang paling berpengaruh dalam proses klasifikasi. Dalam penelitian ini, DEM dan band “red-edge” dari Sentinel-2 adalah yang paling membantu dalam membedakan tipe-tipe hutan. Model klasifikasi yang dibuat juga bisa dipakai untuk citra lain, walaupun untuk tipe hutan hasilnya bisa berbeda karena perbedaan kondisi alam di wilayah tersebut [3].

V. RENCANA PENGERJAAN PROYEK

Adapun tahapan perencanaan dalam pengerjaan proyek penelitian ini sebagai berikut:

Task	Minggu				
	Minggu 1	Minggu 2	Minggu 3	Minggu 4	Minggu 5
Business Understanding					
Data Understanding					
Data Cleaning & Preparation					
Label Construction					
Modeling					
Evaluation & Improvement					
Final Integration & Reporting					
Presentation					

VI. DATA UNDERSTANDING

1. Pengumpulan Data

Tahap Data Understanding berfokus pada eksplorasi untuk mengumpulkan informasi terkait dataset yang digunakan dalam pengerjaan proyek ini. Tahapan ini dilakukan untuk memahami karakteristik data secara menyeluruh.

Langkah awal dalam tahap persiapan data adalah memuat keempat dataset utama yang disediakan, yaitu:

a. GLC25_PA_metadata_train.csv (Presence-Absence Train),
b. GLC25_P0_metadata_train.csv (Presence-Only Train),
c. GLC25-PA-train-soilgrids.csv
d. GLC25-PO-train-soilgrids.csv

Pada tahap ini, data dari berbagai file CSV dimuat dan diperiksa secara awal untuk memastikan bahwa struktur fitur di setiap dataset serupa, serta untuk mempermudah pengunduhan data yang relevan.

a. Load Data

```
import pandas as pd

# Load the datasets with the new names
PA_metadata_train_file = 'GLC25_PA_metadata_train.csv'
P0_metadata_train_file = 'GLC25_P0_metadata_train.csv'
PA_soil_train_file = 'GLC25-PA-train-soilgrids.csv'
P0_soil_train_file = 'GLC25-PO-train-soilgrids.csv'
```

Untuk meload data, maka hal yang dilakukan adalah mengimpor pustaka Pandas dan mendefinisikan variabel yang berisi nama file CSV yaitu PA dan PO.

b. Memilih Jumlah Data

```
PA_metadata_train_top500 = PA_metadata_train.head(500)
P0_metadata_train_top500 = P0_metadata_train.head(500)
PA_soil_train_top500 = PA_soil_train.head(500)
P0_soil_train_top500 = P0_soil_train.head(500)
```

Karena jumlah data yang sangat besar, maka data yang digunakan adalah sebanyak 500 data teratas. Untuk pengambilan data tersebut, code yang digunakan adalah .head(500), kode ini membatasi data yang digunakan hanya sebanyak 500 baris teratas dari masing-masing dataset untuk efisiensi pemrosesan dan analisis.

2. Penelaahan Data

Pada tahap awal eksplorasi, dilakukan pemeriksaan terhadap empat dataset yang telah pilih dan telah dikurangi sebelumnya yaitu PA_metadata_train_top500, PO_metadata_train_top500, PA_soil_train_top500, dan PO_soil_train_top500. Pemeriksaan mencakup struktur data dengan .info(), tampilan awal data dengan .head(), serta identifikasi nilai hilang menggunakan .isnull().sum(). Langkah ini bertujuan memastikan data siap digunakan untuk analisis atau pemodelan.

a. Memeriksa Struktur Data

```
# Memeriksa struktur dataset
PA_metadata_train_top500.info()
P0_metadata_train_top500.info()
PA_soil_train_top500.info()
P0_soil_train_top500.info()
```

Kode ini digunakan untuk memeriksa struktur (informasi) dari empat dataset yang berbeda: PA_metadata_train_top500, P0_metadata_train_top500, PA_soil_train_top500, dan P0_soil_train_top500. Fungsi .info() memberikan rincian tentang jumlah entri, tipe data, dan jumlah nilai yang tidak kosong dalam setiap kolom dataset tersebut.

VII. VALIDASI DATA

Pada tahap validasi data, dilakukan evaluasi untuk memastikan dataset yang digunakan dalam proyek memiliki kualitas dan kelengkapan yang baik. Proses ini mencakup pengecekan atribut yang tidak lengkap, pembersihan data untuk menjaga konsistensi dan relevansi, serta penyederhanaan jumlah dan kompleksitas data. Selain itu, dilakukan pengecekan nilai untuk memastikan semua nilai masuk akal, ejaannya konsisten, dan fitur dengan nilai berbeda memiliki makna yang sesuai.

VIII. DATA PREPARATION

1. Penanganan Missing Value

Dalam tahap persiapan data, langkah pertama yang dilakukan adalah mengidentifikasi keberadaan nilai yang hilang (missing values) pada dataset. Proses ini dilakukan dengan menggunakan fungsi .isnull() untuk mendeteksi nilai yang hilang dalam setiap kolom, yang kemudian dihitung jumlahnya menggunakan metode .sum(). Hasil dari perhitungan ini memberikan informasi terkait jumlah nilai yang hilang pada masing-masing kolom, yang kemudian ditampilkan untuk dianalisis lebih lanjut.

```
# Memeriksa nilai yang hilang pada setiap kolom
PA_metadata_train_top500_missing = PA_metadata_train_top500.isnull().sum()
P0_metadata_train_top500_missing = P0_metadata_train_top500.isnull().sum()
PA_soil_train_top500_missing = PA_soil_train_top500.isnull().sum()
P0_soil_train_top500_missing = P0_soil_train_top500.isnull().sum()

# Menampilkan jumlah nilai yang hilang per kolom
print("Nilai yang hilang pada PA_metadata_train_top500:")
print(PA_metadata_train_top500_missing)

print("Nilai yang hilang pada P0_metadata_train_top500:")
print(P0_metadata_train_top500_missing)

print("Nilai yang hilang pada PA_soil_train_top500:")
print(PA_soil_train_top500_missing)

print("Nilai yang hilang pada P0_soil_train_top500:")
print(P0_soil_train_top500_missing)
```

2. Imputasi Nilai yang Hilang Menggunakan Mean dan Median

Langkah selanjutnya yang dilakukan adalah penanganan terhadap nilai yang hilang (missing values) dengan menggunakan teknik imputasi. Pada dataset PA_metadata_train_top500, nilai yang hilang pada kolom geoUncertaintyInM diisi dengan nilai rata-rata (mean) dari kolom tersebut, sementara nilai yang hilang pada kolom areaInM2 diisi dengan nilai median. Teknik imputasi ini dilakukan menggunakan metode .fillna(), yang menggantikan nilai hilang dengan nilai yang dihitung dari kolom yang bersangkutan, baik itu mean maupun median

```
# Menggunakan mean dan median untuk mengisi missing value pada PA metadata
PA_metadata_train_top500.loc[:, 'geoUncertaintyInM'] = PA_metadata_train_top500['geoUncertaintyInM'].fillna(PA_metadata_train_top500['geoUncertaintyInM'
PA_metadata_train_top500.loc[:, 'areaInM2'] = PA_metadata_train_top500['areaInM2'].fillna(PA_metadata_train_top500['areaInM2'].median())
```

Pada tahap ini, missing values dalam dataset PA_soil_train_top500 dan P0_soil_train_top500 diatasi melalui teknik imputasi menggunakan nilai rata-rata (mean) untuk setiap kolom yang mengandung nilai hilang. Proses ini dimulai dengan membuat salinan dari dataset untuk menjaga integritas data asli. Selanjutnya, kolom-kolom yang memiliki nilai hilang, seperti Soilgrid-bdod, Soilgrid-cec, Soilgrid-cfvo, dan lainnya, diimputasi dengan nilai rata-rata yang dihitung untuk setiap kolom tersebut. Fungsi .fillna() digunakan untuk menggantikan nilai hilang dengan nilai rata-rata yang dihitung menggunakan metode .mean(). Imputasi dengan rata-rata ini dipilih karena cocok digunakan pada kolom dengan distribusi data yang simetris dan tidak terpengaruh oleh nilai ekstrem. Tujuan dari imputasi ini adalah untuk memastikan kelengkapan data, sehingga dataset yang digunakan dalam analisis dan pemodelan tetap utuh dan bebas dari gangguan yang disebabkan oleh nilai yang hilang, yang pada akhirnya mendukung pembuatan model yang lebih akurat dan dapat diandalkan.

```
# Mengatasi Nilai yang Hilang pada PA_soil_train_top500
PA_soil_train_top500 = PA_soil_train_top500.copy()

# Lanjutkan imputasi
PA_soil_train_top500['Soilgrid-bdod'] = PA_soil_train_top500['Soilgrid-bdod'].fillna(PA_soil_train_top500['Soilgrid-bdod'].mean())
PA_soil_train_top500['Soilgrid-cec'] = PA_soil_train_top500['Soilgrid-cec'].fillna(PA_soil_train_top500['Soilgrid-cec'].mean())
PA_soil_train_top500['Soilgrid-cfvo'] = PA_soil_train_top500['Soilgrid-cfvo'].fillna(PA_soil_train_top500['Soilgrid-cfvo'].mean())
PA_soil_train_top500['Soilgrid-clay'] = PA_soil_train_top500['Soilgrid-clay'].fillna(PA_soil_train_top500['Soilgrid-clay'].mean())
PA_soil_train_top500['Soilgrid-nitrogen'] = PA_soil_train_top500['Soilgrid-nitrogen'].fillna(PA_soil_train_top500['Soilgrid-nitrogen'].mean())
PA_soil_train_top500['Soilgrid-phh2o'] = PA_soil_train_top500['Soilgrid-phh2o'].fillna(PA_soil_train_top500['Soilgrid-phh2o'].mean())
PA_soil_train_top500['Soilgrid-sand'] = PA_soil_train_top500['Soilgrid-sand'].fillna(PA_soil_train_top500['Soilgrid-sand'].mean())
PA_soil_train_top500['Soilgrid-silt'] = PA_soil_train_top500['Soilgrid-silt'].fillna(PA_soil_train_top500['Soilgrid-silt'].mean())
PA_soil_train_top500['Soilgrid-soc'] = PA_soil_train_top500['Soilgrid-soc'].fillna(PA_soil_train_top500['Soilgrid-soc'].mean())
```

```
# Mengatasi Nilai yang Hilang pada P0_soil_train_top500
P0_soil_train_top500_copy = P0_soil_train_top500.copy()

# Imputasi nilai yang hilang pada P0_soil_train_top500_copy dengan rata-rata
P0_soil_train_top500_copy['Soilgrid-bdod'] = P0_soil_train_top500_copy['Soilgrid-bdod'].fillna(P0_soil_train_top500_copy['Soilgrid-bdod'].mean())
P0_soil_train_top500_copy['Soilgrid-cec'] = P0_soil_train_top500_copy['Soilgrid-cec'].fillna(P0_soil_train_top500_copy['Soilgrid-cec'].mean())
P0_soil_train_top500_copy['Soilgrid-cfvo'] = P0_soil_train_top500_copy['Soilgrid-cfvo'].fillna(P0_soil_train_top500_copy['Soilgrid-cfvo'].mean())
P0_soil_train_top500_copy['Soilgrid-clay'] = P0_soil_train_top500_copy['Soilgrid-clay'].fillna(P0_soil_train_top500_copy['Soilgrid-clay'].mean())
P0_soil_train_top500_copy['Soilgrid-nitrogen'] = P0_soil_train_top500_copy['Soilgrid-nitrogen'].fillna(P0_soil_train_top500_copy['Soilgrid-nitrogen'].me
P0_soil_train_top500_copy['Soilgrid-phh2o'] = P0_soil_train_top500_copy['Soilgrid-phh2o'].fillna(P0_soil_train_top500_copy['Soilgrid-phh2o'].mean())
P0_soil_train_top500_copy['Soilgrid-sand'] = P0_soil_train_top500_copy['Soilgrid-sand'].fillna(P0_soil_train_top500_copy['Soilgrid-sand'].mean())
P0_soil_train_top500_copy['Soilgrid-silt'] = P0_soil_train_top500_copy['Soilgrid-silt'].fillna(P0_soil_train_top500_copy['Soilgrid-silt'].mean())
P0_soil_train_top500_copy['Soilgrid-soc'] = P0_soil_train_top500_copy['Soilgrid-soc'].fillna(P0_soil_train_top500_copy['Soilgrid-soc'].mean())
```

3. Penggabungan Dataset Tanah

Pada tahap ini, dilakukan penggabungan (merge) antara dua dataset, yaitu PA_metadata_train_top500 dan PA_soil_train_top500, berdasarkan kolom yang memiliki hubungan antara keduanya, yaitu surveyId. Dataset PA_metadata_train_top500 dipilih untuk mencakup kolom-kolom seperti lat, lon, geoUncertaintyInM, speciesId, dan surveyId, sementara PA_soil_train_top500 mencakup kolom-kolom yang berisi data tanah seperti Soilgrid-bdod, Soilgrid-cec, Soilgrid-cfvo, dan sebagainya. Penggabungan ini dilakukan dengan metode inner join, yang hanya akan mencocokkan baris yang memiliki nilai surveyId yang sama di kedua dataset. Hasil penggabungan ini kemudian disimpan ke dalam file CSV untuk digunakan lebih lanjut dalam analisis atau pemodelan. Tujuan dari penggabungan ini adalah untuk menyatukan informasi metadata dengan data tanah, sehingga menghasilkan dataset yang lebih lengkap dan kaya informasi yang siap untuk langkah analisis berikutnya.

```
# Menggabungkan PA metadata dan PA soil berdasarkan surveyId
PA_metadata_selected = PA_metadata_train_top500[['lat', 'lon', 'geoUncertaintyInM', 'speciesId', 'surveyId']]
PA_soil_selected = PA_soil_train_top500[['surveyId', 'Soilgrid-bdod', 'Soilgrid-cec', 'Soilgrid-cfvo',
                                         'Soilgrid-clay', 'Soilgrid-nitrogen', 'Soilgrid-phh2o',
                                         'Soilgrid-sand', 'Soilgrid-silt', 'Soilgrid-soc']]

# Gabungkan PA metadata dengan PA soil berdasarkan surveyId
PA_merged = pd.merge(PA_metadata_selected, PA_soil_selected, on='surveyId', how='inner')

# Menampilkan hasil gabungan PA metadata dan PA soil
print(PA_merged.head())

# Langkah 3: Menyimpan hasil gabungan ke file CSV
PA_merged.to_csv('PA_merged.csv', index=False)
print("Data yang telah digabungkan telah disimpan.")
```

```
Empty DataFrame
Columns: [lat, lon, geoUncertaintyInM, speciesId, surveyId, Soilgrid-bdod, Soilgrid-cec, Soilgrid-cfvo, Soilgrid-clay, Soil
grid-nitrogen, Soilgrid-phh2o, Soilgrid-sand, Soilgrid-silt, Soilgrid-soc]
Index: []
Data yang telah digabungkan telah disimpan.
```

Pada tahap ini, dilakukan penggabungan antara dua dataset, yaitu P0_metadata_train_top500 dan P0_soil_train_top500, berdasarkan kolom yang sama, yakni surveyId. Dataset P0_metadata_train_top500 dipilih untuk menyertakan kolom-kolom seperti lat, lon, geoUncertaintyInM, speciesId, dan surveyId, sementara dataset P0_soil_train_top500 berisi kolom-kolom terkait informasi tanah, seperti Soilgrid-bdod, Soilgrid-cec, Soilgrid-cfvo, dan seterusnya. Penggabungan ini dilakukan dengan metode inner join, yang memastikan hanya data dengan surveyId yang sesuai dari kedua dataset yang akan digabungkan. Setelah penggabungan selesai, hasilnya disimpan dalam file CSV menggunakan fungsi .to_csv(), yang memungkinkan data tersebut untuk digunakan pada langkah analisis atau pemodelan selanjutnya. Tujuan dari penggabungan ini adalah untuk menciptakan dataset yang lebih lengkap dengan menggabungkan informasi metadata dan data tanah yang relevan.

```
# Menggabungkan PO metadata dan PO soil berdasarkan surveyId
P0_metadata_selected = P0_metadata_train_top500[['lat', 'lon', 'geoUncertaintyInM', 'speciesId', 'surveyId']]
P0_soil_selected = P0_soil_train_top500[['surveyId', 'Soilgrid-bdod', 'Soilgrid-cec', 'Soilgrid-cfvo',
                                         'Soilgrid-clay', 'Soilgrid-nitrogen', 'Soilgrid-phh2o',
                                         'Soilgrid-sand', 'Soilgrid-silt', 'Soilgrid-soc']]

# Gabungkan PO metadata dengan PO soil berdasarkan surveyId
PO_merged = pd.merge(P0_metadata_selected, P0_soil_selected, on='surveyId', how='inner')

# Menampilkan hasil gabungan PO metadata dan PO soil
print(PO_merged.head())

# Langkah 3: Menyimpan hasil gabungan ke file CSV
PO_merged.to_csv('PO_merged.csv', index=False)
print("Data yang telah digabungkan telah disimpan.")
```

```
        lat       lon  geoUncertaintyInM  speciesId  surveyId  Soilgrid-bdod  \
0  43.74605  1.573057                6.0     3383.0         1          145.0
1  42.12559  0.314948                5.0     1152.0         2          134.0
2  48.29520 -0.934518               24.9     6772.0         3          127.0
3  53.63367 -2.644535                8.0     3318.0         4          103.0
4  49.79471  7.925086               15.0     3374.0         5          138.0

   Soilgrid-cec  Soilgrid-cfvo  Soilgrid-clay  Soilgrid-nitrogen  \
0         212.0          130.0          303.0              134.0
1         213.0          189.0          302.0              184.0
2         211.0          109.0          187.0              257.0
3         212.0          123.0          216.0              466.0
4         195.0          131.0          310.0              198.0

   Soilgrid-phh2o  Soilgrid-sand  Soilgrid-silt  Soilgrid-soc
0            73.0          306.0          389.0         122.0
1            74.0          294.0          403.0         223.0
2            59.0          132.0          681.0         301.0
3            56.0          429.0          355.0        1044.0
4            73.0          208.0          480.0         208.0
Data yang telah digabungkan telah disimpan.
```

4. Seleksi Fitur Dataset

```
# Mencari fitur yang akan digunakan untuk train

# Menampilkan kolom-kolom yang ada pada PA_merged dan PO_merged
PA_columns = PA_merged.columns
PO_columns = PO_merged.columns

# Mencari kolom yang sama antara PA_merged dan PO_merged
common_columns = PA_columns.intersection(PO_columns)

# Menampilkan kolom yang sama
print("Kolom yang sama antara PA_merged dan PO_merged:")
print(common_columns)
```

```
Kolom yang sama antara PA_merged dan PO_merged:
Index(['lat', 'lon', 'geoUncertaintyInM', 'speciesId', 'surveyId',
       'Soilgrid-bdod', 'Soilgrid-cec', 'Soilgrid-cfvo', 'Soilgrid-clay',
       'Soilgrid-nitrogen', 'Soilgrid-phh2o', 'Soilgrid-sand', 'Soilgrid-silt',
       'Soilgrid-soc'],
      dtype='object')
```

Pada tahap ini, dilakukan seleksi fitur yang akan digunakan dalam proses pelatihan model dengan mencari kolom-kolom yang terdapat pada kedua dataset, PA_merged dan PO_merged. Kolom-kolom dari masing-masing dataset disimpan dalam variabel PA_columns dan PO_columns, kemudian kolom yang sama antara kedua dataset ditemukan menggunakan metode .intersection(). Hasil dari seleksi ini adalah kolom-kolom yang ada pada kedua dataset dan dapat digunakan untuk analisis atau pelatihan model.

5. Pemilihan dan Penggabungan Data

Kolom-kolom yang relevan dari dataset PA_merged dan PO_merged dipilih untuk memastikan hanya fitur yang diperlukan yang digunakan dalam analisis lebih lanjut. Kolom-kolom yang sama pada kedua dataset dipilih, yaitu lat, lon, geoUncertaintyInM, speciesId, surveyId, serta beberapa kolom terkait tanah seperti Soilgrid-bdod, Soilgrid-cec, dan lainnya. Dataset yang telah dipilih kolom-kolomnya kemudian digabungkan secara vertikal menggunakan fungsi pd.concat() untuk membentuk dataset baru yang lebih komprehensif, yaitu final_merged_data. Hasil penggabungan ini kemudian disimpan ke dalam file CSV untuk digunakan lebih lanjut. Tujuan dari langkah ini adalah untuk mengonsolidasikan data yang relevan dari kedua dataset agar dapat dianalisis secara menyeluruh, sekaligus memastikan bahwa data yang digunakan memiliki kolom yang seragam dan konsisten.

```
# Pilih hanya kolom yang relevan (kolom yang sama antara PA_merged dan PO_merged)
common_columns = ['lat', 'lon', 'geoUncertaintyInM', 'speciesId', 'surveyId',
                  'Soilgrid-bdod', 'Soilgrid-cec', 'Soilgrid-cfvo', 'Soilgrid-clay',
                  'Soilgrid-nitrogen', 'Soilgrid-phh2o', 'Soilgrid-sand', 'Soilgrid-silt',
                  'Soilgrid-soc']

PA_merged_selected = PA_merged[common_columns]
PO_merged_selected = PO_merged[common_columns]

# Menggabungkan PA_merged_selected dan PO_merged_selected secara vertikal
final_merged_data = pd.concat([PA_merged_selected, PO_merged_selected], axis=0, ignore_index=True)

# Menampilkan beberapa baris pertama dari final_merged_data untuk memastikan hasil gabungan
print(final_merged_data.head())

# Menyimpan final_merged_data ke dalam file CSV
final_merged_data.to_csv('final_merged_data.csv', index=False)
print("Data yang telah digabungkan dan disimpan sebagai final_merged_data.csv.")
```

```
        lat       lon  geoUncertaintyInM  speciesId  surveyId  Soilgrid-bdod  \
0  43.74605  1.573057                6.0     3383.0         1          145.0
1  42.12559  0.314948                5.0     1152.0         2          134.0
2  48.29520 -0.934518               24.9     6772.0         3          127.0
3  53.63367 -2.644535                8.0     3318.0         4          103.0
4  49.79471  7.925086               15.0     3374.0         5          138.0

   Soilgrid-cec  Soilgrid-cfvo  Soilgrid-clay  Soilgrid-nitrogen  \
0         212.0          130.0          303.0              134.0
1         213.0          189.0          302.0              184.0
2         211.0          109.0          187.0              257.0
3         212.0          123.0          216.0              466.0
4         195.0          131.0          310.0              198.0

   Soilgrid-phh2o  Soilgrid-sand  Soilgrid-silt  Soilgrid-soc
0            73.0          306.0          389.0         122.0
1            74.0          294.0          403.0         223.0
2            59.0          132.0          681.0         301.0
3            56.0          429.0          355.0        1044.0
4            73.0          208.0          480.0         208.0
Data yang telah digabungkan dan disimpan sebagai final_merged_data.csv.
```

IX. MODELING

1. Membangun Skenario Pengujian

Pada bagian ini, dijelaskan mengenai scenario pengujian yang digunakan untuk mengevaluasi performa model. Input yang digunakan adalah dataset final_merged_data, dengan fitur yang dipilih adalah semua kolom kecuali speciesId dan surveyId. Kolom speciesId menjadi target yang diprediksi, sementara surveyId digunakan untuk mengelompokkan hasil prediksi.

Dataset dibagi menjadi dua bagian, yaitu 80% untuk pelatihan dan 20% untuk pengujian, menggunakan metode train_test_split. Model dilatih pada data pelatihan dan diuji pada data pengujian. Akurasi model dihitung dengan membandingkan hasil prediksi dengan nilai sebenarnya menggunakan metrik accuracy_score.

Tujuan utama dari pengujian ini adalah untuk mengevaluasi akurasi model dalam memprediksi speciesId. Hasil prediksi disimpan dalam file CSV untuk analisis lebih lanjut. Model yang digunakan adalah Random Forest Classifier dengan 100 estimator dan random state untuk memastikan hasil yang dapat direproduksi.

2. Membangun model

Pada tahap ini, dilakukan pembangunan model prediksi menggunakan algoritma Random Forest untuk memprediksi nilai kolom speciesId berdasarkan fitur-fitur yang tersedia dalam dataset.

a. Persiapan Data dan Pemisahan Dataset

Melalui data preparation yang telah dilakukan, maka data akhir yang telah disiapkan akan digunakan untuk proses modeling.

```
# Load the datasets (already done, using final_merged_data here for training)
train_data = pd.read_csv('final_merged_data.csv')

# Prepare feature columns (excluding the target column 'speciesId')
X = train_data.drop(columns=['speciesId', 'surveyId'])

# The target is the 'speciesId' column
y = train_data['speciesId']
```

Langkah awal adalah memuat dataset final_merged_data.csv, kemudian memisahkan kolom speciesId sebagai target (y) dan kolom lainnya sebagai fitur (X), yang digunakan untuk melatih model.

b. Pemisahan Data untuk Training dan Testing

Pada bagian ini, kita menyaring kelas yang hanya memiliki satu sampel. Ini membantu menghindari masalah dengan kelas yang terlalu jarang atau tidak seimbang.

```
# Remove classes with only one sample (optional)
train_data_filtered = train_data[train_data['speciesId'].isin(y.value_counts()[y.value_counts() > 1].index)]

# Prepare filtered data
X_filtered = train_data_filtered.drop(columns=['speciesId', 'surveyId'])
y_filtered = train_data_filtered['speciesId']
```

Pada bagian ini, y.value_counts() digunakan untuk menghitung frekuensi kemunculan setiap nilai dalam kolom speciesId. Hal ini membantu untuk mengetahui kelas mana yang memiliki jumlah sampel yang sangat sedikit. Kelas yang hanya memiliki satu sampel disaring dari dataset agar tidak mempengaruhi hasil model, karena kelas-kelas tersebut tidak cukup representatif untuk dilibatkan dalam pelatihan model. Setelah proses penyaringan dilakukan, dataset yang telah difilter kemudian dipisahkan kembali menjadi fitur (X_filtered) dan target (y_filtered), yang akan digunakan untuk pelatihan model selanjutnya. Dengan cara ini, model dapat dilatih hanya dengan data yang memiliki distribusi kelas yang lebih seimbang.

c. StratifiedKFold dan GridSearchCV

Pada bagian ini, kita melakukan pembagian data menggunakan StratifiedKFold yang menjaga distribusi kelas pada setiap fold dalam cross-validation. Kemudian kita melakukan pencarian grid untuk memilih hyperparameter terbaik.

```
# Use StratifiedKFold with the adjusted n_splits based on the smallest class size
n_splits = min(5, y_filtered.value_counts().min())
sss = StratifiedKFold(n_splits=n_splits, shuffle=True, random_state=42)

# Initialize RandomForest model
model = RandomForestClassifier(random_state=42)

# Hyperparameter grid for tuning
param_grid = {
    'n_estimators': [100, 200, 500],
    'max_depth': [None, 10, 20, 30],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2']
}

# Perform GridSearchCV
grid_search = GridSearchCV(model, param_grid, cv=sss, n_jobs=-1, verbose=2)
grid_search.fit(X_filtered, y_filtered)
```

Penjelasan Code :

a) StratifiedKFold digunakan untuk membagi data ke dalam 5 fold (atau jumlah minimum yang sesuai dengan jumlah sampel terkecil dari kelas) sambil memastikan bahwa distribusi kelas tetap proporsional di setiap fold.

b) RandomForestClassifier adalah model yang digunakan untuk klasifikasi. Kami menyiapkan beberapa hyperparameter yang berbeda untuk dicoba menggunakan GridSearchCV.
c) param_grid berisi berbagai nilai yang akan dicoba untuk hyperparameter seperti jumlah pohon keputusan (n_estimators), kedalaman maksimum pohon (max_depth), serta jumlah sampel minimum yang diperlukan untuk melakukan pembagian pada setiap node pohon (min_samples_split dan min_samples_leaf).
d) GridSearchCV melakukan pencarian grid dengan menggunakan StratifiedKFold sebagai strategi validasi silang dan mencari kombinasi terbaik dari hyperparameter yang telah ditentukan.

d. Pelatihan dan Evaluasi Model

Setelah mendapatkan model terbaik dari GridSearchCV, model tersebut dilatih dengan data yang telah difilter, dan dilakukan prediksi untuk mengevaluasi akurasinya.

```
# Best model
best_model = grid_search.best_estimator_

# Train the best model
best_model.fit(X_filtered, y_filtered)

# Predictions and evaluation
y_pred = best_model.predict(X_filtered)

# Calculate accuracy
accuracy = accuracy_score(y_filtered, y_pred)
print(f"Accuracy with tuned model: {accuracy:.3f}")
```

Setelah GridSearchCV selesai, model terbaik dipilih dengan grid_search.best_estimator_. Model ini dilatih dengan data yang sudah difilter menggunakan .fit(). Setelah model dilatih, kita melakukan prediksi pada data tersebut dan menghitung akurasi menggunakan accuracy_score, yang memberikan proporsi data yang diprediksi dengan benar oleh model.

e. Visualisasi Pentingnya Fitur

Setelah evaluasi selesai, hasil prediksi disusun dalam format yang diperlukan untuk pengajuan, seperti dalam kompetisi.

```
# Feature importance plot
feature_importances = best_model.feature_importances_
plt.figure(figsize=(10, 6))
plt.barh(X_filtered.columns, feature_importances)
plt.xlabel("Feature Importance")
plt.title("Feature Importance from RandomForest")
plt.show()
```

feature_importances_ mengembalikan skor pentingnya masing-masing fitur yang digunakan oleh model RandomForest. Plot barh menampilkan fitur-fitur yang lebih penting di sebelah kiri dan yang kurang penting di sebelah kanan. Ini membantu kita memahami fitur mana yang lebih berpengaruh dalam membuat prediksi.

X. EVALUASI MODEL

F1 Score adalah salah satu metrik evaluasi yang digunakan untuk mengukur kinerja model klasifikasi, terutama ketika dataset memiliki distribusi kelas yang tidak seimbang. F1 Score menggabungkan dua metrik penting, yaitu Precision dan Recall, untuk memberikan gambaran yang lebih komprehensif tentang kemampuan model dalam memprediksi kelas positif.

```
from sklearn.metrics import f1_score

# Predictions and evaluation
y_pred = best_model.predict(X_filtered)

# Calculate F1-score (weighted) for the overall model
f1 = f1_score(y_filtered, y_pred, average='weighted')
print(f"Weighted F1-score: {f1:.3f}")
```

Untuk menghitung nilai F1 Score menggunakan metode f1_score dari pustaka sklearn.metrics. , kita pertama-tama melakukan prediksi menggunakan model terbaik yang telah dilatih (best_model.predict(X_filtered)) pada data yang telah difilter (X_filtered). F1 Score di sini dihitung dengan parameter average='weighted', yang berarti F1 Score dihitung untuk setiap kelas, kemudian dihitung rata-rata berbobot berdasarkan jumlah data yang ada pada masing-masing kelas. Ini memberikan penekanan lebih pada kelas yang lebih banyak muncul dalam dataset, sehingga metrik ini cocok untuk digunakan pada masalah klasifikasi dengan kelas yang tidak seimbang.

a. y_test merupakan nilai asli atau target kelas yang benar untuk data pengujian.
b. y_pred merupakan nilai prediksi yang dihasilkan oleh model untuk data pengujian.
c. average='weighted', parameter ini menginstruksikan fungsi f1_score untuk menghitung F1 Score untuk setiap kelas, lalu menghitung rata-rata berbobot sesuai dengan jumlah data di setiap kelas. Dengan ini, kelas dengan lebih banyak contoh memiliki pengaruh lebih besar terhadap skor akhir.

Setelah perhitungan F1 Score selesai, hasilnya dicetak dengan format tiga angka di belakang koma menggunakan print. Ini memberikan gambaran mengenai seberapa baik model mengklasifikasikan data, dengan mempertimbangkan baik precision maupun recall pada setiap kelas.

X. DEPLOYMENT

Deployment merupakan proses penting dalam implementasi model machine learning ke dalam sistem operasional, sehingga model tersebut dapat digunakan secara langsung untuk menghasilkan prediksi dalam konteks dunia nyata. Pada proyek ini, deployment dilakukan untuk memastikan model yang telah dilatih dapat dimanfaatkan secara optimal dalam memprediksi spesies tumbuhan yang kemungkinan besar terdapat pada suatu lokasi geografis tertentu.

Beberapa alasan utama pentingnya tahap deployment antara lain:

1. Efektivitas implementasi: Deployment memungkinkan model yang telah dikembangkan untuk diterapkan

secara langsung dalam sistem prediksi, sehingga pengguna dapat memperoleh hasil secara real-time.

2. Maksimalisasi nilai model: Dengan mengintegrasikan model ke dalam sistem dan menyalurkan hasil prediksi ke komponen lain, nilai praktis dari model dapat dimaksimalkan dalam pengambilan keputusan atau aplikasi nyata.

Namun demikian, proses deployment model juga menghadapi sejumlah tantangan:

1. Tantangan Umum pada Perangkat Lunak:
    - Reliability (keandalan): Menjamin bahwa model dapat berjalan dengan stabil tanpa gangguan.
    - Reusability (dapat digunakan ulang): Memungkinkan penerapan kembali model untuk skenario serupa di masa depan.
    - Maintainability (kemudahan pemeliharaan): Memfasilitasi pemeliharaan dan pembaruan model secara efisien.
    - Flexibility (fleksibilitas): Mengakomodasi perubahan kebutuhan atau integrasi dengan sistem yang berbeda.
2. Tantangan Spesifik pada Machine Learning:
    - Reproducibility (reproduksibilitas): Menjamin bahwa hasil prediksi tetap konsisten ketika model dijalankan di lingkungan yang berbeda.

Proses prediksi dalam deployment model ini dilakukan dengan memanfaatkan empat jenis data utama:

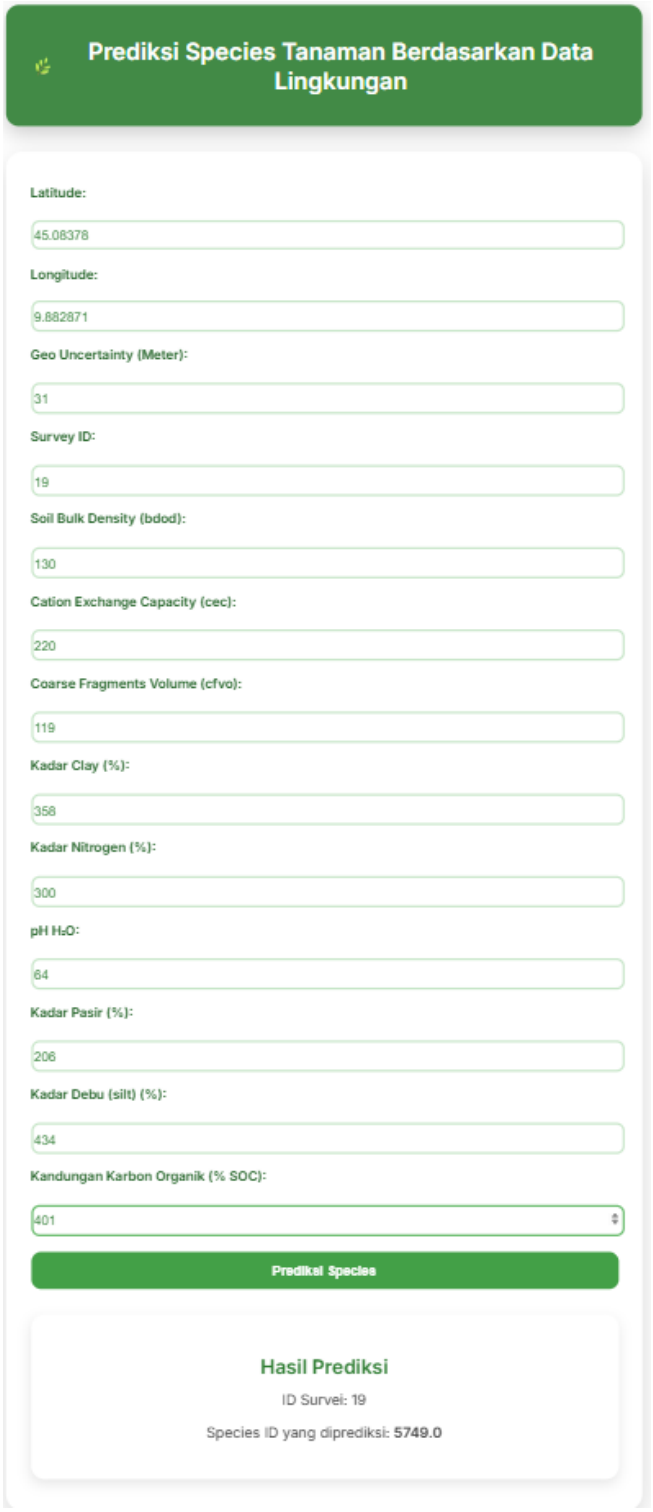
1. Presence-Absence (PA): Data ini menjadi fondasi utama karena menyediakan informasi tentang kehadiran maupun ketidakhadiran spesies di lokasi tertentu.
2. Presence-Only (PO): Digunakan sebagai pelengkap untuk memperkaya informasi distribusi spasial spesies, meskipun hanya mencakup data kehadiran.
3. GLC25-PA-train-soilgrids dan GLC25-PO-train-soilgrids: Menyediakan fitur lingkungan dan karakteristik tanah, seperti pH, kelembapan, serta kandungan nitrogen dan karbon organik, yang sangat berpengaruh terhadap distribusi spesies.

Keempat dataset ini digabungkan berdasarkan survey_id, dengan species_id sebagai target prediksi. Proses penggabungan ini memungkinkan model untuk memanfaatkan informasi lokasi, waktu observasi, serta kondisi lingkungan dan tanah secara menyeluruh. Dengan integrasi ini, model dapat mengenali pola distribusi spesies secara lebih akurat.

Model yang dibangun dievaluasi menggunakan metrik micro F1-score, yaitu ukuran kinerja yang mempertimbangkan keseimbangan antara precision dan recall di seluruh sampel. Penggunaan metrik ini bertujuan untuk menilai seberapa tepat model dalam memprediksi kumpulan spesies yang benar-benar hadir di tiap lokasi survei.

Melalui proses deployment, model ini tidak hanya digunakan sebagai prototipe analisis, tetapi juga telah diintegrasikan ke dalam sistem yang mampu memberikan prediksi secara langsung, memperkuat aplikasi praktis dari penelitian ini dalam konservasi dan studi keanekaragaman hayati.

Berikut gambar model yang telah berhasil di deployment, yang dimana masih memiliki banyak kekurangan dalam pencocokan speciesId tumbuhan tersebut.



XI. KESIMPULAN

Melalui penelitian ini, dikembangkan model prediksi keberadaan spesies tumbuhan menggunakan algoritma Random Forest dengan data lingkungan seperti iklim, citra satelit, dan karakteristik tanah. Model ini dapat membantu dalam pelestarian keanekaragaman hayati dengan memberikan prediksi yang akurat mengenai distribusi spesies. Meskipun model ini menunjukkan akurasi tinggi, tantangan tetap ada dalam interpretabilitas dan pencocokan spesies. Tahap deployment telah berhasil, namun masih perlu penyempurnaan untuk hasil yang lebih optimal.

XII. REFERENSI

[1] S. B. Phillips, V. P. Aneja, D. Kang, and S. P. Arya, "Modelling and analysis of the atmospheric nitrogen deposition in North Carolina," Int. J. Glob. Environ. Issues, vol. 6, no. 2–3, pp. 231–252, 2006, doi: 10.1016/j.ecolmodel.2005.03.026.

[2] J. Elith, S. J. Phillips, T. Hastie, M. Dudík, Y. E. Chee, and C. J. Yates, "A statistical explanation of MaxEnt for ecologists," Divers. Distrib., vol. 17, no. 1, pp. 43–57, 2011, doi: 10.1111/j.1472-4642.2010.00725.x.

[3] A. Wasniewski, A. Hoscilo, B. Zagajewski, and D. Moukétou-Tarazewicz, "Assessment of sentinel-2 satellite images and random forest classifier for rainforest mapping in Gabon," Forests, vol. 11, no. 9, pp. 1–17, 2020, doi: 10.3390/f11090941.

XIII. KONTRIBUSI PENULIS

Berikut pernyataan kontribusi penulis dalam proses pembuatan paper ini:

"Yohana: GitHub, article, dokumen, Data Understanding, Data Preparation, Modeling, Evaluasi

Griselda: GitHub, article, dokumen, Data Understanding, Data Preparation, Modeling, Evaluasi

Lenna : GitHub, article, dokumen, Pendahuluan, Data Understanding, Modeling, Deployment "